Московский технологический университет (МИРЭА, МГУПИ, МИТХТ)

ЦЕНТР ДИСТАНЦИОННОГО ОБУЧЕНИЯ

Проектирование информационных систем

ФИО Асадова Юлия Сергеевна

E-mail: kunaka@bk.ru

Условия обучения

- По итогам изучения дисциплины проводится зачет
- В течение семестра необходимо выполнить все задания по календарному плану, который опубликован на Учебном портале
- Для допуска на сессию набрать 40 баллов

Темы дисциплины

1. **Развитие технологий разработки информационных систем**
2. **Методические аспекты проектирования информационных систем**
3. **Визуальное моделирование**
4. **Технологии создания информационных систем**
5. **Предметно-ориентированное проектирование**

Тема 1

РАЗВИТИЕ ТЕХНОЛОГИЙ РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ

Развитие индустрии программного обеспечения

- середина 50-х г. XX в.
- начало 60-х г. XX в.
- начало 70-х г. XX в.
- начало XXI в.

1.1 Этапы развития технологий разработки программного обеспечения

I этап

Осмысление опыта разработки больших ПС

II этап

Разработка новых техноло-гических подходов

III этап

Принятие стандартов на состав процессов жизненного цикла ПО

1.2 Свойства программного обеспечения

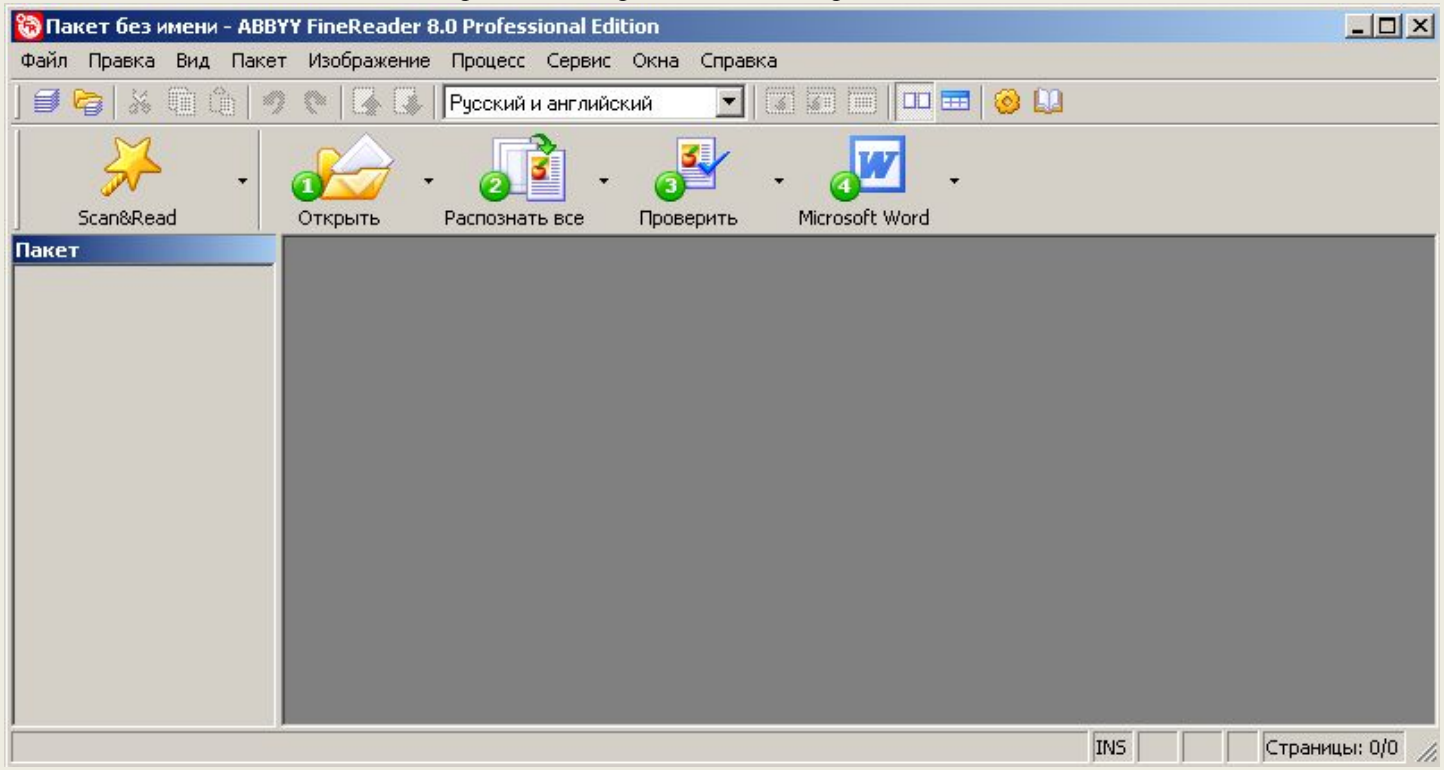
1. сложность
2. согласованность
3. изменяемость
4. незримость

Причины, по которым сложность является неотъемлемым свойством ПО

- сложность реальной предметной области, из которой исходит заказ на разработку
- трудность управления процессом разработки
- неудовлетворительные способы описания поведения больших дискретных систем
  - программные продукты являются дискретными, т.е. событие, внешнее по отношению к ним, может перевести систему в новое состояние
  - элементы системы взаимодействуют между собой нелинейным образом, и сложность целого также возрастает нелинейно

Согласованность

- Во многих случаях новые программы должны согласовываться с уже существующим ПО



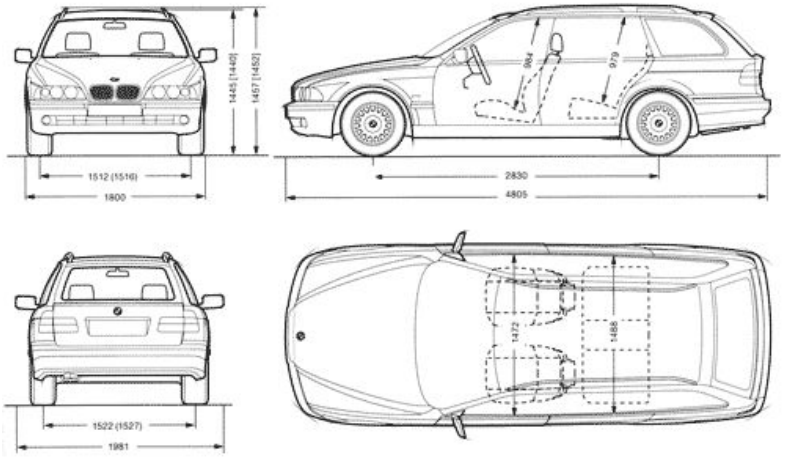
Изменяемость

Все удачные программные продукты подвергаются изменениям

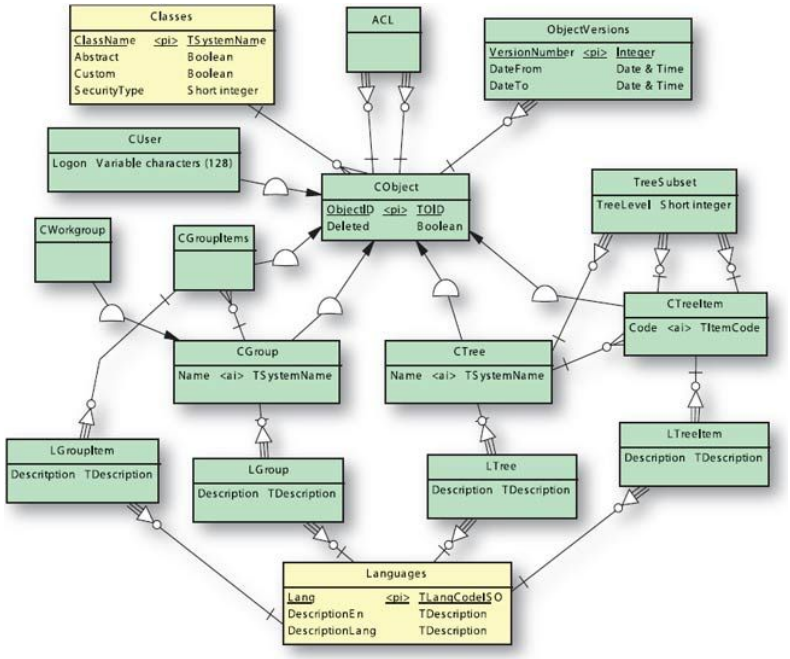
1. как только обнаруживается польза системы, начинаются попытки применения её на грани или за пределами первоначальной области
2. удачное ПО живёт дольше обычного срока существования компьютера, для которого оно первоначально было создано

Незримость

Модель материального объекта



Концептуальная модель БД



1.3 Понятие «программная инженерия» (Software engineering)

- **Программная инженерия** – это область компьютерной науки и технологии, которая занимается построением программных систем, настолько больших и сложных, что для этого требуется участие слаженных команд разработчиков различных специальностей и квалификаций

Фундаментальная идея программной инженерии

- Проектирование программных средств является формальным процессом, который можно изучать и совершенствовать. Освоение и правильное применение методов и средств создания ПО позволяет повысить его качество, обеспечить управляемость процесса проектирования ПО и увеличить срок его жизни

Жизненный цикл программного продукта (10-15 лет)

создание (разработка)

3-4 года

сопровождение и развитие

- начальная стадия сопровождения
- стадия "зрелого" сопровождения
- стадия эволюции/замены

Сопровождение и развитие

- начальная стадия - связана с устранением ошибок и доработками, возникшими из-за не учтённых или вновь возникших требований пользователей
- "зрелое" сопровождение, характеризуется относительно полным удовлетворением основных потребностей пользователей, но:
  - в процессе развития и изменения ПО его первичная структура искажается и усложняется
  - новые требования к ПО выходят за рамки ограничений, заложенных при его создании
  - текучесть кадров приводит к снижению количества специалистов, способных сопровождать ПО
- стадию эволюции/замены, характеризуются достижением ПО порога своей сопровождаемости в существующем виде и не возможностью удовлетворить новые требования без внесения принципиальных изменений

Виды сопровождения

- **корректирующее** – внесение изменений в эксплуатируемый программный продукт в целях исправления обнаруженных ошибок
- **совершенствующее** – повышения производительности и улучшения эксплуатационных характеристик системы
- **адаптирующее** – адаптации к изменившейся или изменяющейся среде

Тема 2

МЕТОДИЧЕСКИЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1 Общие принципы проектирования систем

- **Система** – это совокупность взаимодействующих компонентов, работающих совместно для достижения определенных целей
- Свойства и поведение системных компонентов влияют друг на друга сложным образом
- Корректное функционирование каждого системного компонента зависит от функционирования многих других компонентов

Для решения проблемы сложности системы широко используется метод иерархической декомпозиции

- сложная система разбивается на более простые части
- каждая подсистема, в свою очередь, строится из частей меньшего размера
- и т.д., до тех пор, пока самые небольшие части можно будет строить из имеющегося материала

Правильная декомпозиция системы

- количество связей между отдельными модулями должно быть минимальным
- связность отдельных частей внутри каждого модуля должна быть максимальной

Взаимодействия между модулями

- каждый модуль должен инкапсулировать своё содержимое, т.е. скрывать его от других модулей
- каждый модуль должен иметь чётко определённый интерфейс с другими модулями

Принципы модульности при проектирование ИС

- **Программные модули** решают относительно небольшие функциональные задачи, и каждый реализуется 10-100 операторами языка программирования
- **Функциональные группы программ** (**компоненты**) формируются на базе нескольких или десятков модулей и решают достаточно сложные автономные задачи
- **Комплексы программ** – программный продукт созданный для решения сложных задач управления и обработки информации

Основные подходы к декомпозиции систем

- функционально-модульный (структурный)
- объектно-ориентированный

описывает ся в терминах объектов и связей между ними, а поведение системы описывает

- ся в терминах обмена сообщения ми между объектами
- разделение функций системы на модули производится по функциональной принадлежности

2.2 Основные принципы объектно-ориентированного подхода

Основные элементы объектной модели

- абстрагирование
- инкапсуляция
- модульность
- иерархия

Дополнительные элементы объектной модели

- типизация
- параллелизм
- устойчивость

Основные понятия объектно-ориентированного подхода

Объект

- осязаемая сущность, предмет или явление, имеющие чётко определяемое поведение

Класс

- это множество объектов, связанных общностью структуры и поведения

Свойства объекта

- **Состояние** объекта характеризуется перечнем всех возможных свойств данного объекта и текущими значениями каждого из этих свойств
- **Поведение** определяет воздействие объекта на другие объекты и его реакцию на запросы от других объектов
- **Индивидуальность** – это свойства объекта, отличающие его от всех других объектов

Свойства класса

- **Атрибут** – это поименованное свойство класса, определяющее диапазон допустимых значений, которые могут принимать экземпляры данного свойства
  - public (общий, открытый) – атрибут будет виден всеми остальными классами
  - private (закрытый, секретный) – атрибут не виден никаким другим классом
  - protected (защищенный) – атрибут доступен только самому классу и его потомкам в иерархии наследования
- **Операция** – это реализация услуг, которую можно запросить у любого объекта данного класса
  - Методы являются специфической реализацией операций
  - Интерфейсом называется набор операций, характеризующих поведение элемента

Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм

- **Полиморфизм** – это способность скрывать множество различных реализаций под единственным общим интерфейсом
- **Наследование** означает построение новых классов на основе существующих с возможностью добавления или переопределения свойств и поведения

2.3 Методы проектирования ИС

1. Классификация методов проектирования
2. Каноническое проектирование ИС
3. Типовое проектирование ИС
4. CASE-технологии

Классификация методов проектирования

По степени автоматизации

По степени использования типовых проектных решений

По степени адаптивности проектных решений

Каноническое проектирование ИС

- В основе канонического проектирования лежит каскадная модель ЖЦ ИС
- Стадии:
  1. исследование и обоснование создания системы
  2. разработка технического задания
  3. создание эскизного проекта
  4. техническое проектирование
  5. рабочее проектирование
  6. ввод в действие
  7. функционирование, сопровождение, модернизация

Исследование и обоснование создания системы

- Основная задача данного этапа – это оценка реального объёма проекта, его целей и задач на основе выявленных функций и информационных элементов автоматизируемого объекта
- Результатом является документ (**технико-экономическое обоснование проекта**), где чётко сформулировано, что получит заказчик, если согласится финансировать проект, когда он получит готовый продукт и сколько это будет стоить

При разработке технического задания необходимо решить следующие задачи

- установить общую цель создания ИС, определить состав подсистем и функциональных задач
- разработать и обосновать требования, предъявляемые к подсистемам
- разработать и обосновать требования, предъявляемые к информационной базе, математическому и программному обеспечению, комплексу технических средств
- установить общие требования к проектируемой системе
- определить перечень задач создания системы и исполнителей
- определить этапы создания системы и сроки их выполнения
- провести предварительный расчёт затрат на создание системы и определить уровень экономической эффективности её внедрения

На этапе эскизного проектирования определяются

- функции ИС
- функции подсистем и их цели
- состав комплексов задач и отдельных задач
- концепция информационной базы и её укрупненная структура
- функции СУБД
- состав вычислительной системы и других технических средств
- функции и параметры основных программных средств

Типовое проектирование ИС

Типовое проектное решение (ТПР) – это пригодное к многократному использованию проектное решение

- элементные ТПР – типовые решения по задаче или по отдельному виду обеспечения задачи
- подсистемные ТПР – в качестве элементов типизации выступают отдельные подсистемы, разработанные с учётом функциональной полноты и минимизации внешних информационных связей
- объектные ТПР – типовые отраслевые проекты, которые включают полный набор функциональных и обеспечивающих подсистем ИС

Для реализации типового проектирования используются два подхода

параметрически-ориентированное

- определение критериев пригодности ППП для решения поставленных задач
- анализ и оценка доступных ППП по сформулированным критериям
- выбор и закупка наиболее подходящего пакета
- настройка параметров (доработка) закупленного ППП

модельно-ориентированное

заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации

Реализация типового проекта предусматривает выполнение следующих операций

- установку глобальных параметров системы
- задание структуры объекта автоматизации
- определение структуры основных данных
- задание перечня реализуемых функций и процессов
- описание интерфейсов
- описание отчётов
- настройку авторизации доступа
- настройку системы архивирования

Современные ИС характеризуются следующими особенностями

- сложность описания, требующая тщательного моделирования и анализа данных и процессов
- наличие совокупности тесно взаимодействующих компонентов, имеющих свои локальные задачи и цели функционирования
- отсутствие прямых аналогов, ограничивающее возможность использования типовых проектных решений
- необходимость интеграции существующих и вновь разрабатываемых приложений
- функционирование в неоднородной среде на нескольких аппаратных платформах

Ручная разработка порождает следующие проблемы

- неадекватная спецификация требований
- неспособность обнаруживать ошибки в проектных решениях
- низкое качество документации, снижающее эксплуатационные качества
- затяжной цикл и неудовлетворительные результаты тестирования

CASE (Computer Aided Software Engineering)

- Понятие CASE первоначально было ограниченно только задачами автоматизации разработки ИС
- В настоящее время понятие CASE охватывает все процессы ЖЦ ИС

Основные характерные особенности

- мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности
- интеграция отдельных компонент CASE-средств, обеспечивающая управляемость процессом разработки ИС
- использование специальным образом организованного хранилища проектных метаданных (репозитория)

Тема 3

ВИЗУАЛЬНОЕ МОДЕЛИРОВАНИЕ

3.1 Основные понятия визуального моделирования

- Моделью ИС - формализованное описание системы на определённом уровне абстракции
- Модели строятся для того, чтобы понять и осмыслить структуру и поведение будущей системы, облегчить управление процессом её создания
- **Визуальное моделирование** – это способ восприятия проблем с помощью зримых абстракций, воспроизводящих понятия и объекты реального мира
- Графические модели представляют собой средства для визуализации, описания, проектирования и документирования архитектуры системы

Под архитектурой понимается набор основных правил, определяющих организацию системы:

- совокупность структурных элементов системы и связей между ними
- поведение элементов системы в процессе их взаимодействия
- иерархию подсистем, объединяющих структурные элементы
- архитектурный стиль (используемые методы и средства описания архитектуры, а также архитектурные образцы)

Архитектура ИС предусматривает различные представления, служащие разным целям:

- представлению функциональных возможностей системы
- отображению логической организации системы
- описанию физической структуры программных компонентов в среде реализации
- отображению структуры потоков управления и аспектов параллельной работы
- описанию физического размещения программных компонентов на базовой платформе

3.2 Структурные методы анализа и проектирования ИС

- Методы структурного анализа и проектирования стремятся преодолеть сложность больших систем путём расчленения их на "чёрные ящики" и иерархической их организации

Критерий разбиения сложной системы:

- каждый "чёрный ящик" должен реализовывать единственную функцию системы
- функция каждого "чёрного ящика" должна быть легко понимаема независимо от сложности её реализации
- связь между "чёрными ящиками" должна вводиться только при наличии связи между соответствующими функциями системы
- связи между "чёрными ящиками" должны быть простыми, насколько это возможно, для обеспечения независимости между ними

Для структурных методов характерно:

- разбиение системы на уровни абстракции с ограничением числа элементов на каждом из уровней (3 - 7)
- ограниченный контекст, включающий лишь существенные на каждом уровне детали
- использование строгих формальных правил записи
- последовательное приближение к конечному результату

В структурном анализе и проектировании используются различные модели, описывающие:

- функциональную структуру системы
- последовательность выполняемых действий
- передачу информации между функциональными процессами
- отношения между данными

Наиболее распространенные модели

- функциональная модель SADT (Structured Analysis and Design Technique)
- модель IDEF3
- диаграммы потоков данных (DFD – Data Flow Diagrams)
- модель "сущность – связь" (ERM – Entity-Relationship Model)

Метод функционального моделирования SADT (IDEF0)

- Метод SADT разработан Дугласом Россом (SoftTech, Inc.) в 1969 году для моделирования искусственных систем средней сложности
- Метод SADT представляет собой совокупность правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области

При построении модели выполняются следующие правила:

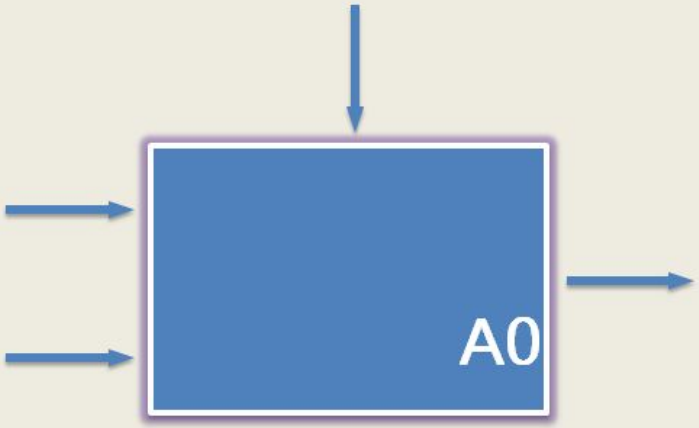
- ограничение количества блоков на каждом уровне декомпозиции
- связность диаграмм
- уникальность меток и наименований
- синтаксические правила для графики
- разделение входов и управлений
- отделение организации от функции

- Результатом применения метода SADT является модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга
- Диаграммы – главные компоненты модели, на них все функции и интерфейсы системы представлены как блоки и дуги соответственно
- Место соединения дуги с блоком определяет тип интерфейса

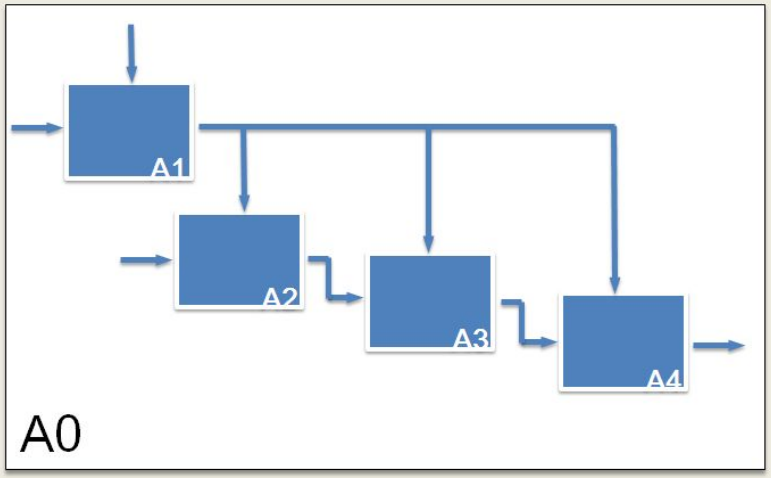
- Важной особенностью метода SADT является постепенное введение всё больших уровней детализации по мере создания диаграмм, отображающих модель
- Построение SADT-модели заключается в выполнении следующих действий:
  - сбор информации об объекте, определение его границ
  - определение цели и точки зрения модели
  - построение, обобщение и декомпозиция диаграмм
  - критическая оценка, рецензирование и комментирование
- Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые изображены в виде блоков
- Детали каждого из основных блоков показаны в виде блоков на других диаграммах
- Каждая детальная диаграмма является декомпозицией блока из диаграммы предыдущего уровня

- Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые изображены в виде блоков
- Детали каждого из основных блоков показаны в виде блоков на других диаграммах
- Каждая детальная диаграмма является декомпозицией блока из диаграммы предыдущего уровня

Общее представление



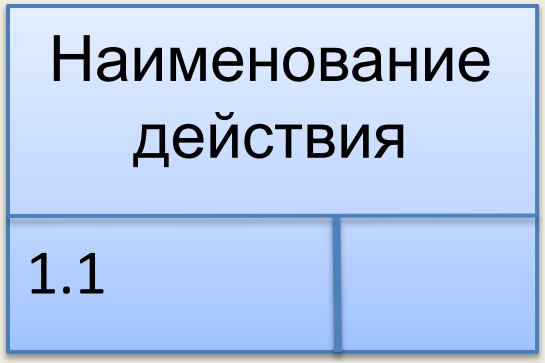
Более детальное представление



- На SADT-диаграммах не указаны явно ни последовательность, ни время
- Механизмы показывают средства, с помощью которых осуществляется выполнение функций
- Каждый блок на диаграмме имеет свой номер
- Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая, в свою очередь, может быть далее детализирована с помощью необходимого числа диаграмм

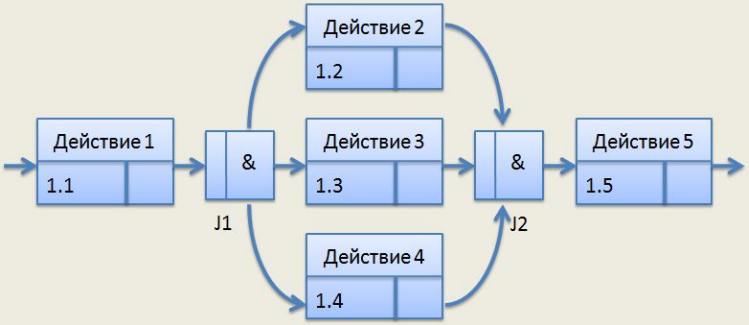
Метод моделирования процессов IDEF3

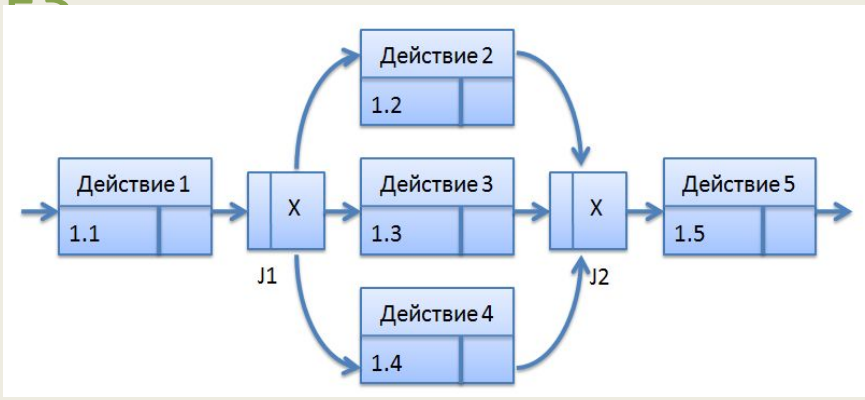
- Метод моделирования IDEF3 был разработан в конце 1980-х годов для закрытого проекта ВВС США
- Метод предназначен для таких моделей процессов, в которых важно понять последовательность выполнения действий и взаимозависимости между ними
- Основная единица модели – диаграмма
- Другой важный компонент модели – действие (единица работы)
- Действие отображается в виде прямоугольника и имеет уникальный номер

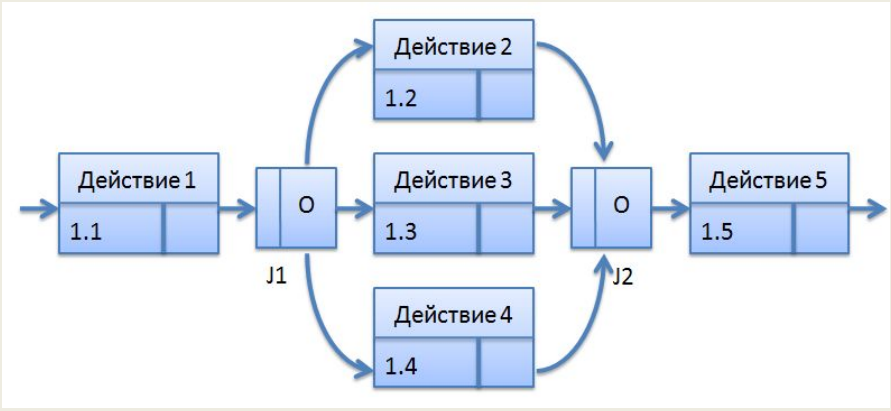


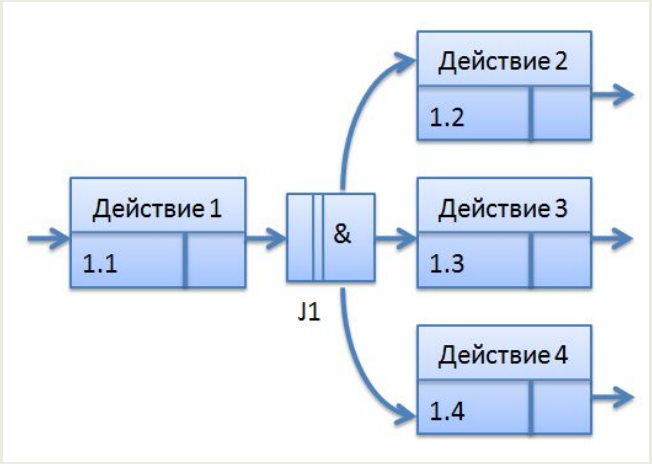
- Завершение одного действия может инициировать начало сразу нескольких других действий или, наоборот, определённое действие может требовать завершения нескольких других действий до начала своего выполнения
  - И (&)
  - ИСКЛЮЧАЮЩЕЕ ИЛИ (X)
  - ИЛИ (0)

Изображение типов соединений в диаграмме IDEF3









Существуют случаи, когда время начала или окончания параллельно выполняемых действий должно быть одинаковым (синхронным)

- Синхронное разворачивающее соединение не обязательно должно иметь парное сворачивающее соединение
- Возможны ситуации синхронного окончания асинхронно начавшихся действий
- Все соединения на диаграммах должны быть парными. Типы соединений не обязательно должны совпадать

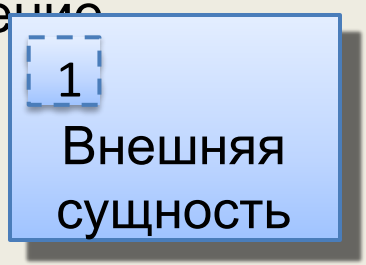
- Диаграммы потоков данных представляют собой иерархию функциональных процессов, связанных потоками данных
- Цель– продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами

Основные компоненты диаграмм потоков данных:

- внешние сущности
- системы и подсистемы
- процессы
- накопители данных
- потоки данных

Внешняя сущность

- представляет собой материальный объект или физическое лицо, представляющее собой источник или приёмник информации
- Графическое изображение



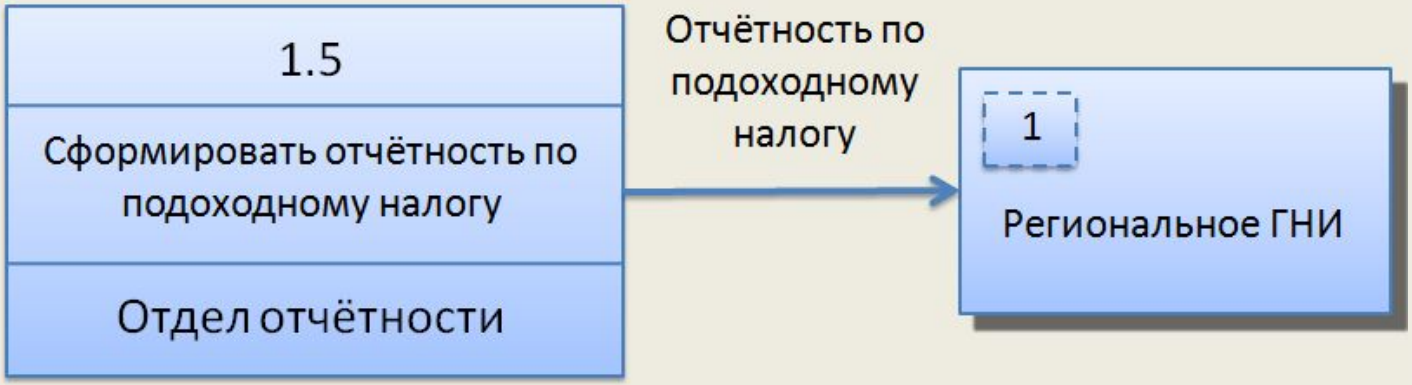
Подсистема (система)

- Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определённым алгоритмом
- Физически процесс может быть реализован различными способами
- Накопитель данных – это абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причём способы помещения и извлечения могут быть любыми

D1	Имя накопителя

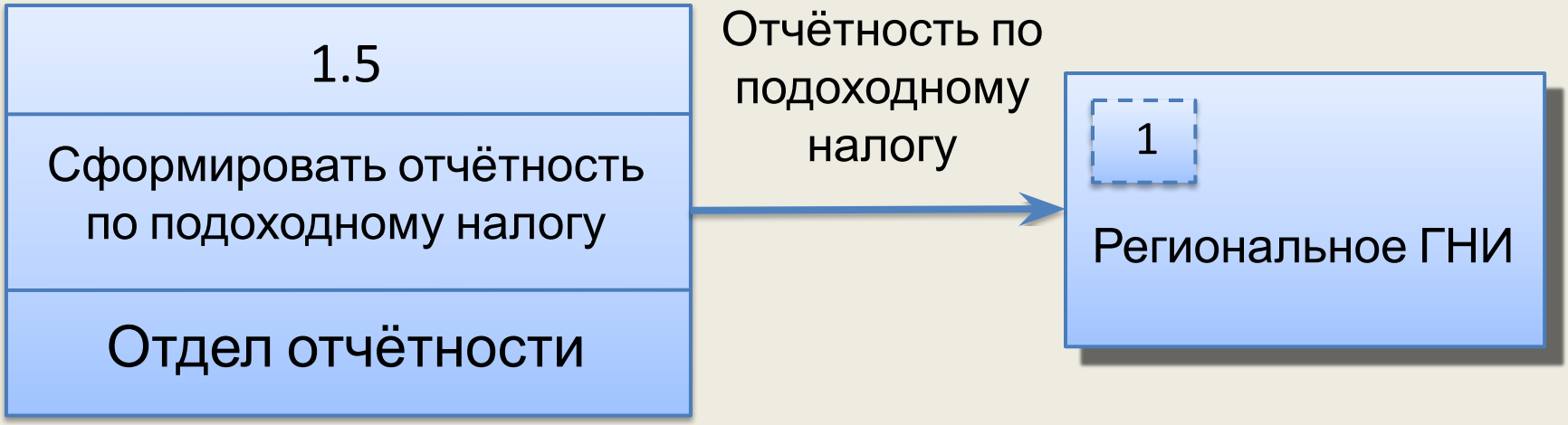
Потоки данных

определяют информацию, передаваемую через некоторое соединение от источника к приёмнику



Потоки данных

- определяют информацию, передаваемую через некоторое соединение от источника к приёмнику



Модель "сущность-связь"

- Цель - обеспечении разработчика ИС концептуальной схемой БД в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных
- С её помощью определяются важные для предметной области объекты, их свойства и отношения друг с другом
- **Сущность** – это реальный или воображаемый объект, имеющий существенное значение для рассматриваемой предметной области
- Конкретного представителя сущности называют **экземпляром сущности**
- **Атрибут** – это любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности
- Каждая сущность должна обладать уникальным идентификатором

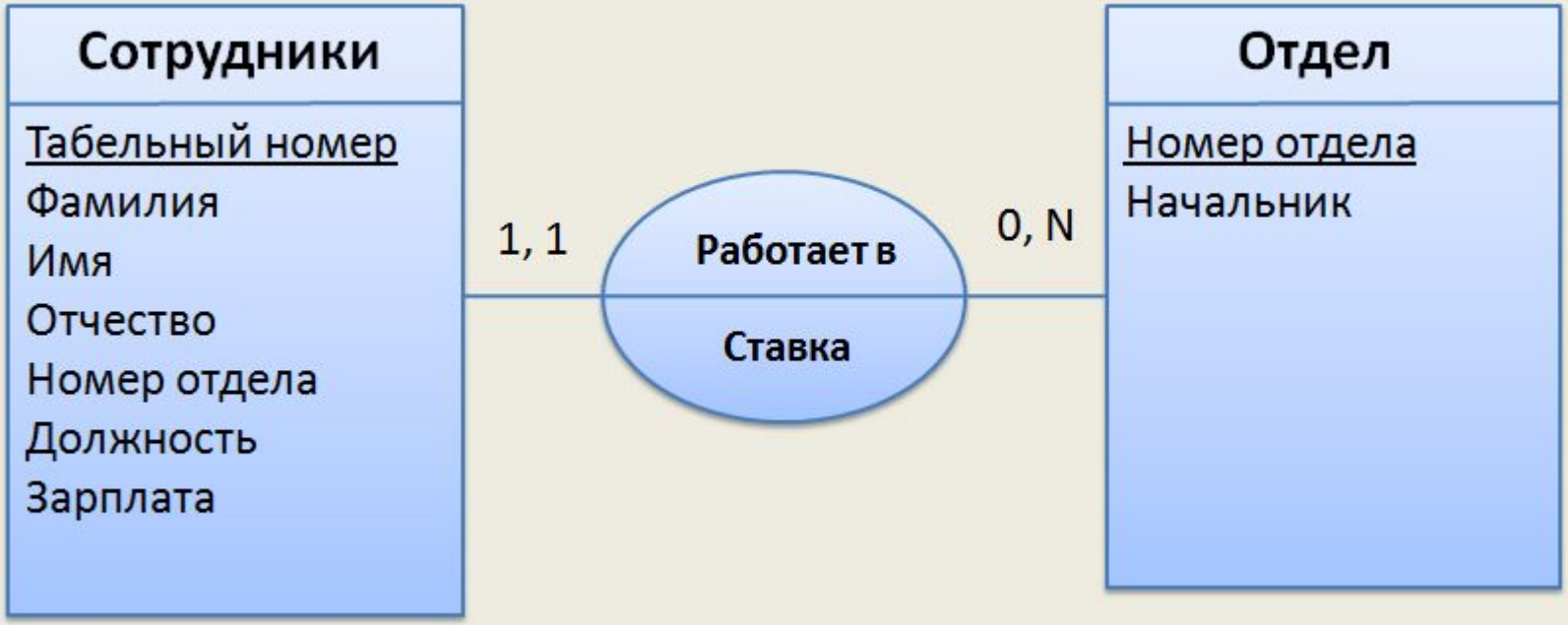
Каждая сущность может обладать любым количеством связей с другими сущностями модели

- **Связь** – это поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области
- Связям можно даваться имя
- Каждая связь имеет степень, которая определяет количество сущностей, участвующих в данной связи
- Каждая связь имеет две важные характеристики:
  - **Класс принадлежности** характеризует обязательность участия экземпляра сущности в связи
  - **Мощностью связи** называется максимальное число экземпляров сущности, которое может быть связано с одним экземпляром данной сущности

В зависимости от значения мощности связь может иметь один из следующих трех типов:

- 1:1
- 1:n
- m:n

Связи могут иметь атрибуты



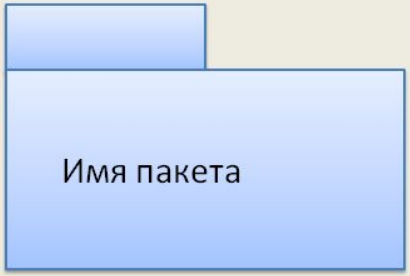
3.3 Унифицированный язык моделирования

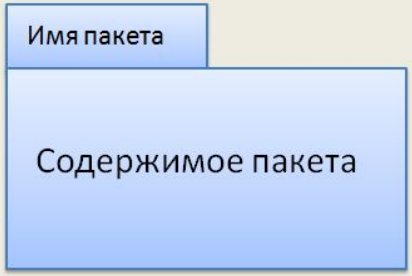
1. Развитие средств объектно-ориентированного анализа и проектирования сложных систем
2. Основные понятия языка UML
3. Компоненты языка UML

- **Унифицированный язык моделирования** (UML – Unified Modeling Language) стал промышленным стандартом для разработки и проектирования программного обеспечения
- UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределённых Web-приложений и встроенных систем реального времени

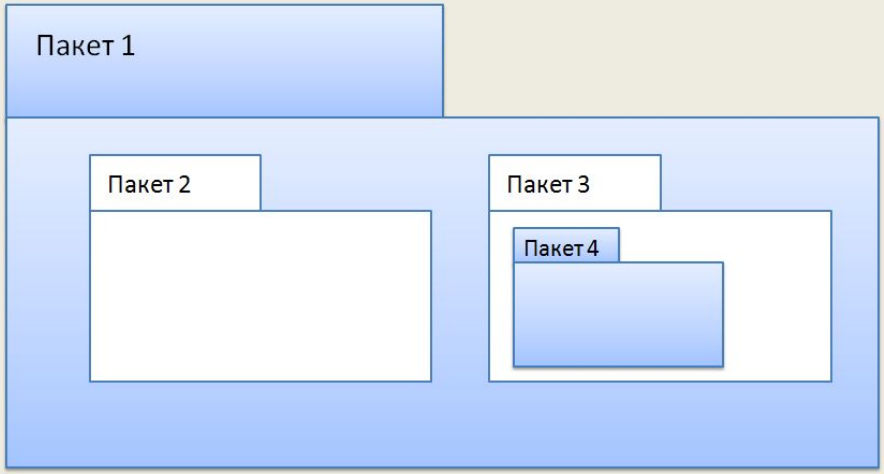
Основные понятия языка UML

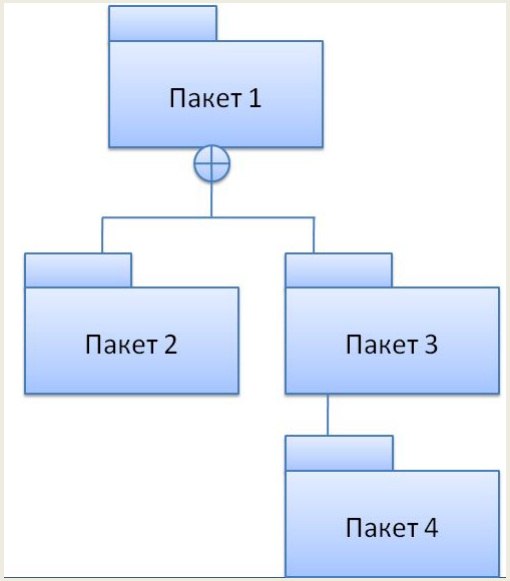
1. **Пакеты** служат основным способом организации элементов модели ИС
   - Каждый пакет владеет всеми элементами, которые в него включены





Вложенные пакеты





2. **Подсистема** – вид пакета, описывающего определённую часть системы, выделенную в единое целое по реализационным или функциональным соображениям

3. **Система** – набор подсистем, организованных для достижения определённого результата и описываемых с помощью совокупности моделей

4. **Модель** – особый тип пакета, представляющий семантически замкнутую абстракцию системы. Она является полным и внутренне непротиворечивым упрощением реальной физической системы
- Для одной и той же физической системы могут быть определены различные модели, описывающие систему с различных представлений

5. **Представление** определяет способ видения системы, на основе которого создаётся её модель
- Представление включает набор графических нотаций и их семантику
- Объектное моделирование использует многообразие приёмов представления, отражающих процесс объектной декомпозиции с помощью логической и физической структуры модели, а также их статические и поведенческие аспекты

Компоненты языка UML

- UML включает набор графических элементов и правила для объединения этих элементов
- Диаграммы используются для отображения различных представлений системы
- Модель UML описывает, что должна делать система, но ничего не сообщает о том, как она будет реализована

Тема 4

ТЕХНОЛОГИИ СОЗДАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

4.1 Основные определения

- **Технология создания ИС** – это упорядоченная совокупность взаимосвязанных технологических процессов в рамках ЖЦ ИС
- **Технологический процесс** – это совокупность взаимосвязанных технологических операций
- **Технологическая операция** – это основная единица работы
- **Инструментальное средство** (CASE-средство) – это программное средство, обеспечивающее автоматизированную поддержку деятельности, выполняемой в рамках технологических операций

4.2 Требования, предъявляемые к технологии создания информационных систем

Реальное применение ТС ИС в конкретном проекте невозможно без выработки ряда стандартов:

- проектирования
- оформления проектной документации
- пользовательского интерфейса

Стандарт проектирования должен устанавливать:

- набор необходимых моделей и степень их детализации
- правила фиксации проектных решений на диаграммах
- требования к конфигурации рабочих мест разработчиков
- механизм обеспечения совместной работы над проектом

Стандарт оформления проектной документации должен устанавливать:

- комплектность, состав и структуру документации на каждой стадии проектирования
- требования к её оформлению
- правила подготовки, рассмотрения, согласования и утверждения документации
- требования к настройке издательской системы
- требования к настройке CASE-средств

Стандарт интерфейса пользователя должен устанавливать:

- правила оформления экранов, состав и расположение окон и элементов управления
- правила использования клавиатуры и мыши
- правила оформления текстов помощи
- перечень стандартных сообщений
- правила обработки реакции пользователя

Технология создания информационных систем должна поддерживать следующие процессы:

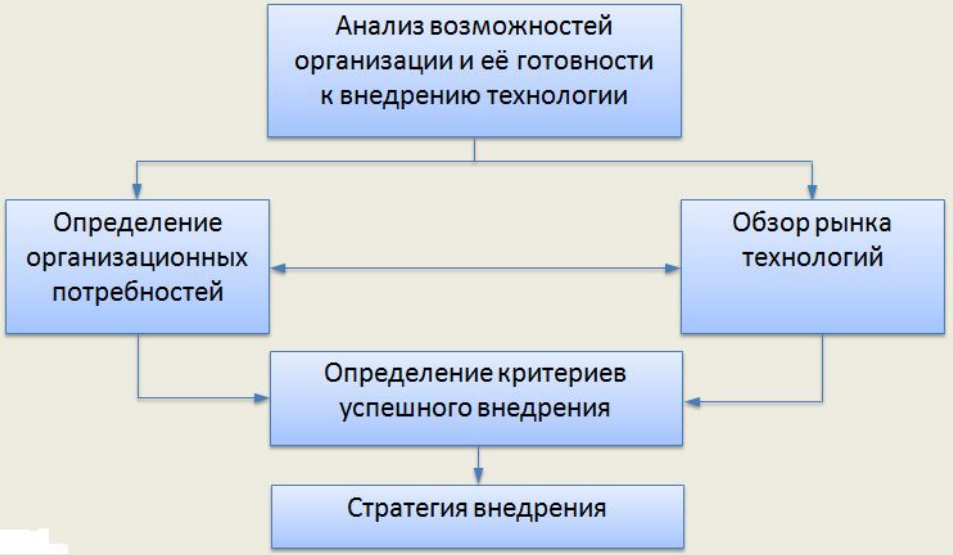
- управление требованиями
- анализ и проектирование ИС
- разработка ИС
- эксплуатация
- сопровождение
- документирование
- управление конфигурацией и изменениями
- тестирование
- управление проектом

Внедрение технологий создания информационных систем

В процессе внедрения ТС ИС можно выделить следующие этапы:

- определение потребностей в ТС ИС
- определение требований, предъявляемых к ТС ИС
- оценка вариантов ТС ИС
- выбор ТС ИС
- адаптация ТС ИС к условиям применения

Определение потребностей в технологии создания информационной системы



- Цель – достижение понимания потребностей организации в ТС ИС
- анализ возможностей организации в отношении её технологической базы, персонала и используемого ПО
- определение того, насколько организация способна воспринять как немедленные, так и долгосрочные последствия внедрения технологии
- организационные потребности следуют непосредственно из проблем организации и целей, которые она стремится достичь

Оценка и выбор технологий создания информационных систем

Входной информацией для процесса оценки и выбора являются:

- требования к ТС ИС
- цели и ограничения проекта
- данные о доступных технологиях

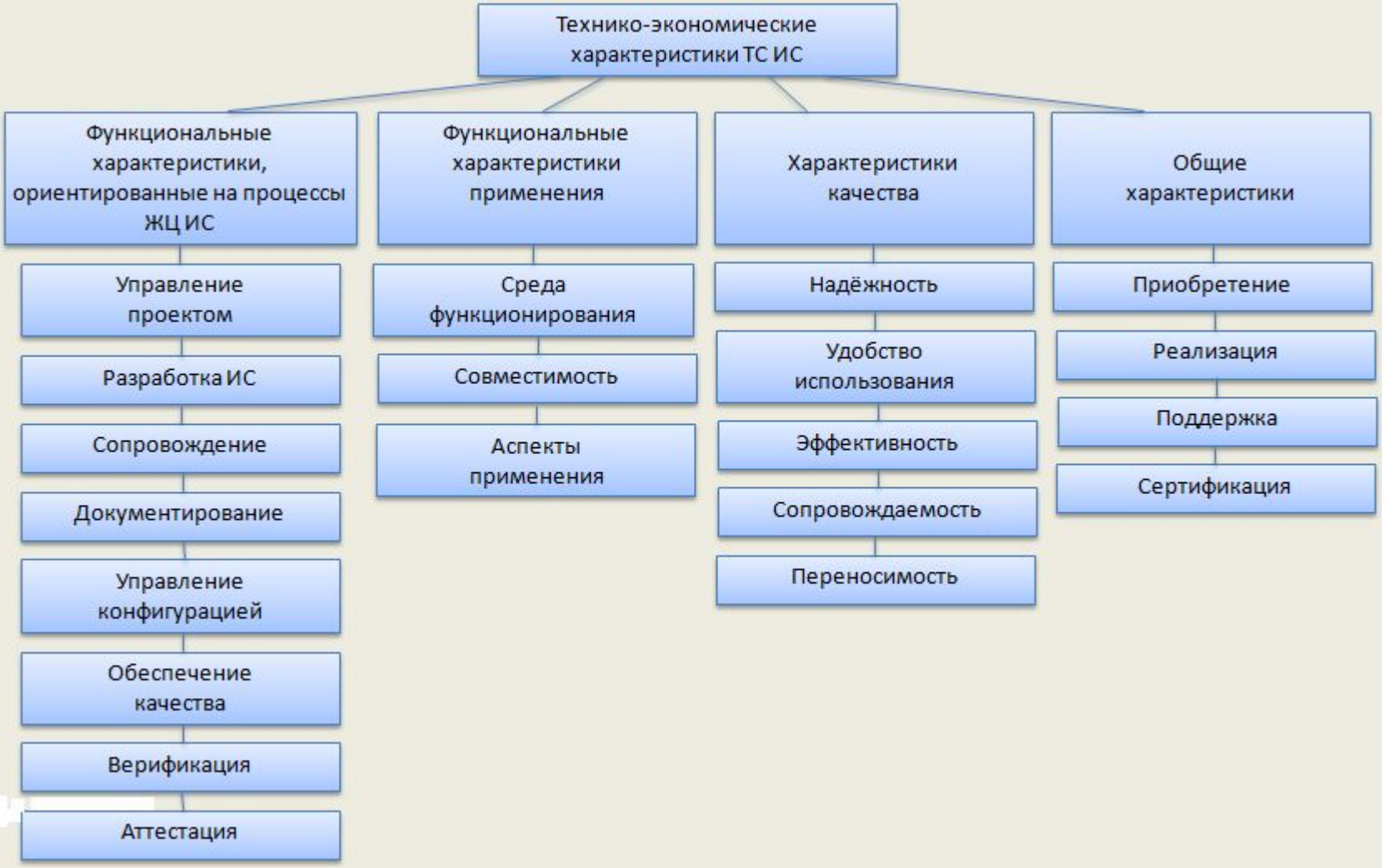
Цель – определение функциональности и качества ТС ИС для последующего выбора

Потребности организации в ТС ИС должны соразмеряться с реальной ситуацией на рынке или собственными возможностями разработки

Оценка и накопление информации о технологиях могут выполняться следующими способами:

- анализ технологий и документации поставщика
- опрос реальных пользователей
- анализ результатов проектов, использовавших данные технологии
- просмотр демонстраций и опрос демонстраторов
- выполнение тестовых примеров
- применение технологий в пилотных проектах
- анализ любых доступных результатов предыдущих оценок

Исходными данными для выбора и оценки применимости ТС ИС



- Критерии успешного внедрения должны позволять количественно оценивать степень удовлетворения каждой из потребностей, связанных с внедрением
- Как правило, внедрение ТС ИС осуществляет для повышения продуктивности процессов разработки и сопровождения ИС, а также качества результатов разработки

Подходы к разработке стратегии внедрения технологии создания информационных систем

Нисходящий

Восходящий

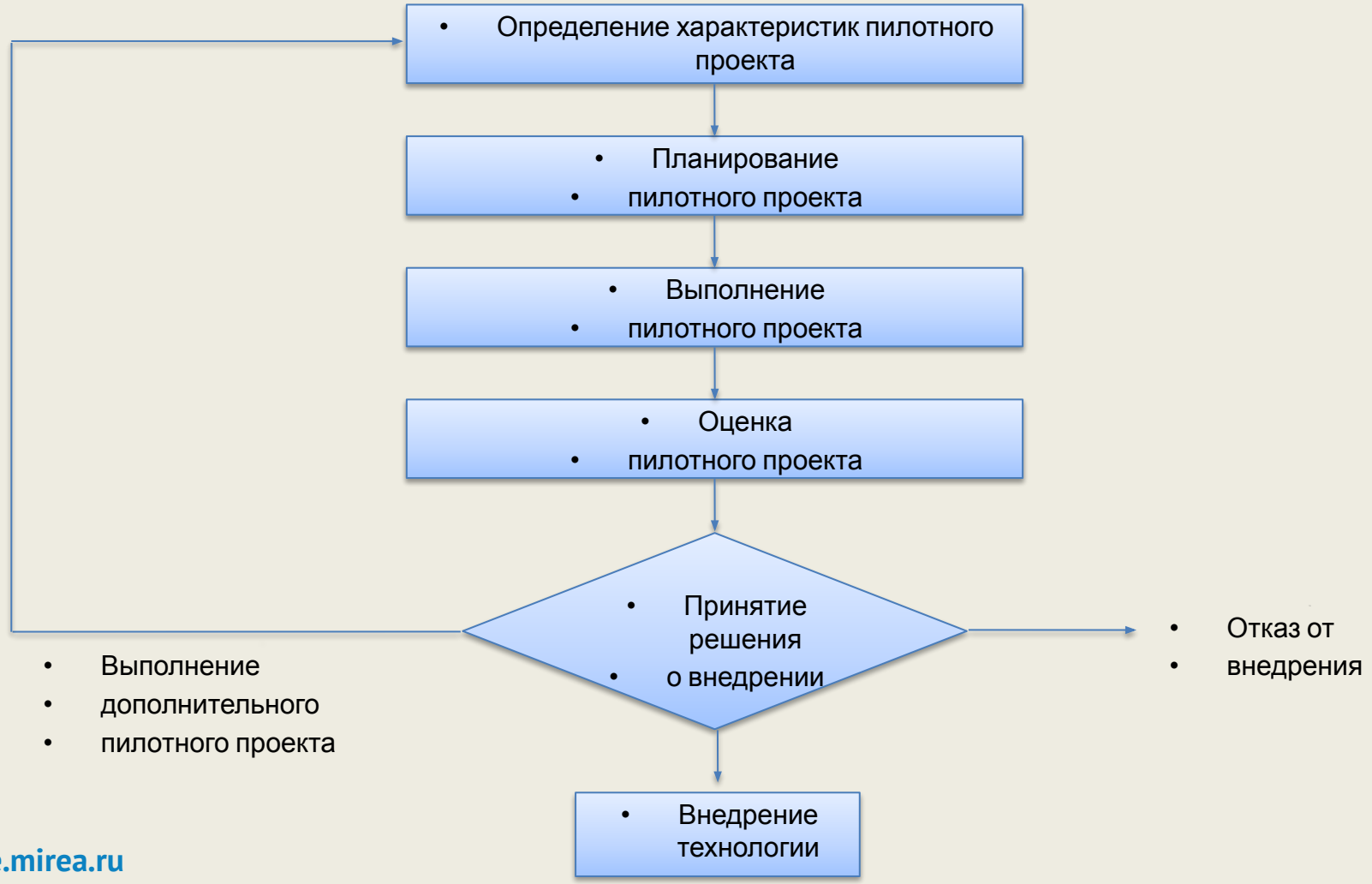
Выполнение пилотного проекта

- С целью проверки правильности выбора ТС ИС, перед её полномасштабным внедрением, выполняется пилотный проект
- Такой проект должен обладать многими характеристиками реальных проектов, для которых предназначено данное средство

Характеристики пилотного проекта :

1. Область применения
2. Представительность
3. Критичность
4. Авторитетность
5. Характеристики проектной группы

Шаги пилотного проекта



При внедрении какой-либо ТС ИС следует учитывать, что:

- ТС ИС не обязательно даёт немедленный эффект, он может быть получен только спустя какое-то время
- реальные затраты на внедрение ТС ИС обычно намного превышают затраты на её приобретение
- технология обеспечивает возможности для получения существенной выгоды только после успешного завершения процесса её внедрения

Примеры технологий создания ИС

- Многие организации-разработчики программных продуктов год за годом накапливали профессиональные знания в области ТС ИС, которые материализовались в виде практических рекомендаций, документации, обучающих программ и книг
- На сегодняшний день практически все ведущие компании – разработчики технологий и программных продуктов располагают развитыми технологиями создания ИС, которые создавались как собственными силами, так и за счёт приобретения продуктов и технологий, созданных небольшими специализированными компаниями

Тема 5

ПРЕДМЕТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ

- Моделирование предметной области и проектирование архитектуры программных объектов на этой основе является ключевой методологией разработки сложных программных систем
- Предметно-ориентированное проектирование (*domain-d riven design, DDD*) – это система взглядов и подходов, используемых, при проектировании сложных программных систем (Эрик Эванс . Предметно-ориентированное проектирование: структуризация сложных программных систем. – М.: 000 "И.Д. Вильямс", 2011. – 448 с.)

Гибкая методика разработки (*agile development processes*)

1. **Итеративный характер разработки**. Итерационная разработка программ лежит в основе гибкой методологии
2. **Тесное взаимодействие между разработчиками и специалистами в предметной области.** Особенность предметно-ориентированного проектирования такова, что в его процессе в модель закладывается огромное количество знаний, отражающих глубокое понимание предметной области и концентрацию на её ключевых концепциях. Это требует сотрудничества между теми, кто владеет предметом, и теми, кто умеет писать программы. Так как разработка носит итеративный характер, сотрудничество должно продолжаться на протяжении всего срока существования проекта

5.1 Модель предметной области в работе

- Модель – это упрощение, это такая интерпретация реальности, при которой из явления извлекаются существенные для решения задачи аспекты, а лишние детали игнорируются
- Всякая компьютерная программа имеет применение в той или иной области деятельности или интересов своего пользователя. Область знания или деятельности, в которой пользователь использует ПО, и есть его предметная область.
- Чтобы создать программу действительно ценную для пользователей из какой-либо области деятельности, разработчики должны использовать знания, относящиеся к этой области
- Модель представляет собой специально отобранный и сознательно упрощённые знания в структурированной форме. Правильна модель придаёт информации смысл и позволяет сконцентрироваться на проблеме

Роль и выбор модели

- Выбор модели в предметно-ориентированном проектировании определяется тремя фундаментальными способами её использования при разработке программы

1. Модель и архитектура программы взаимно определяют друг друга
2. Модель лежит в основе языка, на котором говорят все члены группы разработчиков
3. Модель это выделенное знание

Модель и архитектура программы взаимно определяют друг друга

- Именно тесная связь между моделью и её программной реализацией определяет важность и нужность самой модели, а также гарантирует, что проделанный при её разработке анализ отражен в конечном результате, т.е. в работающей программе
- Связь между моделью и реализацией помогает также в процессе дальнейшей доработки программы и выпуска её новых версий, поскольку программный код можно интерпретировать, основываясь на понимании модели

Модель это выделенное знание

- Модель представляет собой согласованный между разработчиками способ структуризации знаний из предметной области, а также выделения элементов, представляющих наибольший интерес
- Модель передаёт наш способ мыслить о предмете, выраженный в выборе терминов, выделении понятий и установке связей между ними

- Модель предметной области может служить основой общего языка для коммуникации в рамках проекта по разработке ПО
- Модель – это набор понятий, существующих в головах создателей проекта, вместе с названиями (терминами), отношениями и взаимосвязями, отражающими их понимание предмета
- Коммуникация на основе модели не ограничивается рисованием диаграмм (например, на унифицированном языке моделирования)
- Благодаря модели повышается полезность как текстовой документации, так и неформальных диаграмм или дискуссий

5.2 Коммуникация и язык при проектировании информационных систем

- Чтобы разработать гибкую, ресурсоёмкую архитектуру программы, группе разработчиков необходимо иметь богатый возможностями общий язык
- Специалисты в предметных областях не слишком хорошо понимают технический жаргон разработчиков, но сами при этом пользуются собственным профессиональным жаргоном
- Разработчики могут понимать и обсуждать систему в описательных и функциональных терминах, никак не привязываясь к тому смысл, который в свой язык вкладывают специалисты
- Программисты, работающие над разными частями проекта, могут вырабатывать собственные архитектурные концепции и способы описания предметной области

Письменная проектная документация

- Чтобы все участники были в курсе устройства модели, для стабильного обмена информацией группе любой численности необходимо иметь некоторое количество письменных документов
- Как только документ приобретает фиксированную форму, он часто теряет связь с динамично изменяющимися обстоятельствами проекта
- В каждой методологии гибкой разработки программ – своя стратегия работы с документацией (например, в экстремальном программировании предлагается вообще исключить проектную документацию)

- Во всякой методологии гибкой разработки программ между членами группы распределяются роли, и естественным образом возникают неформальные специализации
- Эффективность всего процесса проектирования сильно зависит от качества и согласованности множества мелких проектных и технических решений
- В проектировании по модели фрагмент кода есть выражение модели; если меняется код – надо менять модель. Поэтому лучше организовать проект так, чтобы программисты имели возможность заниматься моделированием
- Технический сотрудник, делающий свой вклад в модель, должен некоторое время работать на кодом
- Каждый разработчик должен участвовать на каком-то уровне в дискуссиях о модели и иметь контакты со специалистами в предметной области

5.3 Структурные элементы предметно-ориентированного проектирования

Изоляция предметной области

- Та часть программы, которая собственно и решает задачи из предметной области, – это, как правило, лишь небольшой фрагмент всей программной системы
- Чтобы применить наиболее эффективные подходы, необходимо посмотреть на элементы модели как на систему
- Необходимо отграничить объекты предметной области от остальных функций системы и тем самым избежать путаницы между понятиям из этой области и программными технологиями

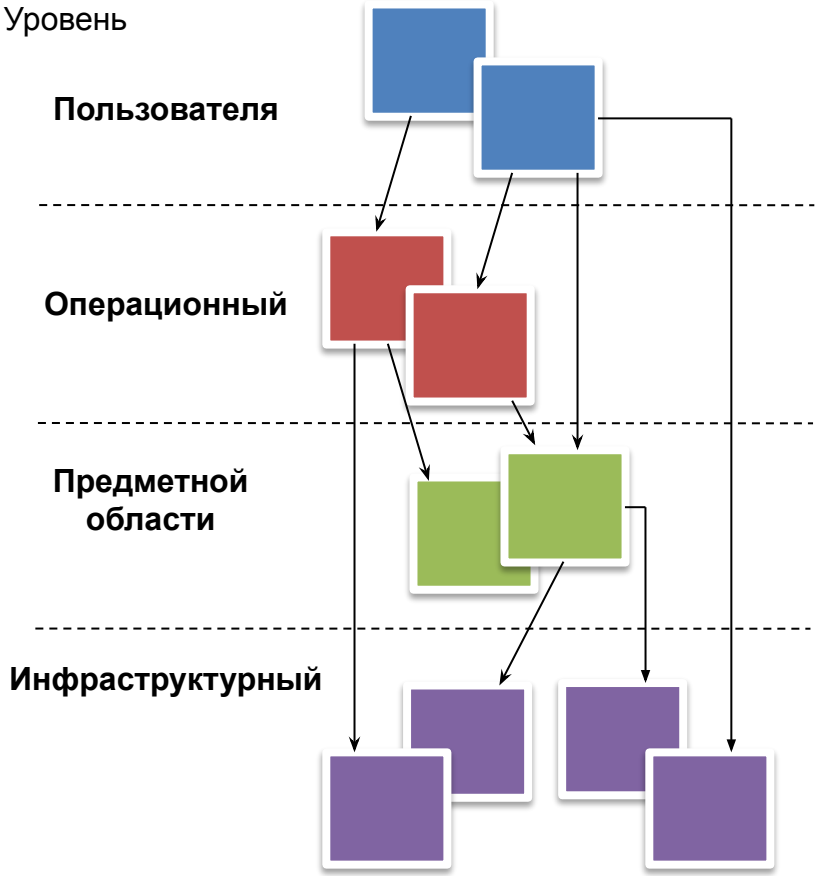
Многоуровневая архитектура

- Структурные элементы и код программы предназначены для выполнения самых разных задач
  - обработка данных и команды, введённых пользователем
  - выполнение прикладных операций
  - обращение к базам данных
  - пересылка данных по сетям
  - отображение информации для пользователя
  - и т.д.
- Разработка программ, которые могут выполнять сложные задачи, требует разделения обязанностей, которое бы позволило сосредоточиться на принципиальных частях архитектуры в отдельности, в изоляции от других
- В то же время, сложные взаимосвязи в пределах системы необходимо поддерживать, несмотря на разделение

- Существует много способов членения программной системы, но по накопленному в программной индустрии опыту преимущество отдаётся многоуровневой архитектуре, состоящей из нескольких достаточно стандартных уровней
- Важнейший принцип многоуровневости состоит в том, что любой элемент какого-нибудь уровня зависит от работы только других элементов того же уровня или элементов более низких уровней
- Передача информации наверх должна выполняться косвенными способами
- Ценность многоуровневости состоит в том, что каждый уровень специализируется на конкретном аспекте программы
- Такая специализация позволяет выполнять более связное проектирование каждого аспекта, и получающуюся архитектуру намного легче интерпретировать

Используемые уровни

- В наиболее успешных архитектурах используются, в том или ином виде, следующие четыре концептуальных уровня
  - Интерфейс пользователя (User Interface) или Уровень представления (Presentation Layer)
  - Операционный уровень или Уровень прикладных операций (Application Layer)
  - Уровень предметной области (Domain Layer) или Уровень модели (Model Layer)
  - Инфраструктурный уровень (Infrastructure Layer)



Задачи, решаемые уровнями

Интерфейс пользователя	Отвечает за вывод информации пользователю и интерпретирование его команд. Внешним действующим субъектом может быть не человек, а другая компьютерная система.
Операционный уровень	Определяет задачи, которые должна решить задача, и распределяет их межу объектами, выражающими суть предметной области. Задания, выполняемые этим уровнем, имеют смысл для пользователя-специалиста или же необходимы для интерактивного взаимодействия с операционными уровнями других систем. В этом уровне не содержатся ни знания, ни деловые регламенты, а только выполняется координирование задач и распределение работы между совокупностями объектов предметной области на следующем, более низком уровне.
Уровень предметной области	Отвечает за представление понятий прикладной предметной области, рабочие состояния, деловые регламенты. Именно здесь контролируется и используется текущее состояние прикладной модели, пусть даже технические подробности манипуляций данными делегируются инфраструктуре. Этот уровень является главной, алгоритмической частью программы.
Инфраструктурный уровень	Обеспечивает непосредственную техническую поддержку для верхних уровней: передачу сообщений на операционном уровне, непрерывность существования объектов на уровне модели, вывод элементов управления на уровне пользовательского интерфейса и т.д. Инфраструктурный уровень может также брать на себя поддержку схемы взаимосвязей между четырьмя уровнями через архитектурную среду приложения.

Спасибо за внимание!