1. **Вопрос:** Схема развития ИС

**Ответ:** Схема предназначена для формирования взгляда на архитектуру ИС с точки зрения участников ее разработки. При моделировании бизнеса рассматривается 3 аспекта:

* Объекты, с которыми оперирует бизнес (данные);
* Процессы, которые он выполняет (функции);
* События, управляющие изменениями процессов и объектов.

Можно определить след.типы моделирования:

* Информационное;
* Функциональное;
* Событийное.

В схеме Захмана каждой строке соответствует точка зрения одного из участников проекта по созданию системы. Точка зрения отображает значения и области ответственности заинтересованных лиц в процессе создания системы.

В основе схемы Захмана рассматривается 3 аспекта, каждому из которых соответствует колонка (или столбец), каждому аспекту соответствуют разные методы формирования представления.

В каждой ячейке представлен вид конечного продукта (архитектурное представление) с точки зрения некоторой группы лиц, участвующих в разработке системы:

* 1, 2 – заказчик, видит систему общих стратегических и тактических аспектов
* 3 – проектировщик, его представление является проектирование системы обеспечивающей удовлетворение требований заказчика, к которой должна быть добавлена точность необходимая для тех, кто будет реализовывать систему.
* 4, 5 – разработчик, его взгляд отражает множество решений ограниченных технологией, временем и стоимостью.
* 6 – пользователь, в каждой ячейки схемы представлен вид конечного продукта с точки зрения некоторой групп лиц участвующей в разработки системы.



*В подходе Захмана не делается акцент на динамике развития ИС. При переходе от одной строки таблицы к другой меняется лишь точка зрения, с которой рассматривается система, причем эта точка зрения не обязана быть связана с уровнем детальности рассмотрения. В схеме Захмана отражены три аспекта системы, приблизительно соответствующие некоторым методикам моделирования (информационное моделирование, диаграмма потоков данных и т.д.).*

1. **Вопрос:** Этапы проектирования ИС

**Ответ:** всего этапов проектирования пять - анализ, проектирование, реализация, тестирование, ввод в действие.

Анализ - должен помочь в формировании требований к ИС, корректно и точно отражающих цели и задачи организации заказчика. Конечной целью моделирования бизнес-процессов, протекающих в организации и реализующих ее цели и задачи является построение бизнес-моделей организации. Также необходимо определить концептуальную модель данных, входные данные, процессы обработки данных и предполагаемые выходные данные. Кроме этого, на этом этапе изучаются имеющееся оборудование и программные средства. Результатом анализа должно стать лучшее понимание функционального назначения системы, существующие и потенциальные проблемы, а также сфера ее действия. Заказчики и проектировщики на данном этапе должны работать сообща. Этап анализа заканчивается разработкой технического задания.

Проектирование. Проектировщики в качестве исходной информации получают результаты анализа. Полученная в результате анализа концептуальная модель данных преобразуется в логическую. Параллельно с этим выполняется проектирование бизнес-процессов. При проектировании модулей определяют интерфейсы ПС: меню, вид окон, горячие клавиши и связанные с ними вызовы. Также на этапе проектирования определяется выбор платформы и ОС и характеристики архитектуры ИС. Этап проектирования заканчивается разработкой технического проекта ИС.

Реализация. На этом этапе осуществляется создание всех компонент ПО ИС, установка технических средств, разработка эксплуатационной документации.

Тестирование обычно оказывается распределенным по времени.

После завершения разработки отдельного модуля системы выполняют автономный тест, который преследует следующие цели:

* Обнаружение отказов модуля (жестких сбоев);
* Соответствие модуля спецификации (наличие всех необходимых функций и отсутствие лишних функций).

После того как автономный тест успешно пройден, модуль включается в состав разработанной части системы и группа сгенерированных модулей проходит тесты связей, которые должны отследить их взаимное влияние.

После тестирования на взаимное влияние модулей необходимо выполнить еще ряд тестов на проверку надежности работы:

* Тест имитации отказов, демонстрирующий, насколько хорошо система восстанавливается после сбоев ПО и отказов аппаратного обеспечения;
* Тест наработки на отказ (устойчивость системы при штатной работе для оценки времени безотказной работы системы);
* Системный тест (проверка функциональности системы);
* Приемо-сдаточные испытания (такой тест предусматривает показ ИС заказчику и должен содержать группу тестов, моделирующих реальные бизнес-процессы, чтобы показать соответствие реализации требованиям заказчика).

Ввод в действие. Эксплуатация и сопровождение. После ввода в действия организуется обучение конечных пользователей. Внесение изменений и исправлений в ИС в случае возникновения новых требований, выполняется службой сопровождения системы, работающей в трех направлениях:

* Корректирующее обслуживание – как ответ на возникающие ошибки системы;
* Адаптивное обслуживание – как ответ на изменение корпоративной среды;
* Усовершенствование – расширение возможностей системы.

Таким образом, все этапы проектирования взаимосвязаны, каждый из них заканчивается выпуском конкретного продукта (модели, программного продукта, документации и т. д.).

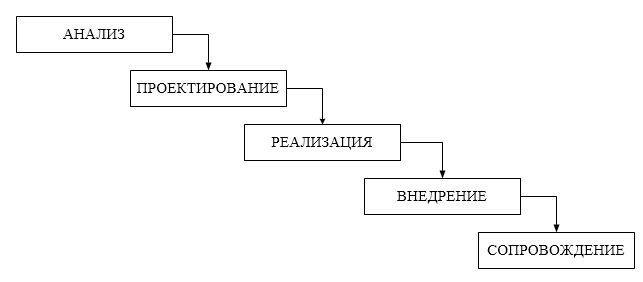
1. **Вопрос:** Понятие жизненного цикла ИС, модели жизненного цикла информационной системы

**Ответ:**

ЖЦ ПО - это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации. Модель ЖЦ отражает различные состояния системы.

В настоящее время известны и используются следующие модели ЖЦ:

Каскадная модель *-* последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

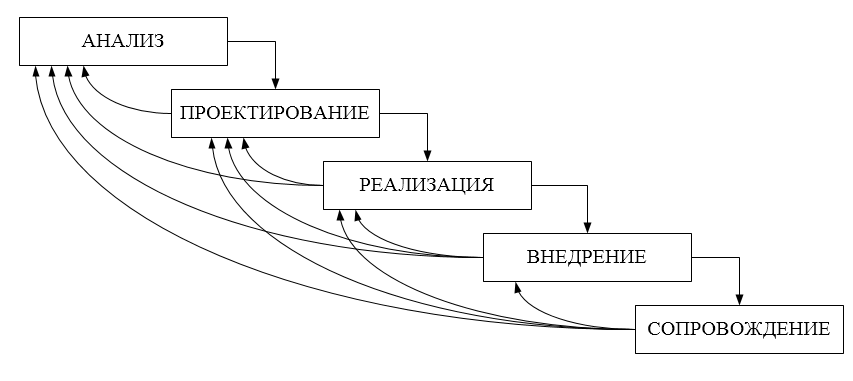


Положительные стороны применения :

* На каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
* Выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

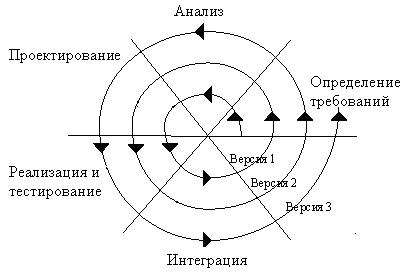
Недостаток: реальный процесс создания системы никогда полностью не укладывается в жесткую схему, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ИС оказывается более соответствующим поэтапной модели с промежуточным контролем.

Поэтапная модель с промежуточным контролем - разработка ведется итерациями с циклами обратной связи между этапами . Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки. Однако и эта модель не позволяет оперативно учитывать возникающие изменения и уточнения требований к системе. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ, требования к ИС «заморожены» в виде технического задания на все время ее создания. Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена. В случае неточного изложения требований или их изменения в ходе создания пользователи получают систему, не удовлетворяющую их потребностям.



Спиральная модель - упор на начальные этапы ЖЦ: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов.

Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы (прототипа), на нем уточняются цели и характеристики проекта, определяется его качество, и планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации.



Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работ на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача - как можно быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований.

Основная проблема спиральной модели - определение момента перехода на следующий этап. Для ее решения необходимо вводить временные ограничения на каждый из этапов ЖЦ, и осуществлять переход в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

1. **Вопрос:** Основные принципы создания ИС

**Ответ:** при создании системы должны быть рационально использованы типовые, унифицированные и стандартизованные элементы, проектные решения, пакеты прикладных программ, комплексы, компоненты.Задачи необходимо разрабатывать таким образом, чтобы они подходили к возможно более широкому кругу объектов

Основополагающие принципы создания ИС:

Принцип системности - заключается в том, что при декомпозиции должны быть установлены такие связи между структурными компонентами системы, которые обеспечивают цельность корпоративной системы и ее взаимодействие с другими системами

Принцип развития (открытости) - заключается в том, что исходя из перспектив развития, объекта автоматизации, АС должна создаваться с учетом возможности пополнения и обновления функций и состава АС без нарушения ее функционирования.

Принцип совместимости - заключается в том, что при создании систем должны быть реализованы информационные интерфейсы, благодаря которым она может взаимодействовать с другими системами в соответствии с установленными правилами.

Принцип стандартизации и унификации - необходимость применения типовых, унифицированных и стандартизованных элементов функционирования ИС.

Принцип эффективности заключается в достижении рационального соотношения между затратами на создание АС и целевыми эффектами, включая конечные результаты, получаемые в результате автоматизации.

Частные принципы (детализирующие общие)

Принцип декомпозиции - разбиение системы на модули для достижения возможности их легкой замены, отладки и тестирования

Принцип первого руководителя - разработка требований к объекту управления, а также создание и внедрение системы управления возглавляются основным руководителем соответствующего объекта.

Принцип новых задач - системы управления должны обеспечивать качественно новые решения управленческих проблем на последующих этапах развития, а не повторять механически приемы управления, реализуемые в предыдущих периодах.

Принцип автоматизации информационных потоков и документооборота - новые данные должны вводиться один раз, а после загружаться из хранилища

Принцип автоматизации проектирования - рутинные задачи проектирования должны делегироваться на вычислительную технику

Организационно-технологические принципы

Принцип абстрагирования - заключается в выделении существенных с некоторых позиций аспектов системы и отвлечение от несущественных с целью представления проблемы в простом общем виде

Принцип формализации - задачи должны быть формализованы для выполнения на тех инструментах, с помощью которых разрабатывается ИС

Принцип концептуальной общности - система должна полностью выполнять задачи для которой она разрабатывается

Принцип непротиворечивости и полноты - компоненты системы и их функционал не должны противоречить друг-другу

Принцип независимости данных - механизм хранения данных должен быть обособлен от основной бизнес-логики

Принцип структурирования данных - данные должны быть представлены в строгом виде с заранее известной структурой

Принцип доступа конечного пользователя - заключается в том, что пользователь должен иметь средства доступа к базе данных, которые он может использовать непосредственно (без программирования)

1. **Вопрос:** Функционально-ориентированные методологии проектирования. Основные принципы функциональной методики IDEF0

**Ответ:** функциональные методологии рассматривают организацию как набор функций, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы.

Основное отличие от объектных методологий заключается в четком отделении функций (методов обработки данных) от самих данных.

Нотация IDEF0 - предназначена для документирования процессов производства и отображения информации об использовании ресурсов на каждом из этапов проектирования систем.

Цель методики - построение функциональной модели исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы.

В основе методологии лежат четыре основных понятия:

* функциональный блок,
* интерфейсная дуга,
* декомпозиция,
* глоссарий.

Декомпозиция является основным понятием стандарта IDEF0.

Принцип декомпозиции применяется для разбиения 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.

Глоссарий.

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.

Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Функциональный блок (Работа, Процесс) представляет собой некоторую конкретную функцию (бизнес-процесс) в рамках рассматриваемой системы.

По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, "производить услуги"). На диаграмме функциональный блок изображается прямоугольником.

Интерфейсная дуга (Поток, Стрелка) отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, представленную данным функциональным блоком.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе.

1. **Вопрос:** Методология IDEF0. Виды стрелок на диаграммах IDEF0

**Ответ:** в IDEF0 различают пять типов стрелок:

* Вход (Input) - объект или информация, которые используются или преобразуются работой для получения результата (выхода).
* Управление (Control) - правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить информация о том, перерабатываются/изменяются ли данные в работе или нет. Если изменяются, то, скорее всего, это вход, если нет - управление.
* Выход (Output) - объект или информация, которые производятся работой.
* Механизм (Mechanism) - ресурсы, которые выполняют работу, например, персонал предприятия, станки, устройства и т. д. По усмотрению аналитика стрелки механизма могут не изображаться в модели.
* Вызов (Call) - специальная стрелка, указывающая на другую модель работы, которая выполняется за пределами моделируемой системы и используется в механизме слияния и разделения моделей.. Стрелка вызова рисуется как исходящая из нижней грани работы.



1. **Вопрос:** Методология IDEF0. Нумерация работ и диаграмм. Каркас диаграммы

**Ответ:** на диаграмме декомпозиции работы нумеруются автоматически слева направо (номер показывается в правом нижнем углу). В левом верхнем углу изображается небольшая диагональная черта, которая показывает, что данная работа не была декомпозирована.

Работы образуют иерархию, то есть каждая работа может иметь одну родительскую и несколько дочерних работ, образуя дерево. Такое дерево называют деревом узлов. Все работы модели нумеруются. Номер работы состоит из префикса А и числа. Контекстная (корневая) работа дерева имеет номер А0. Работы декомпозиции А0 имеют номера А1, А2, A3 и т. д. Работы декомпозиции более нижнего уровня имеют номер родительской работы и очередной порядковый номер, например работы декомпозиции A3 будут иметь номера А31, А32, АЗЗ, А34 и т. д.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть в модели сколь угодно много, поскольку дерево может быть построено на произвольную глубину и не обязательно должно начинаться с корня.

Диаграммы IDEF0 также имеют нумерацию, эта нумерация проводится по номерам узла. Контекстная диаграмма всегда имеет номер А-0, диаграмма декомпозиции контекстной диаграммы - номер А0, остальные диаграммы декомпозиции имеют номера по соответствующему узлу (например, A1, A2, А21, А213 и т. д.).

Каркас (граничные рамки) содержит:

* заголовок (верхняя часть рамки);
* подвал (нижняя часть).

Заголовок каркаса используется для отслеживания диаграммы в процессе моделирования. Нижняя часть используется для идентификации и позиционирования в иерархии диаграммы.

Поля подвала каркаса (слева направо)

|  |  |
| --- | --- |
| Поле | Смысл |
| Node | Номер узла диаграммы (номер родительской работы) |
| Title | Имя диаграммы. По умолчанию — имя родительской работы |
| Number | C-Number, уникальный номер версии диаграммы |
| Page | Номер страницы, может использоваться как номер страницы при формировании папки |

Поля заголовка каркаса (слева направо)

|  |  |
| --- | --- |
| Поле | Смысл |
| Used At | Используется для указания на родительскую работу в случае, если на текущую диаграмму ссылались посредством стрелки вызова |
| Autor, Date, Rev, Project | Имя создателя диаграммы, дата создания и имя проекта, в рамках которого была создана диаграмма. REV-дата последнего редактирования диаграммы |
| Notes 123456789 10 | Используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания |
| Working | Новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы |
| Draft | Диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению |
| Recommended | Диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается |
| Publication | Диаграмма готова к окончательной печати и публикации |
| Reader | Имя читателя (эксперта) |
| Date | Дата прочтения (экспертизы) |
| Context | Схема расположения работ в диаграмме верхнего уровня. Работа, являющаяся родительской, показана темным прямоугольником, остальные – светлым. На контекстной диаграмме (А-0) показана надпись ТОР. В левом нижнем углу показывается номер по узлу родительской диаграммы |

1. **Вопрос:** Методология IDEF0. Виды диаграммы IDEF0

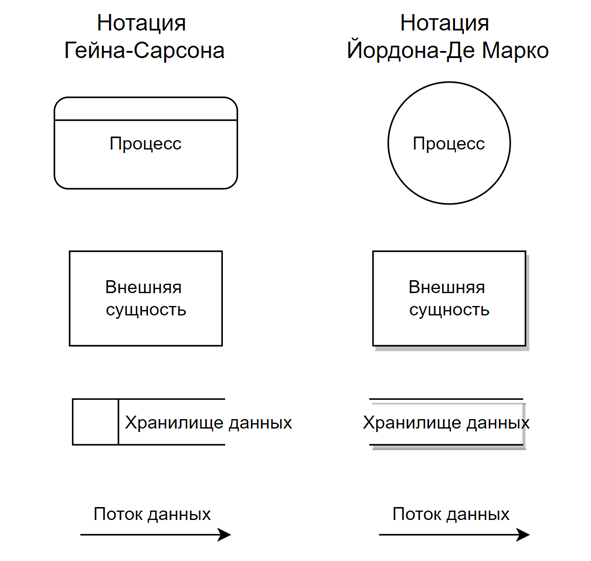
**Ответ:** функциональная модель IDEF0 может содержать четыре типа диаграмм:

* контекстную диаграмму;
* диаграммы декомпозиции;
* диаграммы дерева узлов;
* диаграммы только для экспозиции (FEO).

1. **Вопрос:** Диаграммы потоков данных. Назначение. Нотации DFD. Рекомендации по построению

**Ответ:** диаграммы потоков данных (DFD) - диаграммы, которые разбиваются на функциональные элементы (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Для построения DFD можно использовать различные **нотации**, среди которых наиболее известны нотация Гейна-Сарсона (Gane/Sarson) и нотация Йордона-Де Марко, разработанная авторами Эд Йордон (Yourdon) и Том де Марко (DeMarko).



Правила и советы по построению DFD:

* Каждый процесс должен сопровождаться как минимум одним входным и одним выходным потоком;
* В каждое хранилище должен впадать как минимум один поток данных и как минимум один вытекать;
* Данные, хранимые в системе, должны проходить через процесс;
* Каждый процесс диаграммы DFD должен вести либо к другому процессу, либо к хранилищу данных.

1. **Вопрос:** Основные элементы диаграмм DFD

**Ответ:** основные элементы диаграмм DFD:

* Поток данных - является механизмом, который показывает передачу информации от одного процесса другому. На схеме они обычно отражаются направленной стрелкой, которая показывает направление движения информации или. Имя потока должно быть существительным.
* Процесс - преобразует значения данных. Это имя должно содержать глагол в неопределенной форме. Кроме того, каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.
* Хранилище (накопитель) данных - это пассивный объект в составе DFD, в котором данные сохраняются для последующего доступа. Этот элемент позволяет на определенных участках определять данные, которые будут сохраняться в памяти между обработкой их процессами. Имя хранилища должно идентифицировать его содержимое и быть существительным.
* Внешняя сущность (источник/приемник информации) - представляет сущность вне контекста системы, являющуюся источником или приемником системных данных. Ее имя должно быть существительным.

1. **Вопрос:** Диаграммы DFD. Элементы для декомпозиции данных

**Ответ:** декомпозиция применяется к каждому процессу на диаграмме уровня 1, для которого внутри процесса скрыто достаточно подробностей. Каждый процесс на диаграммах уровня 2 также необходимо проверить на возможную декомпозицию и так далее.

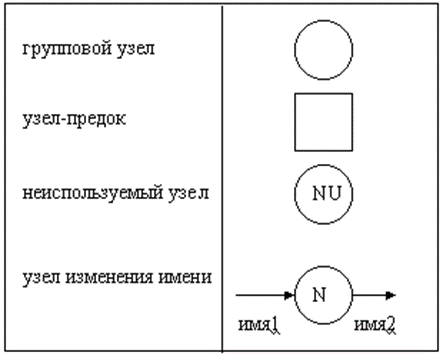
Каждый процесс на диаграмме уровня 1 исследуется более подробно, чтобы лучше понять действия и потоки данных.

Обычно процессы декомпозируются там, где:

* Вокруг процесса существует более шести потоков данных.
* Название процесса сложное или очень общее, что указывает на то, что он включает в себя ряд действий.

Для обеспечения декомпозиции данных и некоторых других сервисных возможностей к DFD добавляются следующие типы объектов:

* Групповой узел. Предназначен для расщепления и объединения потоков. В некоторых случаях может отсутствовать (т.е. Фактически вырождаться в точку слияния/расщепления потоков на диаграмме).
* Узел-предок. Позволяет связывать входящие и выходящие потоки между детализируемым процессом и детализирующей DFD.
* Неиспользуемый узел. Применяется в ситуации, когда декомпозиция данных производится в групповом узле, при этом требуются не все элементы входящего в узел потока.
* Узел изменения имени. Позволяет неоднозначно именовать потоки, при этом их содержимое эквивалентно. Например, если при проектировании разных частей системы один и тот же фрагмент данных получил различные имена, то эквивалентность соответствующих потоков данных обеспечивается узлом изменения имени. При этом один из потоков данных является входным для данного узла, а другой - выходным.
* Текст в свободном формате в любом месте диаграммы.



1. **Вопрос:** Диаграммы DFD. Управляющие элементы диаграмм

**Ответ:** управляющий процесс - логически управляющий процесс есть некий командный пункт, реагирующий на изменения внешних условий, передаваемые ему с помощью управляющих потоков, и продуцирующий в соответствии со своей внутренней логикой выполняемые процессами команды. Фактически управляющий процесс представляет собой преобразователь входных управляющих потоков в выходные управляющие потоки.

Управляющий поток- представляет собой «трубопровод», через который проходит управляющая информация.

Управляющее хранилище - представляет собой «срез» управляющего потока во времени. Содержащаяся в нем управляющая информация может использоваться в любое время после ее внесения в хранилище, при этом соответствующие данные могут быть использованы в произвольном порядке. Управляющее хранилище отличается от традиционного тем, что может содержать только управляющие потоки; все другие их характеристики идентичны.

1. **Вопрос:** Методология IDEF3. Назначение диаграмм. Основные элементы

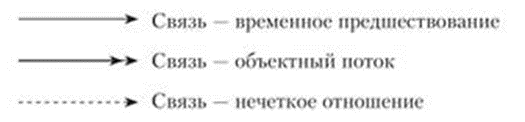
**Ответ:** нотация IDEF3 широко используется для создания моделей бизнес-процессов организации на нижнем уровне декомпозиции, т.е. при описании процессов уровня рабочих мест. В этом случае диаграмма, описывающая процесс может служить основой для создания документов, регламентирующих работу исполнителей.

На основе IDEF3-диаграмм создаются сценарии действий сотрудников предприятия/организации. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

Единица работы (Unit of Work) - основной элемент диаграммы IDEF3 близок по смыслу к работе в нотации IDEF0. В отличие от IDEF0 в нотации IDEF3 стороны прямоугольника, изображающего работу (процесс), не используют для привязки входов различного типа. Более того, в него может входить и выходить только одна стрелка. Имя работы должно быть сформулировано в глагольном наклонении.

Связь (Links) - показывают взаимоотношения работ. Связи изображаются однонаправленными стрелками, обычно диаграммы рисуют таким образом, чтобы связи были направлены слева направо, или сверху вниз. Имя стрелки должно ясно идентифицировать отображаемый объект. В IDEF3 различают три типа связей:

* *связь предшествования* (Precedence) – временное предшествование. Показывает, что исходное действие (работа-источник) должно полностью завершиться, прежде чем начнется выполнение следующего действия (работа-цель). Обозначается сплошной линией со стрелкой.
* *поток объектов* (Object Flow). Показывает участие некоторого объекта в двух или более работах, как, например, если объект производится в ходе выполнения одной работы и потребляется другой работой. Обозначается стрелкой с двумя наконечниками.
* *связь отношения* (Relational). Показывает связь между двумя работами или между работой и объектом ссылки. Используется для отображения взаимоотношений между параллельно выполняющимися действиями, является альтернативой старшей стрелке в смысле задания последовательности выполнения работ – работа-источник не обязательно должна закончиться прежде, чем работа-цель начнется. Более того, работа-цель может закончиться прежде, чем закончится работа-источник. Обозначается пунктирной линией со стрелкой.



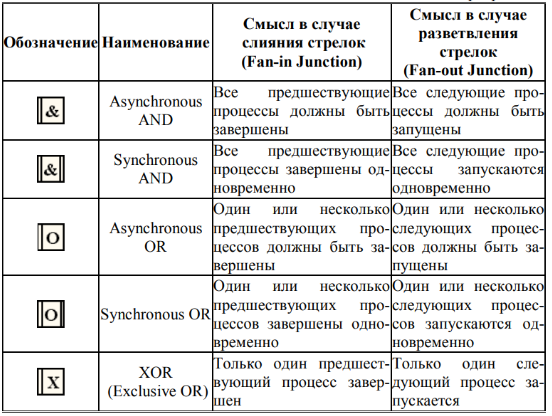
Объект ссылки (Referent) - выражает идею, концепцию данных, которые нельзя связать со стрелкой, перекрестком, работой и используется при построении диаграммы для привлечения внимания пользователя к каким- либо важным аспектам модели. В качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или перекрестками сплошными линиями без стрелок.

Перекрестки (Junctions) - используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния (Fan-in Junction) и разветвления стрелок (Fan-out Junction). Перекресток не может использоваться одновременно для слияния и для разветвления..

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J. В отличие от IDEF0 и DFD, в IDEF3 стрелки могут сливаться и разветвляться только через перекрестки

1. **Вопрос:** Виды перекрестков на диаграммах IDEF3

**Ответ:** типы перекрестков на диаграммах IDEF3 представлены в таблице:



1. **Вопрос:** SwimLane диаграммы и построение сценариев

**Ответ:** SwimLane позволяет явно описать роли и ответственности исполнителей в конкретной технологической операции. Swim Lane диаграммы можно добавлять к любой модели для более наглядного изображения течения процесса. и показывают горизонтальные полосы, которые представляют участие в процессе ролей.

Эти диаграммы использует те же группы ролей, что определены для организационной диаграммы и методологию IDEF3, область диаграммы разделена на горизонтальные полосы, с каждой полосой связана роль. Каждая полоса должна содержать объекты диаграммы IDEF3, относящиеся к соответствующей роли.

Диаграмма дорожек — это тип блок-схемы, которая показывает, кто что делает в процессе. Используя метафору дорожек в бассейне, диаграмма дорожек обеспечивает ясность и подотчетность, помещая этапы процесса в горизонтальные или вертикальные «дорожки» конкретного сотрудника, рабочей группы или отдела. Он показывает соединения, связь и передачу обслуживания между этими дорожками и может служить для выявления потерь, избыточности и неэффективности процесса

В современных организациях с несколькими рабочими группами или отделами такое построение диаграмм может помочь по-разному:

Это может помочь гарантировать, что правая рука знает, что делает левая в организации. Диаграммы плавательных дорожек и плавательные дорожки, используемые в других типах диаграмм, подчеркивают, какие этапы процесса или подпроцессы назначены конкретному участнику в организации.

Обозначив это на диаграмме, вы можете выделить избыточность между различными полосами и определить узкие места, потери и другие факторы неэффективности. Иногда это выявляет повторяющиеся или ненужные шаги в процессе, например, разные отделы выполняют одну и ту же задачу. Он также может выделять задержки процессов или ограничения пропускной способности в пределах определенной дорожки, чтобы их можно было устранить. Это может повысить производительность и качество и сократить ненужную работу и затраты.

Вы можете использовать вторую диаграмму Swimlane для моделирования лучшего способа структурирования процесса или для учета меняющихся обстоятельств, таких как кадровые изменения или изменения технологий.

Диаграммы Swimlane могут передавать информацию более четко, чем описательное описание.

Диаграммы плавательных дорожек могут быть формализованы как способ интеграции процессов между командами или отделами, что приводит к постоянному повышению чистоты процессов.

На основании диаграммы может быть построен сценарий действия, в котором описывается действия актёра и отклик системы на его действие.

1. **Вопрос:** Стоимостной анализ. Принципы связи ABC анализа и IDEF0

**Ответ:** стоимостный анализ представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость процесса. Стоимостный анализ основан на модели работ, потому что количественная оценка невозможна без детального понимания функциональности предприятия. Обычно ABC применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Reengineering, BPR). С помощью стоимостного анализа можно решить такие задачи, как определение действительной стоимости производства продукта, определение действительной стоимости поддержки клиента, идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь), обеспечение менеджеров финансовой мерой предлагаемых изменений т.д.

ABC может проводиться только тогда, когда:

* модель работы последовательная (следует синтаксическим правиламIDEF0)
* корректная (отражает бизнес)
* полная (охватывает всю рассматриваемую область)
* стабильная (проходит цикл экспертизы без изменений)

Связь IDEF0- и ABC-моделей базируется на трех принципах:

* Функция характеризуется числом, которое представляет собой стоимость или время выполнения этой функции.
* Стоимость или время функции, которая не имеет декомпозиции, определяется разработчиком системы.
* Стоимость или время функции, которая имеет декомпозицию, определяется, как сумма стоимостей (времен) всех подфункций на данном уровне декомпозиции.

Оценка информации, полученная по ABC модели позволяет:

* грамотно и наглядно представить технологию работы каждого структурного подразделения компании;
* определить документооборот и информационные потоки;
* выделить основные, вспомогательные и управляющие функции подразделений торговой компании;
* грамотно распределить функции между подразделениями и сотрудниками;
* снизить временные и стоимостные затраты связанные с выполнением бизнес-процессов;
* повысить оперативное управление.

1. Вопрос: Количественный и качественный анализ диаграмм модели IDEF

Ответ: Для проведения количественного анализа диаграмм перечислим показатели модели:

* количество блоков на диаграмме - N;
* уровень декомпозиции диаграммы - L;
* сбалансированность диаграммы - В;
* число стрелок, соединяющихся с блоком, - А.

Данный набор факторов относится к каждой диаграмме модели. Далее будут перечислены рекомендации по желательным значениям факторов диаграммы. Необходимо стремиться к тому, чтобы количество блоков на диаграммах нижних уровней было бы ниже количества блоков на родительских диаграммах, т. е. с увеличением уровня декомпозиции убывал бы коэффициент N/L. Таким образом, убывание этого коэффициента говорит о том, что по мере декомпозиции модели функции должны упрощаться, следовательно, количество блоков должно убывать. Диаграммы должны быть сбалансированы. Это означает, что в рамках одной диаграммы не должно происходить ситуации, у работы 1 входящих стрелок и стрелок управления значительно больше, чем выходящих. Следует отметить, что данная рекомендация может не выполняться в моделях, описывающих производственные процессы.

Помимо анализа графических элементов диаграммы необходимо рассматривать наименования блоков. Для оценки имен составляется словарь элементарных (тривиальных) функций моделируемой системы. Фактически в данный словарь должны попасть функции нижнего, уровня декомпозиции диаграмм. Например, для модели БД элементарными могут являться функции «найти запись», «добавить запись в БД», в то время как функция «регистрация пользователя» требует дальнейшего описания. После формирования словаря и составления пакета диаграмм системы необходимо рассмотреть нижний уровень модели. Если на нем обнаружатся совпадения названий блоков диаграмм и слов из словаря, то это говорит, что достаточный уровень декомпозиции достигнут. Коэффициент, количественно отражающий данный критерий, можно записать как L\*C -произведение уровня модели на число совпадений имен блоков со словами из словаря. Чем ниже уровень модели (больше L), тем ценнее совпадения.

1. ER-диаграммы. Определение сущности, атрибута, связи

Ответ: Наиболее распространенным средством моделирования данных являются диаграммы "сущность-связь" (ERD). С помощью ERD осуществляется детализация накопителей данных DFD – диаграммы, а также документируются информационные аспекты бизнес-системы, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их связей с другими объектами (отношений).

**ERD Сущность** (Entity) — множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, идей, предметов и др.), обладающих общими атрибутами или характеристиками. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

**Связь** (Relationship)— поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь — это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот.

**Атрибут** (Attribute)— любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, предметов и т.д.).

1. Методология IDEF1X. Виды и мощности связей. Понятие зависимых и независимых сущностей

Ответ: **Связь** является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы. По умолчанию имя связи на диаграмме не показывается. На логическом уровне можно установить идентифицирующую связь "один-ко-многим", связь "многие-ко-многим" и неидентифицирующую связь "один-ко-многим". **Мощность** связи представляет собой отношение количества экземпляров одной сущности к соответствующему количеству экземпляров другой сущности. Связи в модели «сущность-связь» могут иметь мощность:«один-к-одному»(1:1),«один-ко-многим»(1:M) и«многие-ко-многим»(M:N). Связь 1:1 («один-к-одному») – это связь между двумя типами сущностей A и B, при которой каждому экземпляру сущности A соответствует один и только один экземпляр сущности B. Связь 1:M («один-ко-многим») – это связь между двумя типами сущностей A и B, при которой одному экземпляру сущности A может соответствовать ноль, один или несколько экземпляров сущности B, но каждому экземпляру сущности B соответствует только один экземпляр сущности A. Связь M:N («многие-ко-многим») – это связь между двумя типами сущностей A и B, при которой каждому экземпляру сущности A может соответствовать ноль, один или несколько экземпляров сущности B и наоборот.

Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями.

1. Методология IDEF1X. Виды зависимых сущностей

Различают несколько типов зависимых сущностей.

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

Ассоциативная — сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.

Именующая — частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

Категориальная— дочерняя сущность в иерархии наследования. Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи, либо когда это диктуется бизнес-правилами.

1. Нормальные формы ER-диаграмм. Получение реляционной схемы из ER-диаграмм

В первой нормальной форме ER-схемы устраняются повторяющиеся атрибуты или группы атрибутов, т.е. производится выявление неявных сущностей, "замаскированных" под атрибуты.

Во второй нормальной форме устраняются атрибуты, зависящие только от части уникального идентификатора. Эта часть уникального идентификатора определяет отдельную сущность.

В третьей нормальной форме устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. Эти атрибуты являются основой отдельной сущности.

**Получение реляционной схемы из ER-диаграмм.**

Шаг 1. Каждая простая сущность превращается в таблицу. Простая сущность - сущность, не являющаяся подтипом и не имеющая подтипов. Имя сущности становится именем таблицы.

Шаг 2. Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, - не могут.

Шаг 3. Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы. Если имеется несколько возможных уникальных идентификатора, выбирается наиболее используемый. Если в состав уникального идентификатора входят связи, к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (этот процесс может продолжаться рекурсивно). Для именования этих столбцов используются имена концов связей и/или имена сущностей.

Шаг 4. Связи многие-к-одному (и один-к-одному) становятся внешними ключами. Т.е. делается копия уникального идентификатора с конца связи "один", и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи - столбцам, не допускающим неопределенные значения.

Шаг 5. Индексы создаются для первичного ключа (уникальный индекс), внешних ключей и тех атрибутов, на которых предполагается в основном базировать запросы.

Шаг 6. Если в концептуальной схеме присутствовали подтипы, то возможны два способа: все подтипы в одной таблице (а) для каждого подтипа - отдельная таблица (б)

1. Реляционная модель данных. Структурная часть. Управляющая связь. Виды ключей

**Структурная часть**– РМ основ на мат понятии «отношение», физ представ кот является двумер. таблица отношения, исп-мая для хранения информации об объектах БД.

**Суперключ** (superkey) – это атрибут или множество атрибутов, которое единственным образом идентифицирует кортеж данного отношения. Суперключ однозначно обозначает каждый кортеж в отношении.

Если ключ состоит из нескольких атрибутов, то он называется **составным** ключом.

**Первичный ключ** – это потенциальный ключ, который выбран для уникальной идентификации кортежей внутри отношения.

**Альтернативный ключ** – это потенциальный ключ, который не выбран в качестве первичного ключа

**Внешний ключ** – это атрибут или множество атрибутов внутри отношения, которое соответствует потенциальному ключу некоторого (может быть, того же самого) отношения.

1. Реляционная модель данных. Набор ограничений целостности. Операции, нарушающие целостность

**Набор ограничений целостности**

**Ограничение целостности** – это совокупность утверждений о допустимости значений в базе д-х и связях между ними. При выполнении операций над базой данных проверяется выполнение ограничения целостности и действия, приводящие к нарушению ограничений целостности, отвергаются.

Ограничения, которые используются только при проверке допустимости корректировки, называются **ограничениями перехода**

**Ограничение целостности атрибута** представляют собой ограничения, накладываемые на допустимые значения атрибута.

**Ограничения целостности кортежа** представляют собой ограничения, накладываемые на допустимые значения отдельного кортежа отношения, и не являющиеся ограничением целостности атрибута.

**Явные ограничения целостностью** – это ограничения, диктуемые предметной областью, они так же называются внешними ограничениями или семантическими.

**Неявные ограничения целостности** – определяются моделью данных, называются также синтаксическими ограничениями целостности.

Операции нарушающие целостность (РО – родительский объект, ДО – дочерний объект)

* Вставка картежа в родительское отношение – не нарушая ссылочной целостности.
* Обновление кортежа в РО – может привести к нарушении СЦ.
* Удаление кортежа в РО – может привести к нарушении СЦ.
* Вставка в ДО – может привести к нарушении СЦ.
* Обновлении в ДО – может привести к нарушении СЦ.
* Удаление из ДО не нарушая ссылочной целостности

1. Объектно-ориентированный подход к проектированию. Методология RUP

Ответ:

Объектно-ориентированные методологии рассматривают моделируемую

предметную область как набор взаимодействующих объектов –

производственных единиц.

Цель их применения - формирование требований на основе понятий классов

и объектов, составляющих словарь предметной области, и распределение

между ними ответственностей за выполняемые действия.

Объект при этом определяется как осязаемая реальность – предмет или

явление, имеющее четко определяемое поведение.

Методология объектно-ориентированного анализа и проектирования

(ООАП) соединяет в себе процесс объектной декомпозиции и многообразие

приемов представления моделей, отражающих логическую структуру системы

(классы и объекты) и физическую структуру системы (модули и процессы), а также ее статические и динамические аспекты.

Суть работы в рамках RUP - это создание и сопровождение моделей на базе языка моделирования Unified Modeling Language (UML), являющегося международным стандартом. В основе развития инструментов и процессов Rational лежат следующие ключевые концепции:

* адаптация процесса под проект;
* управление запросами заинтересованных лиц;
* организация взаимодействия удаленных команд разработчиков;
* итерационный подход к разработке;
* повышение уровня абстрагирования;
* непрерывное повышение качества.

1. Язык моделирования UML. Назначение. Характеристики. Перечень диаграмм

UML представляет собой язык, разработанный

для специализации визуализации, проектирования и документирования компонентов ПО, бизнес-процессов и других систем в рамках ООАП.

Универсальный язык моделирования (Unified Modeling Language – UML) является достаточно строгим и мощным средством моделирования, может быть эффективно использован для построения концептуальных, логических и физических моделей сложных систем различного целевого назначения.

Язык UML предназначен для решения следующих задач:

1. Предоставить в распоряжение пользователей легко воспринимаемый и

выразительный язык визуального моделирования, специально предназначенный для разработки и документирования моделей сложных систем самого различного целевого назначения.

2. Снабдить исходные понятия языка UML возможностью расширения и специализации для более точного представления моделей систем в конкретной предметной области.

3. Описание языка UML должно поддерживать такую спецификацию моделей, которая не зависит от конкретных языков программирования и инструментальных средств проектирования программных систем.

4. Описание языка UML должно включать в себя семантический базис для понимания общих особенностей ООАП.

5. Поощрять развитие рынка объектных инструментальных средств.

6. Способствовать распространению объектных технологий и соответствующих понятий ООАП.

7. Интегрировать в себя новейшие и наилучшие достижения практики ООАП.

Каждая из этих диаграмм детализирует и конкретизирует различные представления о модели сложной системы в терминах языка UML.

При этом **диаграмма вариантов использования** представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм.

**Диаграмма классов**, по своей сути - логическая модель, отражающая статические аспекты структурного построения сложной системы.

**Диаграммы кооперации и последовательностей** представляют собой разновидности логической модели, которые отражают динамические аспекты функционирования сложной системы.

**Диаграммы состояний и деятельности** предназначены для моделирования поведения системы.

И, наконец, **диаграммы компонентов и развертывания** служат для представления физических компонентов сложной системы и поэтому относятся к ее физической

1. Построение диаграммы вариантов использования средствами UML

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

<https://classroom.google.com/u/1/c/NjIyMzcwMjYyNjMw/m/NjYzMTY3NTMwODY2/details>

1. Построение диаграммы классов средствами UML. Основные элементы

**Класс**

Класс - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.

**Атрибуты**

Атрибут – это элемент информации, связанный с классом. Атрибуты хранят инкапсулированные данные класса.

Так как атрибуты содержатся внутри класса, они скрыты от других классов. В связи с этим может понадобиться указать, какие классы имеют право читать и изменять атрибуты. Это свойство называется видимостью атрибута (attribute visibility).

У атрибута можно определить четыре возможных значения этого параметра:

* Public (общий, открытый). Это значение видимости предполагает, что атрибут будет виден всеми остальными классами. Любой класс может просмотреть или изменить значение атрибута. В соответствии с нотацией UML общему атрибуту предшествует знак « + ».
* Private (закрытый, секретный). Соответствующий атрибут не виден никаким другим классом. Закрытый атрибут обозначается знаком « – » в соответствии с нотацией UML.
* Protected (защищенный). Такой атрибут доступен только самому классу и его потомкам. Нотация UML для защищенного атрибута – это знак « # ».
* Package or Implementation (пакетный). Предполагает, что данный атрибут является общим, но только в пределах его пакета. Этот тип видимости не обозначается никаким специальным значком.