1. Понятие информационной системы. Определение ИС. Специфика ПО для ИС. Связь информационных систем и бизнес-процессов. Цели проектирования ИС.

В широком смысле информационная система - совокупность технического, программного и организационного обеспечения, а также персонала, предназначенная для того, чтобы своевременно обеспечивать людей надлежащей информацией.

Федеральный закон РФ от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и о защите информации»:

«Информационная система — совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств».

Стандарт ISO/IEC 2382-1 дает следующее определение:

«Информационная система — система обработки информации, работающая совместно с организационными ресурсами, такими как люди, технические средства и финансовые ресурсы, которые обеспечивают и распределяют информацию».

В узком смысле информационной системой называют только подмножество компонентов ИС в широком смысле, включающее базы данных, СУБД и специализированные прикладные программы. ИС в узком смысле рассматривают как программно-аппаратную систему, предназначенную для автоматизации целенаправленной деятельности конечных пользователей, обеспечивающую, в соответствии с заложенной в нее логикой обработки, возможность получения, модификации и хранения информации.

Специфика ИС:

- Любая информационная система предназначена для сбора, хранения и обработки информации. Поэтому в основе любой информационной системы лежит среда хранения и доступа к данным.
- Информационные системы ориентируются на конечного пользователя. Поэтому информационная система обязана обладать простым, удобным, легко понятным интерфейсом.

Связь информационных систем и бизнес-процессов.

Технологии внедрения информационных систем:

1. Технология построения системы по моделям "как надо", без попыток программирования действующих сейчас алгоритмов. Практика создания систем по модели "как есть" показала, что автоматизация без проведения реинжиниринга бизнес - процессов и модернизации существующей системы управления не приносит желаемых результатов и не эффективна. Ведь использование в работе программных приложений - это не просто сокращение бумажных документов и рутинных операций, но и переход на новые формы ведения документооборота, учёта и отчётности.
2. Технология построения систем с подходом "сверху вниз". Если решение об автоматизации принято и одобрено высшим руководством, то внедрение программных модулей осуществляется с головных предприятий и подразделений, а процесс построения корпоративной системы проходит

гораздо быстрее и эффективнее, чем при внедрении системы первоначально в низовые подразделения. Только при внедрении "сверху вниз" и активном содействии руководства можно изначально правильно оценить и провести весь комплекс работ без незапланированных издержек.

3. Технология поэтапного внедрения. Поскольку комплексная автоматизация - это процесс, в который вовлекаются практически все структурные подразделения предприятия, технология поэтапного внедрения является наиболее предпочтительной. Первыми объектами автоматизации становятся те участки, на которых в первую очередь необходимо наладить процесс учета и формирования отчетных документов для вышестоящих органов и смежных подразделений.
4. Привлечение к разработке будущих пользователей. При выполнении работ по комплексной автоматизации фирмой - интегратором меняются функции отделов информационных технологий фирмы - заказчика, и возрастает их роль в общем процессе перехода предприятия на прогрессивные методы управления.

Подготовка бизнес-процессов:

- Анализ и формирование рекомендаций по совершенствованию бизнес-процессов планирования операционной деятельности, её исполнения, а также ведения нормативных данных для поддержки операционной деятельности.
- Анализ и формирование рекомендаций по достижению соответствия бизнес - процессов рекомендациям методики ИС.
- Разработка моделей бизнес - процессов сбыта, производства, закупок, планирования и других, в соответствии с предметной областью проекта, на различных уровнях иерархии плановых решений, необходимых предприятию Заказчика бизнес - процессов, которые будут поддерживаться системой.

Цели проектирования ИС:

- Улучшение организации учетной, плановой и аналитической работы;
- Разработка рациональной технологии решения задач и получения результатной информации;
- Получение графиков прохождения информации как внутри, так и между производственными и функциональными подразделениями;
- Создание БД, обеспечивающей оптимальное использование информации, касающейся планирования, учета и анализа хозяйственной деятельности;
- Создание нормативно-справочной информации.

2. Жизненный цикл проектирования ИС. Классификация ИС. Представление результатов проектирования ИС.

(Добавлено из записей лекций)
Жизненный цикл проектирования ИС.
Этапы:

1. Постановка задачи;
2. Анализ и моделирование объекта автоматизированной системы;
3. Выявление и формулирование требований;
4. Концепция проектирования ИС (Общее виденье, не столь детальное описание как в архитектуре, расплывчатое описание, включает общие видение и некие сценарии того, как это будет жить);
5. Архитектура (Подробность зависит от целей использования, бывает разного уровня);
6. Выбор технологий (Этап выбора технологии может возникнуть в любом месте этого списка и в любое время);
7. Верификация и анализ принятых решений (Имеет смысл сделать верификацию на соответствие всех принятых решений между собой. Иногда имеет смысл сделать прототипирование узких мест);
8. Отчуждение результата (Например, составление ТЗ. Цель - создать документ или что-то самодостаточно описывающее все этапы, без необходимости пояснения от человека).

(Конец части информации из лекций)
Классификация ИС.

1. <u>По степени распределенности отличают:</u>

- настольные (desktop), или локальные ИС, в которых все компоненты (БД, СУБД, клиентские приложения) находятся на одном компьютере;
- распределённые (distributed) ИС, в которых компоненты распределены по нескольким компьютерам.

Распределённые ИС, в свою очередь, разделяют на:

- файл-серверные ИС (ИС с архитектурой «файл-сервер»);
- клиент-серверные ИС (ИС с архитектурой «клиент-сервер»).

В файл-серверных ИС база данных находится на файловом сервере, а СУБД и клиентские приложения находятся на рабочих станциях.В клиент-серверных ИС база данных и СУБД находятся на сервере, а на рабочих станциях находятся только клиентские приложения.
В свою очередь, клиент-серверные ИС разделяют на двухзвенные и многозвенные.

- В двухзвенных ИС всего два типа «звеньев»: сервер базы данных, на котором находятся БД и СУБД (back-end), и рабочие станции, на которых находятся клиентские приложения (front-end).
- В многозвенных ИС добавляются промежуточные «звенья»: серверы приложений (application servers). Пользовательские клиентские приложения не обращаются к СУБД напрямую, они взаимодействуют с промежуточными звеньями.

2. <u>По степени автоматизации ИС делятся на:</u>

- автоматизированные: информационные системы, в которых автоматизация может быть неполной (то есть требуется постоянное вмешательство персонала);
- автоматические: информационные системы, в которых автоматизация является полной, то есть вмешательство персонала не требуется или требуется только эпизодически.

3. По характеру обработки данных ИС делятся на:

- информационно-справочные, или информационно-поисковые ИС, в которых целью системы является поиск и выдача информации в удобном виде;
- ИС обработки данных, или решающие ИС, в которых данные подвергаются обработке по сложным алгоритмам. К таким системам в первую очередь относят автоматизированные системы управления и системы поддержки принятия решений.

Представление результатов проектирования ИС.

Результаты обследования представляют объективную основу для формирования технического задания на информационную систему.

(Добавлено из записей лекций)

Разработка ТЗ делается с определенными целями и для определенных лиц.

Цели создания ТЗ:

1. Для внешнего исполнителя (Неточно, пропустил этот момент!)
2. Для организации тендера

(*комм.* Вероятно, первый пункт подразумевает составление ТЗ для создания системы определенным исполнителем, нежели второй вариант, где исполнитель ещё не определён. Соответственно, степень детализации разная.)

ТЗ (Препод ссылается на определение из вики!) – исходный документ для разработки технического объекта

Что устанавливает ТЗ:

1. Назначение технического объекта (ТО)
2. Технические характеристики ТО
3. Показатели качества
4. Технико-экономические требования
5. Требования к документации
6. Спец. требования для конкретных объектов

Вопросы при составлении ТЗ:

1. В каких понятиях составлять и на каком языке?
2. С какой степенью детализации (пример: надо ли прописывать структуру БД или нет?)

В каких понятиях и на каком языке? - В терминах бизнеса, а не техники.

Необходимо различать Техническое Задание с Техническим Проектом.

Технический проект может содержать техническое описание.

ТЗ – пишется в терминах бизнеса, но может включать дополнение в виде технического проекта. Но без нужды этим не надо грузить заказчика.

ТЗ – по-хорошему пишет аналитик (НЕТ ДЕТАЛЕЙ РЕАЛИЗАЦИИ)

Технический Проект – архитектор (ЗДЕСЬ ДЕТАЛИ РЕАЛИЗАЦИИ)
Какова степень детализации ТЗ?

1. Необходимо чтобы было понятно для заказчика, что будет разработано
2. Чтобы была возможность по ТЗ оценить объем разработки

Структура технического задания

1. Общие сведения
2. Назначение и цели создания системы
3. Характеристика объекта автоматизации
4. Требования к системе
5. Состав и содержание работ по созданию системы
6. Порядок контроля и приемки системы
7. Состав и содержание работ по вводу системы в действие
8. Требования к документированию
9. Источники разработки (Перечисляются документы и информационные материалы, на основании которых разрабатывалось ТЗ и которые должны быть использованы при создании системы.)

Общие сведения:

- Идентификатор, название и т.д.
- Шифр темы и номер договора.
- Наименование и реквизиты заказчика и исполнителя.
- Перечень документов, на основании которых создается система (!полезно!).
- Плановые сроки начала и окончания.
- Сведения об источнике и порядке финансирования работ.

Общее отношение к стандарту: если работаем на гос. структуру, то все соблюдаем. Если обычный заказ, то вытаскиваем полезные детали.

Назначение и цели создания системы: система для того-то и того, и (!ВАЖНО!) в соответствии с требованиями настоящего ТЗ

Цели создания ТЗ - не уверены не пишите, если можете явно написать то пишите.

Характеристика объекта автоматизации – тот контекст, в котором предстоит работать в вашей системе.
Включает:

1. Краткие сведения об объекте автоматизации (например: инф. система клиники заказчика включает то-то, и персоналу приходится дублировать то-то, то-то) и ссылки на её документацию.
2. Сведения об условиях эксплуатации системы (!для ИТ почти неактуально!)

Требования к системе (!главный раздел!) - изложение тех требований, на основе которых составляется ТЗ.
Включает:

1. Требования к структуре и функционированию системы (функциональные требования)
2. Требования к численности персонала и их квалификации по работе
3. Требования к надежности
4. Требования к безопасности
5. Требования к эргономике и технической эстетики

6. Требования к транспортабельности
7. Требования к эксплуатации и техническому обслуживанию
8. Требования к защите информации от несанкционированных действий
9. Требования к сохранности информации при авариях
10. Требования к защите от влияния внешних воздействий
11. Требования к патентной чистоте
12. Требования к стандартизации и унификации
13. Требования к видам обеспечения:
    a. Математическое
    b. Информационное
    c. Лингвистическое
    d. И др.

Состав и содержание работ по созданию системы (сейчас в ТЗ обычно не пишется. Это есть либо в тексте договора, либо в календарном плане).
Пример календарного плана:

Номер	Список работ	Сроки	Стоимость

Порядок контроля и приемки – что и как должно быть сделано, чтобы заказчик убедился, что система сделана. В ТЗ теперь это теперь тоже никогда не делается. Пишется в специальном документе в программе и методике испытания (ПМИ). Там записано, что необходимо сделать, чтобы систему развернуть, выполнить тест-кейсы,а также способ проверки, что она работает.
Пункты в ТЗ должны быть такими, чтобы можно было проверить.
Требования к составу и содержанию работ по вводу в действие:

1. Создание соответствующей инфраструктуры (!это важно!)
2. Приведение поступающей информации в структуру необходимой для системе (!это важно и необходимо прописать, это по сути формат данных!)
3. Кадровая подготовка

Требования к документированию - Объем и форма документации.
Источники разработки (!не очень понятно, формальный, о нем можно просто забыть!).
Общие слова:

- Требования должны быть понятными и проверяемыми.
- Заказчик будет всегда пытаться манипулировать недосказанными требованиями.
- По возможности писать короткими предложениями. Четко и коротко. Цель: избежание многозначности. Сюда же: не использовать многозначные термины. Избегать всяких оценочных слов.

(Конец части информации из записей лекций)

3. Процесс проектирования ИС и его основные компоненты

Проектирование - это поиск способа создания системы, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений.

(Добавлено из записей лекций)

Процесс проектирования ИС.

Этапы:

1. Постановка задачи;
2. Анализ и моделирование объекта автоматизированной системы;
3. Выявление и формулирование требований;
4. Концепция проектирования ИС (Общее виденье, не столь детальное описание как в архитектуре, расплывчатое описание, включает общие видение и некие сценарии того, как это будет жить);
5. Архитектура (Подробность зависит от целей использования, бывает разного уровня);
6. Выбор технологий (Этап выбора технологии может возникнуть в любом месте этого списка и в любое время);
7. Верификация и анализ принятых решений (Имеет смысл сделать верификацию на соответствие всех принятых решений между собой. Иногда имеет смысл сделать прототипирование узких мест);
8. Отчуждение результата (Например, составление ТЗ. Цель - создать документ или что-то самодостаточно описывающее все этапы, без необходимости пояснения от человека).

(Конец части информации из лекций)

Основу проекта любой ИС составляют следующие компоненты:

- методология проектирования;
- технологии проектирования;
- стандарты и методики проектирования;
- инструментальные средства проектирования (CASE-средства).

При этом взаимосвязь этих компонентов следующая: методология реализуется через конкретные технологии, каждая технология поддерживается соответствующими стандартами и методиками, а инструментальные средства обеспечивают выполнение процессов проектирования, описанных в методиках и стандартах.

Технология проектирования ИС – совокупность методологии и средств проектирования ИС, а также методов и средств организации проектирования.

Технология проектирования определяется как совокупность трех составляющих:

- пошаговой процедуры, определяющей последовательность технологических операций проектирования;
- критериев и правил, используемых для оценки результатов выполнения технологических операций;
- нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

К основным требованиям, предъявляемым к выбираемой технологии проектирования, относятся следующие:

- созданный проект должен отвечать требованиям заказчика;
- максимальное отражение всех этапов жизненного цикла проекта;
- обеспечение минимальных трудовых и стоимостных затрат на проектирование и сопровождение проекта;
- технология должна быть основой связи между проектированием и сопровождением проекта;
- рост производительности труда проектировщика;
- надежность процесса проектирования и эксплуатации проекта;
- простое ведение проектной документации

Реальное применение любой технологии проектирования, разработки и сопровождения ИС в конкретной организации и конкретном проекте невозможно без выработки ряда стандартов (правил, соглашений), которые должны соблюдаться всеми участниками проекта. К таким стандартам относятся следующие:

- стандарт проектирования;
- стандарт оформления проектной документации;
- стандарт пользовательского интерфейса.

Методология проектирования предполагает наличие некоторой концепции, принципов проектирования, реализуемых набором методов, которые, в свою очередь, должны поддерживаться некоторыми средствами.

Организация проектирования предполагает определение методов взаимодействия проектировщиков между собой и с заказчиком в процессе создания проекта ИС, которые могут также поддерживаться набором специфических средств.

CASE (англ. computer-aided software engineering) — набор инструментов и методов программной инженерии для проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее следующими основными характерными особенностями:

- 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).

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный ЖЦ ПО) содержит следующие компоненты;

- репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;
- графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;
- средства разработки приложений, включая языки 4GL и генераторы кодов;

- средства конфигурационного управления;
- средства документирования;
- средства тестирования;
- средства управления проектом;
- средства реинжиниринга.

4. Постановка задачи. Анализ и моделирование проектируемой системы. Выявление и формулировка требований.

Из лекций

Постановка задачи

С точки зрения заказчика:

1. От него исходит постановка задачи
2. Обязательно формулирует цель.
3. Формулирует критерии эффективности, т.е. как разрабатываемая систему улучшит его жизнь.
4. Видит место, где будет проходить автоматизация.
5. Имеет мысленное представление как ИС должна быть.

Анализ и моделирование проектируемой системы

Делается для того, чтобы лучше структурировать знания о системе.
Для этих целей используют либо текстовые документы, таблицы и диаграммы.
Включает в себя анализ процессов, так и структуру системы которую нужно переделать.
Делается для лучшего понимания для нас !мб исполнителя!, во-вторых для коммуникации с ними !мб с заказчиком!
Формализация позволяет выделить логические противоречия.
Нужна как общий способ коммуникации.

Выявление и формулировка требований

Требования –

- условия или возможность, которым должна соответствовать система.
- Совокупность некоторых утверждений относительно свойств или качеств разрабатываемой системы.

Принципиально, чтобы это было как-то документально зафиксировано в каком-то виде.

Свойства требований

1. Единичность. Требования должно касаться чего-то одного.
2. Завершенность. Требование должно быть самодостаточно.
3. Однозначность
4. Проверяемость /Измеримость
5. Обязательность
6. Актуальность

Типы требований:

1. Бизнес-требования
2. Функциональность
3. Нефункциональные требования
    a. Производительность
    b. Надежность

    c. Безопасность
    d. Дизайн и UI
    e. Технологии
    f. Железо
4. Требования к документации
5. Требования к условиям эксплуатации (в том числе к квалификации человека)

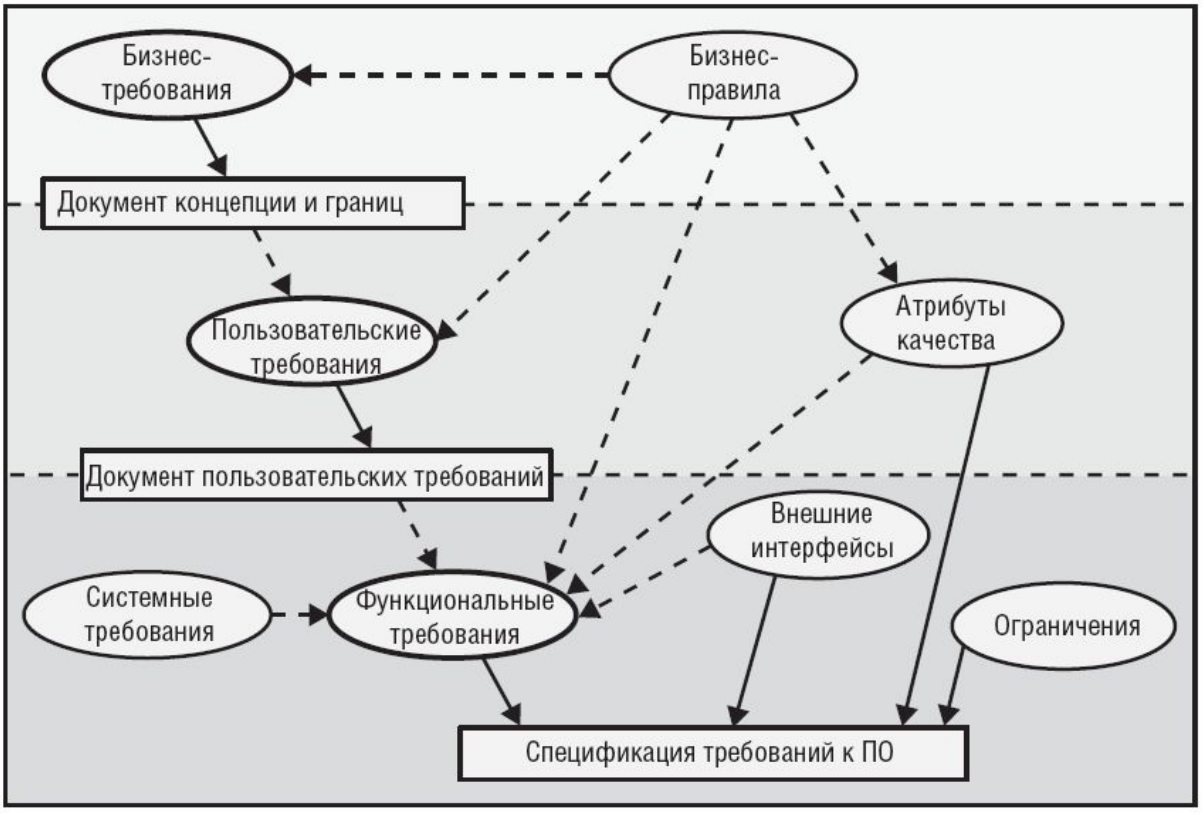
Источники требований:
1. Представления участников процесса
2. Документы и описания
3. Законодательство
4. Сложившаяся практика
5. Физические и инфраструктурные ограничения
6. Конкурирующие решения и продукты

Методы выявления требований:
1. Диалоги и интервью
2. Документация (ее анализ)
3. Наблюдение за пользователями
4. Семинары (продумывания выводимых и вытекающих требований, brainstorm)

Жизненный цикл требований:
1. Формулирование
2. Анализ
3. Проверка непротиворечивости с другими требования
4. Приоритезация (важно)
5. Закрытие требования
6. Переоткрытие требования

По Вигерсу:
Бизнес-требование Высокоуровневая бизнес-цель организации или заказчиков системы
Бизнес-правило Политика, предписание, стандарт или правило, определяющее или ограничивающее некоторые стороны бизнес-процессов. По своей сути это не требование к ПО, но оно служит источником нескольких типов требований к ПО
Ограничение Ограничение на выбор вариантов, доступных разработчику при проектировании и разработке продукта
Внешнее требование к интерфейсу Описание взаимодействия между ПО и пользователем, другой программной системой или устройством
Характеристика Одна или несколько логически связанных возможностей системы, которые представляют ценность для пользователя и описаны рядом функциональных требований
Функциональное требование
Описание требуемого поведения системы в определенных условиях
Нефункциональное требование
Описание свойства или особенности, которым должна

обладать система, или ограничение, которое должна соблюдать система
Атрибут качества Вид нефункционального требования, описывающего характеристику сервиса или производительности продукта
Системное требование
Требование верхнего уровня к продукту, состоящему из многих подсистем, которые могут представлять собой ПО или совокупность ПО и оборудования
Пользовательское требование
Задача, которую определенные классы пользователей должны иметь возможность выполнять в системе, или требуемый атрибут продукта



Прочее:
Под термином “моделирование” обычно понимают процесс создания точного описания системы; метод познания, состоящий в создании и исследовании моделей. Моделирование облегчает изучение объекта с целью его создания, дальнейшего преобразования и развития. Оно используется для исследования существующей системы, когда реальный эксперимент проводить нецелесообразно из-за значительных финансовых и трудовых затрат, а также при необходимости проведения анализа проектируемой системы, т.е. которая ещё физически не существует в данной организации.

Для формирования модели используются:
структурная схема объекта;
структурно-функциональная схема объекта;
алгоритмы функционирования системы;
схема расположения технических средств на объекте;

схема связи и др.

5. Формирование общей концепции ИС. Проектирование архитектуры. Архитектурные шаблоны

Формирование общей концепции ИС

Разработка концепции является рекомендуемой фазой создания автоматизированной системы, стоящей между фазами “принятие решения о рассмотрении/проработке идеи” и “оценка экономической эффективности внедрения”.

Концепция призвана:

- до определенного уровня структурировать идею автоматизации,
- учесть стратегические аспекты деятельности организации,
- определить, существует ли принципиальная возможность создания системы,
- сформулировать цели,
- определить автоматизируемые процессы и участников автоматизации,
- а также выявить стратегические риски проекта и способы их минимизации.

Кроме того, в рамках концепции делается первое предположение о целесообразности внедрения, которое в последующем будет проверено расчетами экономического обоснования.

Так как большая автоматизированная система охватывает более одной критической для дальнейшей деятельности группы бизнес-процессов, у такой системы будет фактически несколько заказчиков. Разработка концепции позволит идентифицировать этих заказчиков и учесть их требования, а также договориться им между собой.

Подход к разработке концепции включает несколько основных этапов.

1. Во-первых, необходимо сформировать рабочую группу и определить ответственного за подготовку документа. В состав рабочей группы необходимо включить как экспертов от бизнеса, так и ИТ-экспертов. Необходимо рассмотреть вопрос о включении в рабочую группу внешних консультантов.
2. Во-вторых, необходимо провести краткое обследование существующих бизнес-процессов и используемых ИТ-систем.
3. В-третьих, следует проанализировать опыт внедрения аналогичных систем в отрасли, стране и мире.
4. В-четвертых, организуется разработка концептуальной архитектуры системы: бизнес-процессы, приложения, данные, технологии. При этом необходимо учесть критические изменения, которые может повлечь внедрение системы на смежные бизнес-процессы, технологии и т. д. Также рассматриваются варианты построения непосредственно информационной системы и определяется наиболее эффективный вариант внедрения.
5. На основе исследования опыта других организаций и экспертного мнения участников рабочей группы проводится анализ ожидаемых эффектов внедрения.
6. В заключение составляется экспертное мнение рабочей группы о целесообразности внедрения, основанное на сопоставлении возможного эффекта и рисков внедрения.

Структура концепции

Ключевых блоки и темы, которые рекомендуется раскрыть в документе:

- Краткий анализ текущей ситуации: автоматизируемые бизнес-процессы, задачи по повышению эффективности данных процессов, участники этих процессов, действующие на момент обследования автоматизированные системы.
- Верхнеуровневое описание концептуальной модели системы, включая цели и задачи системы, основные требования к системе и ограничения, состав и структуру системы, подходы к интеграции с действующими системами и обеспечению безопасности.
- Обзор мирового опыта: результаты анализа аналогичных проектов, используемых для решения подобных задач программных продуктов, качественные и количественные результаты автоматизации.
- Подход к процессу планирования проекта включает описание верхнеуровневой этапности реализации проекта, организационного управления проектом, оценку необходимого ресурсного обеспечения и т. д.
- Ожидаемый эффект от создания автоматизированной системы. В данном разделе приводится оценка качественных и количественных эффектов от внедрения системы в организации. Оценка приводится на основании информации, собранной при анализе международного опыта, а также оценок рабочей группы. В этот раздел следует включить описание рисков внедрения и способов их минимизации.

Ключевые моменты создания концепции

- Анализ текущей ситуации не должен занимать более 50% времени и трудозатрат.
- Концепция должна быть именно концепцией, а не детальным планом с техническими и функциональными требованиями и архитектурой. Излишняя детализация потребует слишком много времени, а ситуация на момент внедрения может уже измениться, и все придется делать заново.
- Концепция должна быть увязана со стратегией организации и понятна руководству.
- Желательно рассматривать в концепции несколько вариантов построения будущей системы.
- Концепция не отвечает на вопрос об экономической эффективности внедрения системы. На данный вопрос отвечает технико-экономическое обоснование, которое готовится на основании концепции. В редких случаях ТЭО включается отдельным разделом в концепцию.

Трудности при разработке и утверждении концепции:

- Наличие нескольких противоречивых точек зрения на уровне руководства относительно развития компании, принципиальных моментов развития ИТ и других стратегических моментов.

  Решение: Необходимо добиваться формулирования бизнес- и ИТ-стратегии или, по крайней мере, единого видения по ключевым вопросам на уровне руководства организации как ключевого фактора успеха.
- Риск отсутствия спонсора и заказчика, обладающего полномочиями решать конфликтные ситуации.

  Решение: В этом случае необходимо обеспечить возможность обсуждения спорных ситуаций на самом высоком уровне компании.

- Сопротивление изменениям на уровне руководителей среднего и низшего звена.
  Решение: В таком случае чрезвычайно важно вовлекать заинтересованных руководителей высшего звена в разработку концепции мотивации и управления изменениями.
- Наличие негативного опыта работы с ИТ. Такой опыт вызовет первоначальную негативную реакцию со стороны заказчиков на любую новую идею или концепцию, исходящую со стороны ИТ-службы.
  Решение: Поэтому необходимо тщательно проанализировать причины прошлых неудач и заранее спланировать меры, которые будут предприняты, чтобы избежать повторения этого негативного опыта.

Проектирование архитектуры. Архитектурные шаблоны.
(Информация из записей лекций)

Архитектура: что является, что не является

- Много разрезов и много представлений.
- Различная степень детализации

Шаблоны проектирования содержат определенные атрибуты качества.
Архитектура способствует или препятствует атрибутам качества.
Удачная архитектура - шаблон для будущего использования

Принципы программной архитектуры:

1. Принцип декомпозиции
2. Обращение к элементу только через интерфейс
3. Аккуратное, разделение между подсистемами
4. Единый механизм обработки обратных веток(ветка алгоритма) (проработка ситуаций, когда что-то пошло не так) - например механизм исключений

(Конец части информации из записей лекций)

Архитектура программы или компьютерной системы – это структура или структуры системы, которые включают элементы программы, видимые извне свойства этих элементов и связи между ними.
Под архитектурой программных систем будем понимать совокупность решений относительно:

- организации программной системы;
- выбора структурных элементов, составляющих систему и их интерфейсов;
- поведения этих элементов во взаимодействии с другими элементами;
- объединение этих элементов в подсистемы;
- архитектурного стиля, определяющего логическую и физическую организацию системы: статические и динамические элементы, их интерфейсы и способы их объединения.

Для того чтобы построить правильную и надежную архитектуру и грамотно спроектировать интеграцию программных систем необходимо четко следовать современным стандартам в этих областях. Без этого велика вероятность создать архитектуру, которая неспособна развиваться и удовлетворять растущим потребностям пользователей ИТ. В качестве законодателей стандартов в этой области выступают такие международные организации как SEI (Software Engineering Institute), WWW (консорциум World Wide Web), OMG (Object Management Group),

организация разработчиков Java – JCP (Java Community Process), IEEE (Institute of Electrical and Electronics Engineers) и другие.
Рассмотрим классификацию программных систем по их архитектуре:

- Централизованная архитектура;
- Архитектура "файл-сервер";
- Двухзвенная архитектура "клиент-сервер";
- Многозвенная архитектура "клиент-сервер";
- Архитектура распределенных систем;
- Архитектура Веб-приложений;
- Сервис-ориентированная архитектура.

Централизованная архитектура

Централизованная архитектура вычислительных систем была распространена в 70-х и 80-х годах и реализовывалась на базе мейнфреймов, либо на базе мини-ЭВМ. Характерная особенность такой архитектуры – полная "неинтеллектуальность" терминалов. Их работой управляет хост-ЭВМ.
Достоинства такой архитектуры:

- пользователи совместно используют дорогие ресурсы ЭВМ и дорогие периферийные устройства;
- централизация ресурсов и оборудования облегчает обслуживание и эксплуатацию вычислительной системы;
- отсутствует необходимость администрирования рабочих мест пользователей;

Архитектура "файл-сервер"

Файл-серверные приложения – приложения, схожие по своей структуре с локальными приложениями и использующие сетевой ресурс для хранения программы и данных.

- Функции сервера: хранения данных и кода программы.
- Функции клиента: обработка данных происходит исключительно на стороне клиента.

Достоинства такой архитектуры:

- многопользовательский режим работы с данными;
- удобство централизованного управления доступом;
- низкая стоимость разработки;
- высокая скорость разработки;
- невысокая стоимость обновления и изменения ПО.

Недостатки:

- проблемы многопользовательской работы с данными: последовательный доступ, отсутствие гарантии целостности;
- низкая производительность (зависит от производительности сети, сервера, клиента);
- плохая возможность подключения новых клиентов;
- ненадежность системы.

Архитектура "клиент-сервер"

Клиент-сервер (Client-server) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами.
Преимуществами данной архитектуры являются:

- возможность, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети;
- все данные хранятся на сервере, который, как правило, защищен гораздо лучше большинства клиентов, а также на сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа;
- поддержка многопользовательской работы;
- гарантия целостности данных.

Недостатки:

- неработоспособность сервера может сделать неработоспособной всю вычислительную сеть;
- администрирование данной системы требует квалифицированного профессионала;
- высокая стоимость оборудования;
- бизнес логика приложений осталась в клиентском ПО.

Многоуровневый "клиент-сервер"

Многоуровневая архитектура клиент-сервер (Multi-tier architecture) – разновидность архитектуры клиент-сервер, в которой функция обработки данных вынесена на один или несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов.

Среди многоуровневой архитектуры клиент-сервер наиболее распространена трехуровневая архитектура, предполагающая наличие следующих компонентов приложения: клиентское приложение (обычно говорят "тонкий клиент" или терминал), подключенное к серверу приложений, который в свою очередь подключен к серверу базы данных.

Плюсами данной архитектуры являются:

- клиентское ПО не нуждается в администрировании;
- масштабируемость;
- конфигурируемость – изолированность уровней друг от друга позволяет быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;
- высокая безопасность;
- высокая надежность;
- низкие требования к скорости канала (сети) между терминалами и сервером приложений;
- низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости.

Минусы:

- растет сложность серверной части и, как следствие, затраты на администрирование и обслуживание;
- более высокая сложность создания приложений;
- сложнее в разворачивании и администрировании;
- высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;

- высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

Архитектура распределенных систем

Такой тип систем является более сложным с точки зрения организации системы. Суть распределенной системы заключается в том, чтобы хранить локальные копии важных данных.

Архитектура Веб-приложений

Обычно Веб-приложения создаются как приложения в архитектуре "клиент-сервер", но серверная часть имеет различные архитектурные решения

Можно выделить основные особенности веб-архитектуры:

- отсутствие необходимости использовать дополнительное ПО на стороне клиента – это позволяет автоматически реализовать клиентскую часть на всех платформах;
- возможность подключения практически неограниченного количества клиентов;
- благодаря единственному месту хранения данных и наличия системы управления базами данных обеспечиваются минимальные требования для поддержания целостности данных;
- доступность при работоспособности сервера и каналов связи;
- недоступность при отсутствии работоспособности сервера или каналов связи;
- достаточно низкая скорость Веб сервера и каналов передачи данных;
- относительно объема данных – архитектура Веб систем не имеет существенных ограничений.

Сервис-ориентированная архитектура

Решение многих описанных выше задач, возникающих при создании современных Веб-приложений, теперь начинает возлагаться на Веб-сервисы – не зависящие от платформы, объектной модели и клиента программные компоненты, которые можно вызывать из клиентских Веб-приложений (а также из самих Веб-сервисов) через основанный на протоколе HTTP и языке XML протокол SOAP. Для описания Веб-сервисов используется XML-подобный язык WSDL, а для организации реестров Веб-сервисов, в которых разработчики и компании могут искать необходимые им сервисы, а также публиковать данные о своих сервисах – интерфейс UDDI.

Основными целями применения SOA для крупных информационных систем, уровня предприятия, и выше являются:

- сокращение издержек при разработке приложений, за счет упорядочивания процесса разработки;
- расширение повторного использования кода;
- независимость от используемых платформ, инструментов, языков разработки;
- повышение масштабируемости создаваемых систем;
- улучшение управляемости создаваемых систем.

Принципы SOA:

- архитектура, как таковая, не привязана к какой-то определенной технологии;
- независимость организации системы от используемой вычислительной платформы (платформ);
- независимость организации системы от применяемых языков программирования;

- использование сервисов, независимых от конкретных приложений, с единообразными интерфейсами доступа к ним;
- организация сервисов как слабосвязанных компонентов для построения систем.

6. Атрибуты качества. Связь выбранной архитектуры с атрибутами качества. Тактики реализации атрибутов качества. Примеры.

(Информация из записей лекций)
Атрибут качества - обобщённые требования, которые применимы к любой системе - проекция содержательных требований.
(Конец информации из записей лекций)
Атрибуты качества делятся на три класса:

1. Атрибуты качества системы. Из них мы рассмотрим:
    - производительность,
    - надежность,
    - безопасность,
    - готовность, (устойчивость, возможность при критической ситуации продолжить работу или восстановиться),
    - юзабилити (практичность),
    - модифицируемость,
    - тестируемость (контролепригодность).
2. Коммерческие атрибуты качества (например, срок вывода продукта на рынок, коммерческая отдача), реализация которых обусловливается архитектурой.
3. Атрибуты качества самой архитектуры (например, концептуальная целостность), которые косвенно влияют на другие качества — например, модифицируемость.

(Информация из записей лекций)
Для описания конкретных атрибутов качества используются сценарии атрибутов качества (что-то типа use-case).
(Конец информации из записей лекций)
(Информация из литературы)

1. Архитектура определяет возможность реализации предполагаемых атрибутов качества системы; проектирование и оценку этих атрибутов следует проводить на архитектурном уровне.
2. Сама по себе архитектура не реализует никаких атрибутов качества, но она образует основу для достижения качества - впрочем, если не уделять должного внимания к деталям, пользы от этой основы не будет никакой.

(Конец информации из литературы)
Сценарием атрибута качества называется требование, путем выполнения которого этот атрибут реализуется. Сценарий состоит из шести элементов.

1. Источник стимула. Это некий субъект (человек, вычислительная система или любой другой), который порождает стимул.
2. Воздействие (стимул). Стимулом называется наблюдаемое в системе явление, требующее к себе внимания. (например, грязные внешние данные, не предусмотренный ввод)
3. Условия. Стимул возникает в определенных условиях. К примеру, система может находиться в состоянии перегрузки или исполняться в обычном режиме.

4. Артефакт. Объектом воздействия стимула является некий артефакт. В этом качестве может выступать как система в целом, так и ее отдельные элементы.
5. Реакция. Реакция — это действие, предпринятое в ответ на появление стимула.
6. Количественная мера реакции. Предпринимаемые в ответ на стимул действия должны быть измеримы — только в этом случае соответствие требованию можно проверить.

Пример сценария модифицируемости: «В период проектирования разработчику требуется внести изменения в код пользовательского интерфейса, чтобы сделать фоновый цвет синим. Соответствующие операции и тестирование проводились в течение трех часов и не привели к появлению побочных эффектов в поведении».

(Информация из литературы)

Тактика - это проектное решение, которое влияет на управление реакцией по атрибуту качества. Совокупность тактик называется архитектурной стратегией.

Тактики готовности

Тактики обнаружения неисправностей:

- Ping/Echo-пакеты
- Heartbeat
- Исключения

Тактики восстановления после неисправностей:

- Голосование - все компоненты выполняют некоторую операцию и сравнивают результаты, компоненты с отклонениями отключаются.
- Активное резервирование - резервные компоненты выполняют такую же работу, что и главный, принимается первый полученный результат.
- Пассивное резервирование - главный компонент обновляет состояние резервных после каждой операции, при выходе из строя, включается резервный компонент.
- Резерв - постоянное сохранение актуального состояния на отказоустойчивый узел. При выходе из строя, компонент перезагружается и инициализируется актуальным состоянием.

Тактики восстановления с повторным введением компонентов:

- Затенение - после отказа, компонент переходит в “теневой режим”, синхронизируется с работающими компонентами, и снова вводится в действие.
- Повторная синхронизация состояния - при пассивном или активном резервировании, восстанавливаемый компонент обновляет свое состояние перед вводом в действие.
- Использование контрольных точек/ Откат.

Тактики предотвращения неисправностей:

- Снятие с эксплуатации - отключение компонента для предотвращения прогнозируемых отказов (например, утечки памяти).
- Транзакционность действий.
- Использование диспетчера процессов для перезагрузки неисправных компонентов.

Тактики модифицируемости

Локализация изменений - изменения воздействуют на меньшее количество модулей.

Предотвращение волнового эффекта - для достижения результата требуется минимальное количество изменений. Волновой эффект - каскадное возрастание необходимых изменений.
Локализация изменений:

- Обеспечение семантической связности. Увеличение связности (взаимосвязи внутри модуля) и уменьшение сцепления (связей между модулями).
- Прогнозирование ожидаемых изменений. Уменьшение последствий изменений.
- Обобщение модуля. Чем более общий характер носит модуль, тем шире диапазон функций, рассчитываемых из одних входных данных.
- Ограничение возможных альтернатив. Чем меньше альтернатив, тем меньше последствия модификаций.

Предотвращение волнового эффекта:

- Информационная закрытость. Разделение информации на приватную и публичную.
- Обслуживание существующих интерфейсов. Отделение интерфейса от реализации:
    - Введение новых интерфейсов.
    - Введение нового адаптера.
    - Введение заглушки модуля.
- Ограничение каналов связи. Уменьшение зависимостей между модулями.
- Введение посредника.

Откладывание связывания:

- Регистрация в период прогона. По типу Plug-and-Play.
- Использование конфигурационных файлов.
- Полиморфизм, отложенное связывание вызова методов.
- Замена компонентов. (Например, подмена dll)
- Использование стандартизированных протоколов для связывания независимых процессов.

Тактики производительности

Производительность - время ответа на запрос.
Продолжительность блокирования - время использования (блокирования) ресурсов. Делится на следующие компоненты:

- Состязание за ресурсы (Race condition).
- Готовность ресурсов. В системе может не оказаться необходимого количества свободных ресурсов.
- Ожидание результатов других вычислений.

Потребление ресурсов:

- Повышение вычислительной эффективности. Например, ускорение алгоритмов, уменьшение времени доступа к памяти.
- Сокращение издержек вычислений. Уменьшение накладных расходов, отказ от тяжелых операций.
- Уменьшение частоты поступления событий.
- Уменьшение количества опросов. Если уменьшить частоту опроса не получается, можно уменьшить количество опросов от различных модулей.
- Ограничение времени исполнения. Например, ограничение количества итераций.

- Ограничение длины очереди. Уменьшение событий, ожидающих обработки в очереди и, соответственно, выделенных на них ресурсов.

Управление ресурсами:

- Введение параллелизма.
- Создание копий часто используемых данных, кэширование.
- Увеличение количества ресурсов - апгрейд машины.

Арбитраж ресурсов - стратегии планирования распределения ресурсов.

Арбитраж ресурсов:

- FIFO.
- Планирование с фиксированным приоритетом:
    - По характеристикам порождающего процесса.
    - По предельным срокам обработки.
    - По частоте вызова.
- Динамическое приоритетное планирование:
    - Циклическое обслуживание (Round Robin).
    - Приоритет коротких предельных сроков.
- Статическое планирование (???). Стратегия циклического планирования предполагает неоперативное определение моментов прерывания и порядка распределения ресурсов между запросами.

Тактики безопасности

Противодействие атакам:

- Аутентификация пользователей. Пароли, сертификаты, биометрические данные.
- Авторизация пользователей. Проверка прав доступа пользователя.
- Обеспечение конфиденциальности данных. Шифрование, VPN, SSL.
- Обеспечение целостности. Проверка неизменности данных, хэширование, контрольные суммы.
- Минимизация подверженности внешним воздействиям. Распределение служб между разными хостами.
- Ограничение доступа. Firewall, DMZ - демилитаризованная зона, сеть-прослойка между внешней сетью и внутренней.

Обнаружение атак:

- Использование систем обнаружения вторжений. Сравнение запросов с сигнатурами атак.

Восстановление после атак:

- Восстановление предшествующего состояния (такие же как и в готовности):
    - Резервирование систем.
    - Сохранение “удаленных” данных в БД.
- Идентификация исполнителя атаки:
    - Ведение журнала действий.

Тактики контролепригодности (Тестирования)

Сбор входных/выходных данных:

- Запись/считывание. Сохранение данных используемых при нормальном исполнении программы для тестирования в будущем.
- Отделение интерфейса от реализации.

- Специализация путей/интерфейсов доступа. Создание объектов-интерфейсов для тестирования, mock-объектов.

Внутренний мониторинг:

- Использование средств мониторинга параметров, загрузки ресурсов. Встраивание таких средств может быть постоянным или временным. Также увеличивает необходимое количество тестов, так как при выключении требуется снова проводить тестирование.

Тактики практичности (Usability)

Тактики периода исполнения:

- Обслуживание модели задачи. Предложение подсказок и автоматические исправления в зависимости от контекста.
- Обслуживание модели пользователя. Уменьшение времени задержки до допустимого пользователем, подстраивание темпа прокрутки страниц под скорость чтения пользователя.
- Обслуживание модели системы. Прогнозирование времени выполнения операций.

Тактики периода проектирования:

- Отделение пользовательского интерфейса от остальных элементов приложения.

(Конец информации из литературы)

(Информация из записей лекций)

Тактики Usability

1. Сортировка по частоте обращения – путь доступа обратно пропорционален частоте обращения.
2. Локальность видения
3. Чтобы люди не бесились при задержках – интуитивно быстрые операции должны выполняться быстро. Если не возможно, система должна как-то информировать о своих действиях
4. Возможность отмены операции и действий пользователя
5. Желательно закладывать в систему возможности АБ-тестирования (когда пользователю дают 2 интерфейса, где он чаще задерживается дольше, там ему дают дополнительный вес)

(Конец информации из записей лекций)

(Информация из литературы)

Любой образец (паттерн) реализует сразу несколько тактик, которые зачастую даже относятся к разным атрибутам качества. При реализации такого образца также принимаются решения о реализации тех или иных тактик. Например, реализация может предусматривать ведение журнала запросов к активному объекту в расчете на возможное восстановление, ведение контрольного журнала или обеспечивать контролепригодность (тестируемость).

Для проведения анализа архитектор должен иметь представление обо всех встроенных в реализацию тактиках, а на этапе проектирования от него требуется принятие решений о том, какие тактики смогут наилучшим образом выполнять поставленные перед системой задачи.

(Конец информации из литературы)

7. Принципы и критерии выбора технологий для проекта. Высокоуровневые и низкоуровневые технологии. Связь с параметрами проекта и внешними требованиями к системе

(Аня использовала свои конспекты, сверила с ребятами)

Критерии выбора технологий:

1) Те, что вы можете использовать (предполагается знание технологии, я смогу её использовать, наличие ресурсов у вашей команды)
2) В зависимости от масштаба проекта
3) Требуемая скорость разработки, соотношение приоритетов скорость/эффективность/качество
4) Стоимость нужных специалистов (для заказчика), доступность специалистов
5) (иногда) функциональные требования производительность, надежность, безопасность, управление памятью…
6) Стабильность и надежность выбираемого фреймворка, предпочтение устоявшимся технологиям; или наоборот, стремление к новому и инновационному
7) Наличие community, инструментальной среды, хорошей документации
8) Набор выбранных технологий коррелируется с типом проекта (его сложностью, спецификой бизнеса).
9) Дополнительно: наличие готовых решений; гибкость решения; стоимость поддержки; кроссплатформенность; возможность интеграции с другими решениями; ***тренды.***

Типы технологий:

1. **Нативное приложение, на «чистом» языке** – целесообразно, если у вас сильно нагруженные и большие системы, есть строгие требования к производительности; системы реального времени; что-то очень оригинальное и инновационное; система «плохо ложится» на традиционные средства;
2. **Фреймворк, платформа** (e.g. Xamarin) – необходимо повысить эффективность разработки, разработать быстрее и легче, снизить стоимость и временные затраты; производительность не так критична и можно работать в рамках ограничений фреймворка; не страшат проблемы с лицензированием и поддержкой
3. **Высокоуровневые средства/конструкторы** (здесь больше настройки, чем разработки) - примеры: CMS (система управления контентом с шаблонами) вроде WordPress, Joomlam OpenCart, Drupal; **Content Management Framework;** Целесообразно, если у вас проект с определенной и известной тематикой (допустим, интернет-магазин) - скорее всего, существует уже готовое решение или система управления, позволяющая создать проект быстро и дешево. Раньше простые сайты, для которых и создавались CMS, писались на PHP, поэтому сейчас таких большинство. Сегодня CMS на других языках развиваются слабо, потому что уже есть сильные конкуренты на PHP, а для простого сайта язык не играет большой роли, поэтому все смотрят на возможности этих готовых продуктов.

Главные ограничения на выбор: время и бюджет

(Добавлено из интернета)
Важно! Оноссовский явно использовал вот эту статью (слишком уж знакомый текст), рекомендуется к прочтению:
https://habrahabr.ru/company/SECL_GROUP/blog/315734/
Связь типов проектов и типов технологий:
По сложности проекты делятся:

1. Простые (визитки, лендинги, простые интернет-магазины, простые приложения) — такие решения обычно делаются на тематических коробочных решениях, CMS или шаблонах.
2. Средние (сложные интернет-магазины и маркетплейсы, порталы национального масштаба, разнообразные сервисы, продвинутые приложения) — такие решения обычно делаются на фреймворках.
3. Сложные (огромные порталы, социальные сети, инновационные и нетиповые решения) — ядро таких проектов обычно разрабатывается на чистом (нативном) языке программирования.

По тематике: интернет-магазины, доски объявлений, социальные сети и т.д. Для большинства популярных тематических решений уже давно есть коробочные продукты!

Чем больше проект, тем

- выше требования по гибкости, нагрузкам и безопасности.
- бюджет играет всё меньше значения по сравнению с эффективностью.
- следовательно, будет проще писать все с нуля, выделяя на это лучших специалистов, чем если брать какие-то готовые решения, которые непонятно кем писались и непонятно какие проблемы в них скрыты
- тем больше стек технологий, который в нем используется; части могут работать на разных серверах, с разными доменами (поддоменами) и разными технологиями.

Подробнее о высокоуровневых (в статье только о CMS и CMF)
CMS: готовое программное обеспечение, которое нужно только настроить, реже — дописать / переписать какую-то из частей.
CMF, если говорить простым языком, — это что-то среднее между CMS и фреймворком по возможностям. Обычно CMF используют для самых сложных сайтов из этой категории. Этот подход позволяет избавиться от лишних частей CMS, которые не нужны конкретному проекту.

CMS бывают разные по назначению: общие, для интернет-магазинов, для блогов и т.д. Разные по условиям использования: платные и бесплатные. Для каждой популярной CMS есть много разных платных и бесплатных модулей, которые легко подключать и использовать.
Минусы:

- Больше всего непонимания среди конечных заказчиков таких решений. Любая CMS — это тонны готового программного кода, десятки и сотни модулей. Все это очень сильно ограничивает специалистов.
- Такие решения сильно «тормозят», они абсолютно не гибкие,

- Небезопасно: Их очень легко взломать, особенно бесплатные CMS. Еще часто взламывают CMS через модули сторонних разработчиков, в которых есть критические уязвимости, потому что мы никогда не знаем, какого уровня программист писал тот или иной модуль.
- НЕ рассчитана для большого и сложного сайта.
- Она не может выдержать большие нагрузки.

Шаблоны

Это еще на одну ступеньку выше, чем CMS. Если CMS — это конструктор, и его нужно настраивать, то шаблоны — это уже готовые решения под типовые случаи. Например, в каждом городе есть свои рестораны, такси, клиники и т.д. Для всех этих типов малого бизнеса нужно примерно одно и тоже. Поэтому, можно просто выбрать готовый тематический шаблон, заменить в нем логотип, цвета и контент. При желании такие шаблоны можно дорабатывать по усмотрению владельца.

+очень дешевые
+можно запускать моментально.
- не учитывают особенностей бизнеса и конверсия будет не очень высокой.

Есть специальные каталоги шаблонов: TemplateMonster, ThemeForest и др. Часто встречаются онлайн-конструкторы, в том числе тематические: Wix, PageCloud и др.

Тренды:

Языки сильно меняются каждые 5-7 лет, фреймворки — каждые 2-3 года, а CMS — каждые 1-2 года. Важно выбрать не просто хорошую технологию сегодня, а предугадать тренды развития так, чтобы не пришлось переписывать проект.

8. Методологии проектирования ИС и их связь с методологиями разработки

Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде жизненного цикла (ЖЦ) ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов. Для каждого этапа определяются состав и последовательность выполняемых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т.д.

Внедрение методологии должно приводить к снижению сложности процесса создания ИС за счет полного и точного описания этого процесса, а также применения современных методов и технологий создания ИС на всем жизненном цикле ИС - от замысла до реализации.

(Материал из лекций)

Два типа расчета оплаты: Fix price и Time & Material.

Методологии:

Waterfall (каскадная модель)

Этапы:

1. Проектирование - проектирование на уровне требований, архитектуры
2. Дизайн (детальное техническое проектирование, проектирование на уровне требований, архитектуры) - до уровня “садимся и делаем”
3. Разработка
4. Тестирование
5. Внедрение
6. Поддержка и сопровождение

Преимущества:

- Хорошо для Fix price
- Лучше делать небольшие проект: меньше шансов, что то-то изменится
- Большие проекты - для госзаказов, где строгие регламенты
- Отличное документирование
- Фиксированные ресурсы
- Легко управлять

Недостатки:

- Для среднего уже плохо, много доп. штук
- Гибкости нет

Когда можно использовать:

- Все требования зафиксированы
- Есть необходимые ресурсы
- Лучше бы в небольших проектах

V - методология validation and Verification

Этапы:

1. Бизнес требования
2. Функциональные требования (верификация проектных материалов)
3. Архитектура
4. Реализация компонент (unit-тестирование)

5. Интеграция компонент (интеграционное тестирование)
6. Функциональное тестирование (функциональное тестирование)
7. Приёмо-сдаточные испытания (окончательное тестирования)

Смысл: тестируется всё что можно. Тестовая команда участвует с самого начала. Тесты-тесты-тесты…

Когда используется:

- Когда цена ошибок очень высокая
- Четкие фиксированные требования
- Приоритетом является надежность и качество
- Когда доступны необходимые тестовые ресурсы.

Общее с каскадной моделью:

- Предсказуем по срокам
- Предсказуем по стоимости
- Фиксированные требования
- Проектирование выполняется один раз в начале и после верификации больше не меняется.
- => нет гибкости

Инкрементная модель

Идея: разрабатываем ядро (базовая функциональность) по каскадной модели, дальше начинает обрастать дополнительными фрагментами. Каждый дополнительный фрагмент разрабатывается по каскадной модели.

Этапы:

1. Планирование
2. Проектирование
3. Разработка
4. Тестирование
5. Оценка
6. Развертывание
7. Переходим на пункт 1.

Особенности:

- У нас есть несколько версий системы. Каждая версия - законченная система, но с расширенным функционалом.
- Каждый фрагмент ограничен по функциям и не предполагает доработок.
- Выше гибкость
- Меньше предсказуемость по цене и ресурсам

Применяется:

- Когда базовые требования чёткие и понятные
- Когда нужны выложить первую версию максимально быстро.
- Когда какой-либо фрагмент имеет высокие риски по времени.

RAD - rapid application development

Что это?

1. Есть система, которую нужно быстро собрать и выпустить
2. Берём конструктор, по описанию БП или моделей данных конструируется система. По uml, bpmn, entity relation.

Процесс:

1. Первоначальный анализ и быстрое проектирование

2. Бизнес моделирование
3. Моделирование данных
4. Моделирование процесса
5. Генерация и сборка приложения
6. Тестирование
7. Этот процесс может быть итеративным.

Особенности:

- Нет отдельного этапа проектирования
- Проектирования и разработка сливаются вместе.

Когда применимо:

- Есть такая система гибкой разработки (либо сами разработали, либо купили) и есть специалисты
- Точное и уверенное знание бизнеса, к которому разрабатывается (понимание задачи)
- Очень жесткие временные рамки, либо используется как средство прототипирования.

Agile

Этапы:

1. Старт
2. Планирование и проектирование
3. Разработка
4. Тестирование
5. Демонстрация
6. Перейти на пункт 2

Достоинства:

- Гибкость
- Есть результат на выходе каждой итерации

Недостатки:

- Не фиксированная цена
- Сложно оценить трудозатраты и стоимость
- Плохо применима к существенно сложны проектам (например, компилятор или ось)

Применяется:

- В крупных проектах, которые должны очень быстро адаптироваться
- При сильно изменчивых требованиях
- При слабой связности системы

Кусочное проектирование снизу-вверх, но у кого-то должна быть вся картина, хоть она нигде и не зафиксирована

Итеративная модель

Особенности:

- Похоже на инкрементную модель
- Есть базовая функциональность, реализуется каскадом, но результат реализации не является законченным продуктом. (Например, проработан протокол, но не компонентов, которые с ним работают)
- При правильном проектировании достаточно быстро появляется базовый функционал, а дальше нарастает.

Где используется:

- В больших и очень больших проектах.
- Конечные требования достаточно определенные понятные и не изменяются.
- Основная задача определена, но детали могут меняться

Спиральная модель

Особенности:

- Похожа на итеративную модель
- Акцент делается на анализе рисков на каждой итерации

Этапы:

1. Планирование,
2. Проектирование,
3. Формулировка требований
4. Анализ рисков
    - Относится к системам, которые плотно вписаны в бизнес процесс. (Например, платежная система в банке)
    - Необходимо проверить, что не появится дыр, не просядет производительность, не будет конфликтов.
5. Конструирование
6. Оценка результата
7. Идём на пункт 1

Проектирование и анализ рисков занимает намного больше времени чем конструирование

Где применяется:

- Сложные и дорогие проекты

(Конец материала из лекций)

9. Методологии проектирования ИС на основе CASE-технологий. Методологии SADT, DFD, ERD. Стандарт DATARUN.

(Материал из лекций)

При увеличении сложности систем возникла проблема снижения рисков программирования, а также снижения порога входа в программирование. Результатом стало появление и развитие множества стандартов диаграмм и блок-схем.

Графические языки программирования позволяют писать какую-то логику, которая будет компилироваться в исполняемый код. Но, например, при обвале ошибки ошибка произойдет на уровне исполняемого кода, а не графического, и интерпретировать ее на более высокий уровень не представляется возможным.

Структурный подход к проектированию информационных систем – иерархическое разбиение большой системы на компоненты. (Типичное проектирование сверху вниз)

Принципы иерархической декомпозиции:

1. Разбиение сложной проблемы на набор мелких независимых задач
2. Иерархическое упорядочивание (Представление в виде дерева)
3. Принцип абстрагирования (от менее существенного, что раскрывается на более глубоком уровне)
4. Принцип формализации – строгий подход к решению проблем, следование единым принципам решения проблем
5. Принцип непротиворечивости (элементы разбиения не должны противоречить друг другу)
6. Структурирование данных

(Конец материала из лекций)

Под **CASE-средством** понимается специальное программное обеспечение, поддерживающее процессы создания и сопровождения информационных систем.

Технология проектирования определяется как совокупность трех составляющих:

- пошаговой процедуры, определяющей последовательность технологических операций проектирования;
- критериев и правил, используемых для оценки результатов выполнения технологических операций;
- нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

(Материал из лекций)

Поговорим о конкретных case-системах. Все делают акцент на этапе анализа и проектирования. Генерировать визуальное представление системы, подготовка проектной документации.

CASE-систему состоит из следующих компонентов:

1. Репозиторий
2. Графические средства анализа и проектирования (DFD, UML и т.д.)
3. RAD – Средства быстрой разработки приложений (Генераторы кода из диаграмм, LGL языки и т.д.) Создание условных прототипов, но очень не эффективны и ограничены
4. Средства конфигурационного контроля (версионирование)
5. Средство документирования

6. Средства управления проектом
7. Средства реинжиниринга – прежде всего, относится к базам данных, восстановление метаданных, в меньшей мере к реальным программам из-за сложности
8. Средства анализа (upperCase, Design/IDEF, BPWIN и т.д.)
9. Средства анализа и проектирования (Oracle designer, Sybase Power Designer)
10. Средства проектирования баз данных (Oracle database designer и т.д.)
11. Средства разработки приложения (Oracle developer, Rational Rows, Silver run и т.д.)

(Конец материала из лекций)

Методология SADT.

Процесс моделирования в SADT включает

1. сбор информации об исследуемой области,
2. документирование полученной информации,
3. представление ее в виде модели
4. уточнение модели.

Этот процесс подсказывает путь выполнения согласованной и достоверной структурной декомпозиции. SADT способна обеспечить как графический язык, так и процесс создания непротиворечивой и полезной системы описаний.

В терминах IDEF0 система представляется в виде комбинации блоков и дуг.

- Блоки представляют функции системы, дуги представляют физические объекты, информацию или действия, которые образуют связи между функциональными блоками.
- Взаимодействие работ с внешним миром и между собой описывается в виде стрелок или дуг.
- С дугами связываются метки на естественном языке, описывающие данные, которые они представляют.
- Дуги показывают, как функции системы связаны между собой, как они обмениваются данными и осуществляют управление друг другом.
- Дуги могут разветвляться и соединяться. Ветвление означает множественность (идентичные копии одного объекта) или расщепление (различные части одного объекта). Соединение означает объединение или слияние объектов.

Каждый блок IDEF0-диаграммы может быть представлен несколькими блоками, соединенными интерфейсными дугами, на диаграмме следующего уровня. Эти блоки представляют подфункции (подмодули) исходной функции. Каждый из подмодулей может быть декомпозирован аналогичным образом. Число уровней не ограничивается, зато рекомендуется на одной диаграмме использовать 3-7 блоков.

Пять типов стрелок допускаются в диаграммах:

1. Вход (Input) - материал или информация, которые используются работой для получения результата (стрелка, входящая в левую грань).
2. Управление (Control) - правила, стратегии, стандарты, которыми руководствуется работа (стрелка, входящая в верхнюю грань). В отличии от входной информации управление не подлежит изменению.
3. Выход (Output) - материал или информация, которые производятся работой (стрелка, исходящая из правой грани). Каждая работа должна иметь хотя бы

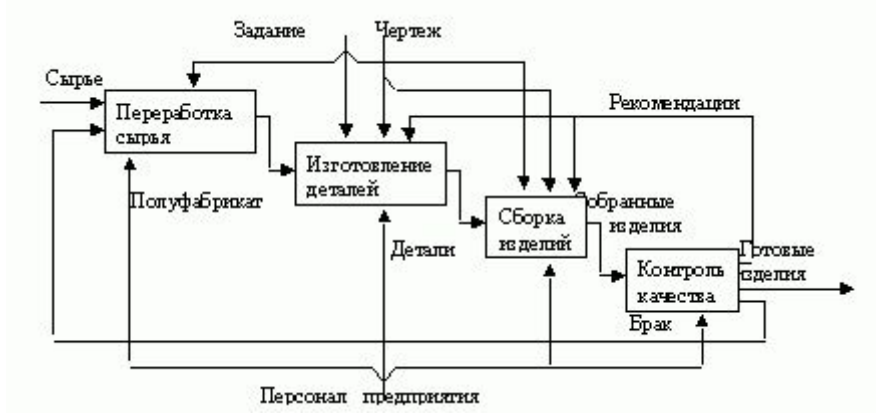
одну стрелку выхода, т.к. работа без результата не имеет смысла и не должна моделироваться.

4. Механизм (Mechanism) - ресурсы, которые выполняют работу (персонал, станки, устройства - стрелка, входящая в нижнюю грань).
5. Вызов (Call) - стрелка, указывающая на другую модель работы (стрелка, исходящая из нижней грани).

Различают в IDEF0 пять типов связей работ.

1. Связь по входу (input-output), когда выход вышестоящей работы направляется на вход следующей работы.
2. Связь по управлению (output-control), когда выход вышестоящей работы направляется на управление следующей работы. Связь показывает доминирование вышестоящей работы.
3. Обратная связь по входу (output-input feedback), когда выход нижестоящей работы направляется на вход вышестоящей. Используется для описания циклов.
4. Обратная связь по управлению (output-control feedback), когда выход нижестоящей работы направляется на управление вышестоящей. Является показателем эффективности бизнес-процесса.
5. Связь выход-механизм (output-mechanism), когда выход одной работы направляется на механизм другой и показывает, что работа подготавливает ресурсы для проведения другой работы.

Из перечисленных блоков, как из отдельных кирпичиков, строится диаграмма. Пример SADT-диаграммы:



Диаграммы потоков данных используются для описания движения документов и обработки информации как дополнение к IDEF0.

Созданные модели потоков Данных организации могут быть использованы при решении таких задач, как:

1. определение существующих хранилищ данных;
2. определение и анализ данных, необходимых для выполнения каждой функции процесса;
3. подготовка к созданию модели структуры данных организации, так называемая ERD-модель (IDEF1X);
4. выделение основных и вспомогательных бизнес-процессов организации.

DFD - это граф, на котором показано движение значений данных от их источников через преобразующие их процессы к их потребителям в других объектах.

DFD содержит процессы, которые преобразуют данные, потоки данных, которые переносят данные, активные объекты, которые производят и потребляют данные, и хранилища данных, которые пассивно хранят данные.

Процессы. Процесс преобразует значения данных. Могут быть с побочными эффектами и без. На DFD процесс изображается в виде эллипса, внутри которого помещается имя процесса; каждый процесс имеет фиксированное число входных и выходных данных, изображаемых стрелками.

Потоки данных. Поток данных соединяет выход объекта (или процесса) с входом другого объекта (или процесса). Дуги могут разветвляться или сливаться, что означает, соответственно, разделение потока данных на части, либо слияние объектов.

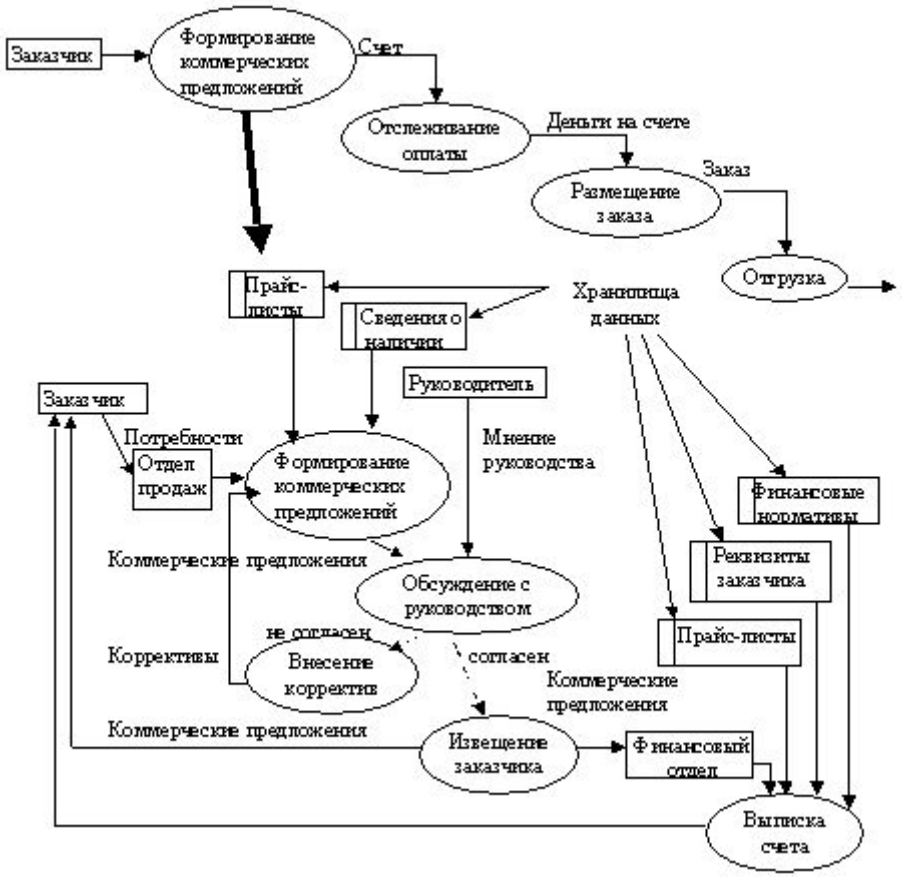
Активные объекты. Активным называется объект, который обеспечивает движение данных, поставляя или потребляя их. Активные объекты обычно бывают присоединены к входам и выходам DFD. (обозначаются прямоугольниками с заштрихованными тенями)

Хранилища данных. Хранилище данных - это пассивный объект в составе DFD, в котором данные сохраняются для последующего доступа.

Потоки управления. DFD показывает все пути вычисления значений, но не показывает в каком порядке значения вычисляются. Решения о порядке вычислений связаны с управлением программой, которое отражается в динамической модели. Тем не менее, иногда бывает полезно включать указанные предикаты в функциональную модель, чтобы в ней были отражены условия выполнения соответствующего процесса. Функция, принимающая решение о запуске процесса, будучи включенной в DFD, порождает в DFD поток управления и изображается пунктирной стрелкой.

Для сложных информационных систем строится иерархия контекстных диаграмм.

Ниже приведена диаграмма потоков данных верхнего уровня с ее последующим уточнением:



Методология ERD.
(Материал из конспектов)

ERD (Entity Relationship Diagram) – представление системы как набор сущностей и связей между ними

Ключевые элементы: сущности и связи.

Сущность – объект, существенный для системы и предметной области. В сущности, много общего с классами (объектами).

- Когда говорят про сущность, обычно говорят про экземпляр
- Обладает уникальным именем
- Набором атрибутов
- Обладает одним или нескольким ключевых атрибутов
- Может иметь ряд связей с другими сущностями

Связь – поименованная взаимосвязь между сущностями.

- Обязательная/необязательная
- Одни/Много к одному/имногим
- Могут быть взаимоисключающими

Атрибут – характеристика сущности.

- Обязательный/необязательный
- Ключ/не ключ (можно идентифицировать)
- Тип (данных)

Использование диаграмм по этапам:

1. Анализ системы
    a. Используется большая контекстная диаграмма – отображает взаимодействие системы с внешним миром (DFD – верхний уровень)

    b. Матрица списка событий (ELM – заголовки use case, реакции системы)
    c. Детализация контекстных диаграмм (DFD)
2. Глобальное проектирование
    a. Модель процессов (SADT или DFD)
    b. Модель данных – ERD
    c. Модель UI – screen flow (SFD – screen flow diagram)
3. Детальное проектирование – то же что и на глобальном уровне, но подробнее
    a. Модель процессов (SADT или DFD) – подробнее чтобы уже можно было проектировать код
    b. Модель данных – ERD
    c. Модель UI – screen flow (SFD – screen flow diagram) – с привязкой к модели данных (байндингом)
4. Реализация

(Конец материала из конспектов)

Методология DATARUN.

Одной из наиболее распространенных в мире электронных методологий является методология DATARUN. В соответствии с методологией DATARUN ЖЦ ПО разбивается на стадии, которые связываются с результатами выполнения основных процессов, определяемых стандартом ISO 12207. Каждую стадию, помимо ее результатов, должен завершать план работ на следующую стадию.

1. Стадия формирования требований и планирования.
2. Стадия концептуального проектирования начинается с детального анализа первичных данных и уточнения концептуальной модели данных, после чего проектируется архитектура системы.
3. На стадии спецификации приложений продолжается процесс создания и детализации проекта.
4. На стадии разработки, интеграции и тестирования должна быть создана тестовая БД, частные и комплексные тесты. Приложения интегрируются в систему, проводится тестирование приложений в составе системы и испытания системы.
5. Стадия внедрения включает в себя действия по установке и внедрению БД и приложений.
6. Стадии сопровождения и развития включают процессы и операции, связанные с исправлением ошибок, внесением изменений и доработок, распространением новых версий ПО, переносом приложений на новую платформу и масштабированием системы. Стадия развития фактически является повторной итерацией стадии разработки.

Методология DATARUN опирается на две модели, или два представления: модель организации и модель АИС.

Методология DATARUN базируется на системном подходе к описанию деятельности организации. Построение моделей начинается с описания процессов, из которых затем извлекаются первичные данные (стабильное подмножество данных, которые организация должна использовать для своей деятельности).

Основной принцип DATARUN заключается в том, что первичные данные, если они должным образом организованы в модель данных, становятся основой для

проектирования архитектуры АИС. Архитектура АИС будет более стабильной, если она основана на первичных данных, а не на традиционной функциональной модели.
В процессе разработки АИС создается ряд моделей:

1. BPM (Business Process Model) — модуль построения модели процессов (бизнес-процессов);
2. PDS (Primary Data Structure) — структура первичных данных;
3. CDM (Conceptual Data Model) — концептуальная модель данных;
4. SPM (System Process Model) — модель процессов системы;
5. ISA (Information System Architecture) — архитектура информационной системы;
6. ADM (Application Data Model) — модель данных приложения;
7. IPM (Interface Presentation Model) — модель представления интерфейса;
8. ISM (Interface Specification Model) — модель спецификации интерфейса.

10. Понятие Enterprise Architecture Management. Матричные методологии EAM. Модель Закмана. Модель системы ARIS

Enterprise Architecture Management - практика менеджмента, которая устанавливает, поддерживает и использует согласованный набор руководящих принципов, принципов архитектуры и режимов управления, которые обеспечивают руководство и практическую помощь в проектировании и разработке архитектуры предприятия для достижения своего видения и стратегии. (Wiki)
EAM - описание ландшафта всего предприятия (более широкая точка зрения. БП - с одной стороны, информационные потоки с другой)
Отличительные черты EAM.

- Цель: комплексное описание предприятия.
- Рассматриваются различные перспективы с точки зрения разных участников.
- Цели использования таких систем:
    - Инвентаризация того, что есть (as-is)
    - Формирование ландшафта будущего - к чему хочется прийти (to-be)
    - Применение общих практик (архитектурные практики, методологи) на переходе из as-is в to-be.
    - Обратная связь от участников процесса

Такие системы рассматриваются, как средства поддержки реализации долгосрочных целей (например путь из as-is в to-be)

EAM - более управленческий подход, выстраиваем модель всего предприятия и ведём его в светлое будущее
EAM близки к онтологиям. Они структурируют возможные срезы и углы зрения как можно анализировать предприятие - это и есть модель EAM.
Обычно моделируется через BPMN, eEPC. UML очень редко.

Модель Захмана - Zachman Framework - чувак из IBM.
В начале начиналась с описания IT архитектуры, затем расширялась и на остальные отделы.
Сейчас можно описать любую структуру по созданию сложных систем.
Основана на наборе перспектив(углов зрения) для описания сложных корпоративных систем.
Пространственная структура вместо временной (вместо жизненного цикла).
Список перспектив:

1. Планировщик, инвестор, владелец бизнеса - смотрит на то, как эта деятельность вписывается в остальной ландшафт предприятия.
2. Владелец (распорядитель) - операционный управляющий предприятия
3. Конструктор или архитектор - тот кто проектирует процессы; тот, кто создаёт концептуально процессы. (уровень технического задания)
4. Проектировщик системы (уровень технического проекта)
5. Разработчик
6. Эксплуатант

1-3: Бизнес перспективы
4-6: IT-management и разработка
По другому измерению: Категории:

1. Что? - уровень сущностей, структур данных
2. Как? - это разрез описания процессов, функций, описания того как система работает.
3. Где? - зависит от уровня видения. В принципе, это инфраструктура. Может быть территориальное расположение (в какой стране), а может быть и как будут организованы подсети в локально сети.
4. Кто? - субъекты участвующие в процессе.
5. Когда? - События, которые могут приходить в систему и влиять на поведение системы.
6. Зачем? - Цели и ограничения. (Какое-то целеполагание)

Сама модель - это таблица 6 на 6 (Перспективы на категории).
Пример:
Планировщик - видит контекст:

	Что?	Как?	Где?	Кто?	Когда?	Зачем?
Планировщик - самый высокий уровень (контекст)	список основных понятий и объектов	Список БП и процессов самого верхнего уровня	Территориальные и инфраструктурные ограничения	Ключевые организации, которые взаимодействуют с предприятием	Важнейшие события (высокоуровневый план)	Бизнес-стратегия
Распорядитель		более подробная модель	внутренняя логистика предприятия		План разработки/ реализации. С точки зрения: когда должно быть разработано то-то и то-то	Уровень бизнес плана.
Операционный руководитель						
Разработчик	Детали реализации	БЛ на уровне реализации	Сетевая инфраструктура - архитектура	Роли пользователь системы, архитектура безопасности	Реал-тайм аспект разрабатываемой системы	Реализация бизнес-логики

Модель ARIS - придумали немцы
Architecture Of Integrated Information System
Включает в себя методологию и комплекс программных средств
Базовая нотация eEPC, поддерживает Entity Relation и UML.

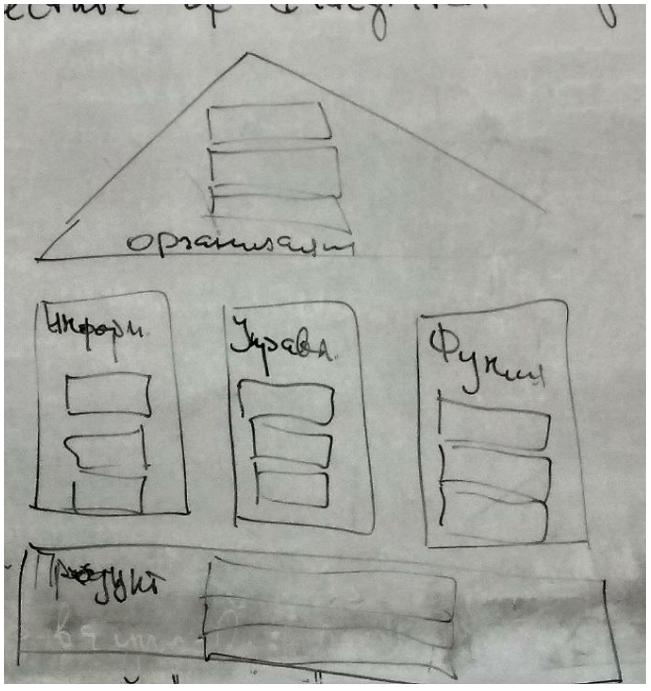
Идея: Структурированные аспекты взглядов на систему.
Похожа на Захмана, но немного по другим критериям.
Перспективы -> **Аспекты:**

1. Организационный
2. Информационный
3. Управляющий
4. Функциональный
5. Продуктовый

Для каждого аспекта 3 уровня детализации:

1. Уровень требований
2. Уровень спецификации
3. Уровень реализации

Каждое сочетание - это view - взгляд на систему, предприятие.



11. Типы и категории инструментальных средств проектирования информационных систем.

Проблема снижения рисков программирования и порога входа в программирование ⇒ появление и развитие множества диаграмм и блок схем.

Структурный подход к проектированию информационных систем – иерархическое разбиение большой системы на компоненты.

1. Сложная проблема – набор мелких независимых задач
2. Иерархическое упорядочивание (вид дерева)
3. Принцип абстрагирования (от менее существенного, что раскрывается на более глубоком уровне)
4. Принцип формализации – строгий подход к решению проблем, следование единым принципам решения проблем
5. Принцип непротиворечивости (элементы разбиения не должны противоречить друг другу)
6. Структурирование данных

Далее пойдет супер краткое перечисление названий из конспекта, за подробностями идти сюда:
https://vk.com/doc14591583_454076408?hash=0bfe89a75bf2614667&dl=52d130f755b006f6ab

1) SADT (Structure Analysis and Design Technology) – отражает функциональную суть, действия и связи между ними. Объекты: блоки (функции), дуги (связи). (EDF0 - он же)
2) DFD (Data Flow Diagram) – система как преобразователь данных. Полезна для отображения потоков данных. Диаграмма не акцентирует внимание на структуре самих данных
3) ERD (Entity Relationship Diagram) – представление системы как набор сущностей и связей между ними.

CASE (англ. computer-aided software engineering) — набор инструментов и методов программной инженерии для проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов

Акцент на этапе анализа и проектирования. Генерировать визуальное представление системы, подготовка проектной документации.
1) Репозиторий
2) Графические средства анализа и проектирования (DFD, UML и т.д.)
3) RAD – Средства быстрой разработки приложений (Генераторы кода из диаграмм, LGL языки и т.д.) Создание условных прототипов, но очень не эффективны и ограничены
4) Средства конфигурационного контроля (версионирование)
5) Средство документирования
6) Средства управления проектом

7) Средства реинжиниринга – прежде всего, относится к базам данных, восстановление метаданных, в меньшей мере к реальным программам из-за сложности
8) Средства анализа (upper Case) (IDEF, BPWIN и т.д.)
9) Средства анализа и проектирования (Oracle designer, Power Designer)
10) Средства проектирования баз данных (Oracle database designer и т.д.)
11) Средства разработки приложения (Oracle developer, Rational Rows, Silver run и т.д.)

На сегодняшний день Российский рынок программного обеспечения располагает следующими наиболее развитыми СП http://citforum.ru/database/kbd96/42.shtml

- Westmount I-CASE - интегрированный программный продукт,
- Uniface - представляет собой среду разработки крупномасштабных приложений "клиент-сервер"
- Designer/2000+Developer/2000 (ORACLE)
- SILVERRUN+JAM;
- ERwin/ERX+PowerBuilder.

В качестве примера можно привести результаты предварительного анализа перечисленных выше СП, которые сведены в краткую таблицу характеристик, приведенную ниже.

Таблица характеристик СП

СП	West-mount I-CASE + Uniface	Designer/2000+Developer/2000	SILVER-RUN + JAM	ERwin/ERX + PowerBuilder
Поддержка полного жизненного цикла ИС	+	+	+	+
Обеспечение целостности проекта	+	+	-	-
Независимость от платформы	+ (ORACLE, Informix, Sybase, Ingres и др., dbf-файлы)	- (целевая СУБД - только ORACLE)	+ (ORACLE, Informix, Sybase, Ingres и др.)	+ (ORACLE, Informix, Sybase, поддержка ODBC)
Одновременная групповая разработка БД и приложений	+	- *)	- *)	- *)

**) разработчики приложений могут начинать работу с базой данных только после завершения ее проектирования.*

Анализ данных, приведенных в таблице, показывает, что из перечисленных СП только комплекс Westmount I-CASE+Uniface наиболее полно удовлетворяет всем критериям, принятым в качестве основных. Так, например, в комплексе Westmount I-CASE+Uniface целостность базы проектных данных и единая технология сквозного проектирования ИС обеспечивается за счет использования интерфейса Westmount-Uniface Bridge. Следует отметить, что каждый из двух продуктов сам по себе является одним из наиболее мощных в своем классе.

Таким образом, наиболее развитыми средствами разработки крупномасштабных ИС на сегодняшний день является, по мнению автора, комплекс Westmount I-CASE+Uniface. С другой стороны, его применение не исключает использования в том же самом проекте таких средств, как PowerBuilder, для разработки сравнительно небольших прикладных систем в среде MS Windows.

12. Методологии и инструментальные средства проектирования ИС, основанные на понятии жизненного цикла. Методологии и инструментальные средства, основанные на понятии онтологии.

Методологии, технологии и инструментальные средства проектирования (CASE-средства) составляют основу проекта любой ИС. Методология реализуется через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства, которые обеспечивают выполнение процессов ЖЦ.

Технология проектирования определяется как совокупность трех составляющих:

- пошаговой процедуры, определяющей последовательность технологических операций проектирования (рис. 1.4);
- критериев и правил, используемых для оценки результатов выполнения технологических операций;
- нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

CASE-средства можно классифицировать по следующим признакам:

- применяемым методологиям и моделям систем и БД;
- степени интегрированности с СУБД;
- доступным платформам.

Классификация по типам в основном совпадает с компонентным составом CASE-средств и включает следующие основные типы:

- средства анализа (Upper CASE), предназначенные для построения и анализа моделей предметной области (Design/IDEF (Meta Software), BPwin (Logic Works));
- средства анализа и проектирования (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналитик (МакроПроджект)). Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;
- средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin (Logic Works), S-Designor (SDP) и DataBase Designer (ORACLE). Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;
- средства разработки приложений. К ним относятся средства 4GL (Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland)) и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично - в Silverrun;
- средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных

спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin и S-Designor. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Вспомогательные типы включают:

- средства планирования и управления проектом (SE Companion, Microsoft Project); • средства конфигурационного управления (PVCS (Intersolv));
- средства тестирования (Quality Works (Segue Software));
- средства документирования (SoDA (Rational Software)).

А. Основанные на понятии жизненного цикла.

Методология RAD

Одним из возможных подходов к разработке ПО в рамках спиральной модели ЖЦ является получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Под этим термином обычно понимается процесс разработки ПО, содержащий 3 элемента:

- небольшую команду программистов (от 2 до 10 человек);
- короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
- повторяющийся цикл, при котором разработчики, по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком

Жизненный цикл ПО по методологии RAD состоит из четырех фаз:

- фаза анализа и планирования требований;
- фаза проектирования;
- фаза построения;
- фаза внедрения.

Основные принципы методологии RAD:

- разработка приложений итерациями;
- необязательность полного завершения работ на каждом из этапов жизненного цикла;
- обязательное вовлечение пользователей в процесс разработки ИС;
- необходимое применение CASE-средств, обеспечивающих целостность проекта;
- применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
- необходимое использование генераторов кода;
- использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
- тестирование и развитие проекта, осуществляемые одновременно с разработкой;
- ведение разработки немногочисленной хорошо управляемой командой профессионалов;

- грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

Похоже, методологии из вопроса 9 тоже сюда подходят.

Б. Основанные на понятии онтологии. Подробнее здесь (вспомни Гусарову!)

Методология проектирования онтологии предметной области основывается на конструктивном использовании следующих категорий – множества концептов, отношений, функций интерпретации и аксиом.

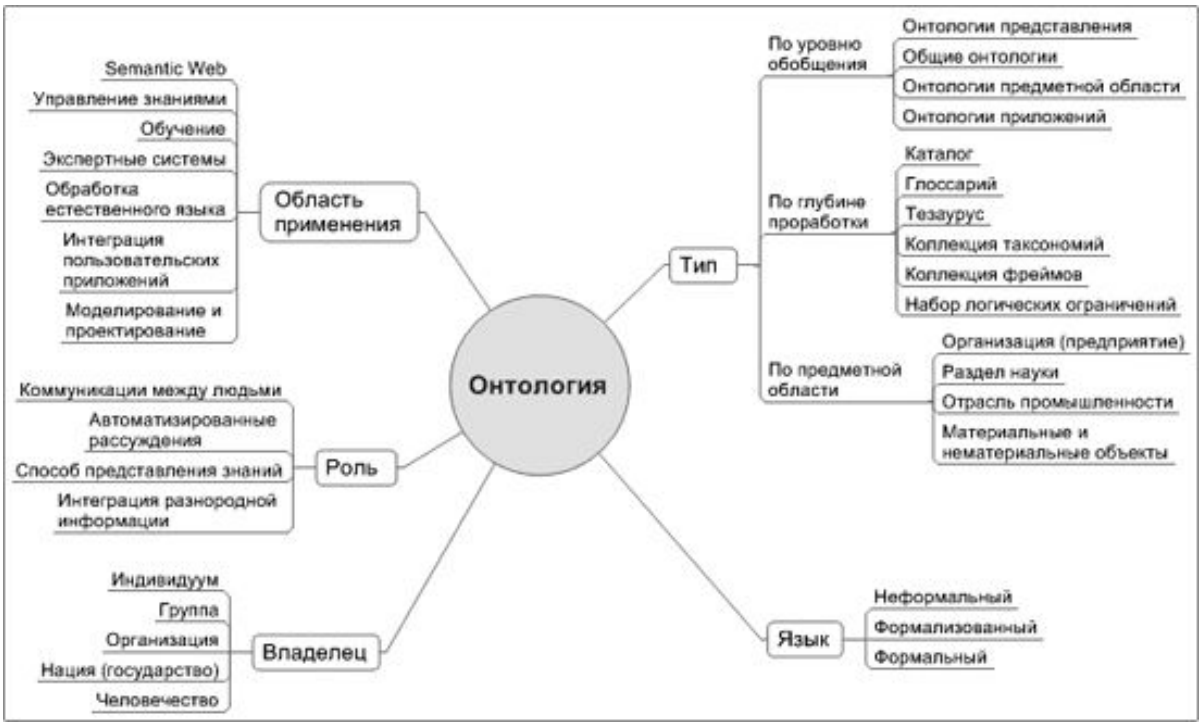
Основные характеристики комплексных программных систем онтологического инжиниринга:

- поддерживаемые формализмы и форматы представления онтологий,
- архитектура программного обеспечения,
- интерфейс пользователя,
- функционал редактора онтологии,
- средства хранения онтологий,
- доступность.

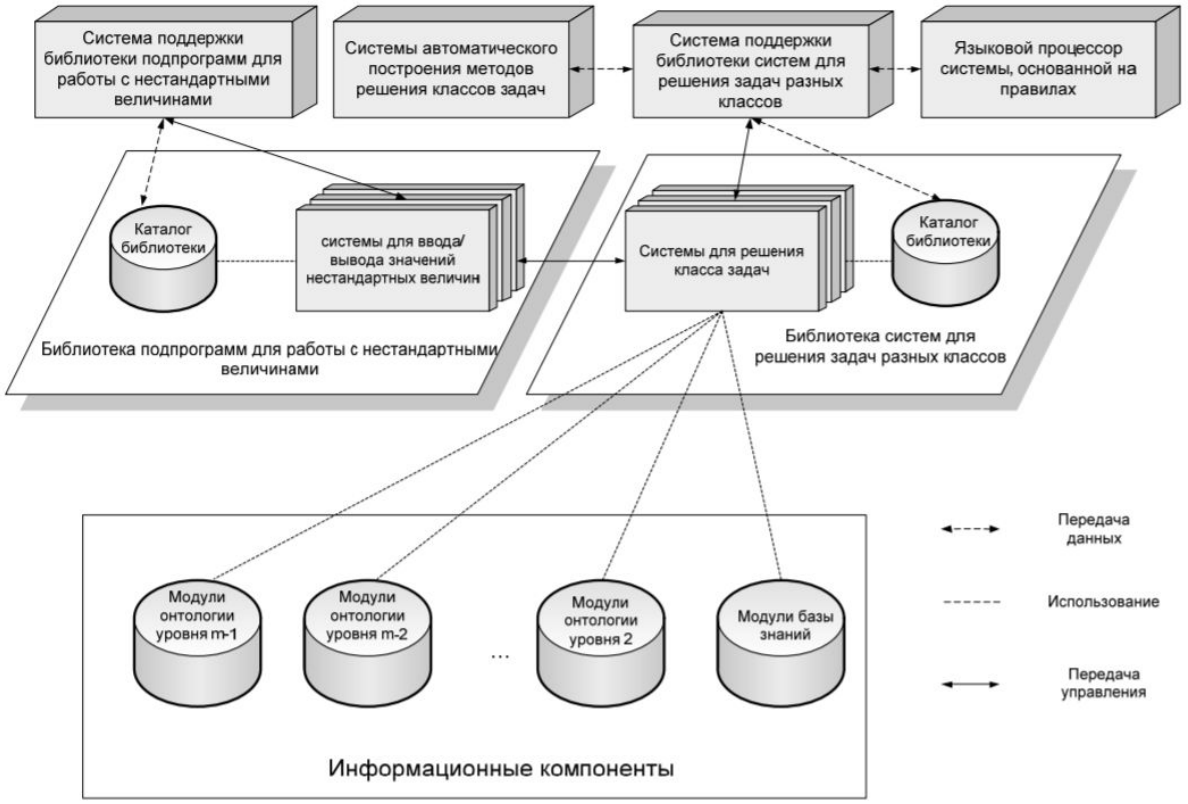
К дополнительным возможностям относят поддержку языка запросов, анализ целостности, использование механизма логического вывода, поддержку удаленного доступа через Интернет, документирование.

Известны три группы инструментальных средств онтологического инжиниринга
.

1. Инструменты создания онтологий, которые предполагают поддержку совместной разработки и просмотра, создание онтологии в соответствии с заданной (произвольной) методологией, поддержку рассуждений.
2. Инструменты объединения, отображения и выравнивания онтологий. Объединение предполагает нахождение сходств и различий между исходными онтологиями и создание результирующей онтологии, которая содержит элементы исходных онтологий.
3. Инструменты для аннотирования Web-ресурсов на основе онтологий.



Систематизация знаний в области онтологий



Системы для решения задач и средства их разработки

Специализированная система обработки текстовых документов «ЛоТА» - является системой класса Text Mining. Система предназначена для анализа специализированных текстов «Логика работы», описывающих логику работы сложной технической системы в различных режимах функционирования.

Основной задачей анализа является извлечение из данных текстов информационной модели алгоритмов, решающих определенную задачу в определенной проблемной ситуации, и контроль структурной и информационной целостности выделенной схемы алгоритмов.

Информационная модель алгоритма включает:

- описание входного информационного потока (типы информационных сигналов или семантическое описание информационного потока с указанием источника информации конкретный алгоритм, конкретное измерительное устройство);
- описание процессов преобразования входных данных в выходные (допустимый способ разрешения проблемы);
- описание выходного информационного потока (типы информационных сигналов или семантическое описание информационного потока с указанием точки приема информации).

Решение основной задачи обеспечивается комплексом технологий обработки текстов, включающих:

- технологии морфосинтаксического анализа;
- технологии семантико-синтаксического анализа;
- технологии взаимодействия с прикладной онтологией.

Интеллектуальная система извлечения данных и их анализа (на основе текстов) ИСИДА-Т - является извлечение значимой информации определенного типа из (больших массивов) текста для дальнейшей аналитической обработки. Результатом работы систем является получение структурированных данных и отношений на них.

Основные компоненты ИСИДА-Т:

- Инфраструктурные службы (конфигурирование, параллельная обработка, взаимодействие модулей);
- Лингвистический процессор;
- Модули работы со знаниями ПрО;
- Интерпретатор правил извлечения информации.

Разработанные в рамках проекта ИСИДА-Т технологии, инструменты и продукты позволяют:

- обнаруживать в электронных документах,
- извлекать и структурировать информацию о представляющих интерес фактах, событиях, объектах и отношениях;
- выполнять мониторинг сайтов в сети Интернет на предмет появления там значимой для пользователя информации

Некоторые области применения технологий семантического анализа и структурирования текстовой информации:

- информационная поддержка бизнеса (business intelligence) и управление знаниями (knowledge management);
- маркетинговые исследования;
- финансовая аналитика;

- военная и коммерческая разведка и мониторинг;
- информационная поддержка органов государственной власти (в рамках направления «Электронное правительство»);

Инструментальное средство проектирования онтологий Protégé - локальная, свободно распространяемая Java-программа. Предназначена для построения онтологий прикладной области. Её первоначальная цель – помочь разработчикам программного обеспечения в создании и поддержке явных моделей предметной области и включение этих моделей непосредственно в программный код.

Protégé включает редактор онтологий, позволяющий проектировать онтологии, разворачивая иерархическую структуру абстрактных или конкретных классов и слотов. Структура онтологии сделана аналогично иерархической структуре каталога. На основе сформированной онтологии, Protégé может генерировать формы получения знаний для введения экземпляров классов и подклассов.

Инструмент имеет графический интерфейс, удобный для использования неопытными пользователями, снабжен справками и примерами. Protégé основан на фреймовой модели представления знания OKBC (Open Knowledge Base Connectivity) и снабжен рядом плагинов, что позволяет адаптировать его для редактирования моделей, хранимых в разных форматах (стандартный текстовый, в базе данных JDBC, UML, языков XML, XOL, SHOE, RDF и RDFS, DAML+OIL, OWL).

Применение онтологии в enterprise.
К настоящему моменту существует ряд примеров использования онтологий в организационном моделировании и управлении:

1. Enterprise Project [Stader, 1996; Uschold, 1997],
2. Process Specification Language (PSL) project [Bock, 2005],
3. Toronto Virtual Enterprise (TOVE) project [Fox, 1992; Gruninger, 2000],
4. SUPER Project [Born, 2007; Hepp, 2007],
5. Понятийное и объектное моделирование властных структур на региональном уровне ИПУСС РАН [Виттих и др., 2006]
6. Конфигурирование услуг электронного государства на основе онтологий —Onto-Gov [Abecker, 2004]

Подробнее про каждую здесь.

13. Различные типы финансирования программных проектов. Fixed Price. T&M. Трудозатраты, сроки и стоимость проекта. Основные компоненты, из которых складывается стоимость проекта.

Наиболее популярными на сегодняшний день являются два типа финансирования программных проектов - Fixed Price и Time and Materials.

Особенности Fixed Price:

- фиксированный бюджет
- фиксированный объём работ
- фиксированные временные рамки
- отсутствие возможности внести изменения или дополнения после подписания контракта
- возможные компромиссы относительно качества продукта
- исполнитель склонен закладывать непредвиденные расходы в бюджет

Особенности Time and Materials:

- проект разбивается на подзадачи, каждая из которых оценивается отдельно
- заказчик оплачивает каждую задачу по отдельности, основываясь на времени, затраченном на разработку
- возможность вносить изменения по ходу проекта без необходимости перерасчёта стоимости всего проекта

Fixed price более привлекателен для заказчика, однако применение такого типа финансирования целесообразно в случае относительно небольшого размера проекта и строго ограниченного функционала, в который не будут вноситься изменения. В иных случаях применение такой модели финансирования ведёт к невыполнению условий контракта, резкому снижению качества продукта или уходу исполнителя в убыток.

В то же время, Time and Materials - более гибкая модель, которая позволяет вносить изменения по ходу проекта и приступить к работе над проектом почти сразу. Такая модель требует более тесного взаимодействия между заказчиком и исполнителем.

Советы при формировании бюджета (из лекций) - максимальная детализация выполняемых работ:

- явно определить критерии оценки и приёмки
- составить как можно более полный список требований и ограничений
- составить максимально детальный перечень задач
- явно выделять внутренние задачи
- включать в ТЗ обсуждаемые пункты, которые было решено не делать
- зафиксировать, что должно быть выполнено силами заказчика
- как можно раньше показывать прототипы заказчику
- включить в договор механизмы возможных изменений
- в рамках договора выделить буфер для “хотелок” заказчика

Оценка трудоёмкости (из чего складывается стоимость проекта):

- разработка (база)
- затраты на управление проектом (~15%)
- затраты на тестирование (~20-30%)
- встречи с заказчиком
- написание требований, написание ТЗ
- написание и согласование самого договора
- налаживание инфраструктуры
- демонстрации для заказчика
- непредвиденные риски (>=10%)
- буфер на рефакторинг и внесение изменений (>=10%)

Чтобы как можно лучше оценить базу, необходимо максимально мелкое деление на подзадачи

Новый разработчик в среднем встраивается неделю

Оценка времени (черт знает какого, но пусть будет) : 2.5 * (корень кубический из трудозатрат). Эмпирическая формула

Следующий вопрос - это стоимость.
На основе нормы часа. Как это?
Пусть есть разработчик, который получает 100 условных единиц (у.е.). На руки получает 87.
Затем проценты от зп разработчика
На это начисляются налоги , которые начисляет предприятия 13-30% (страховка, фсс и т.д.)
Накладные расходы предприятия – это аренда, амортизация оборудования, зарплаты бухгалтеров и управленцев, интернет, зарплата персонала: 50-70% в небольшой, в крупной до 300%
Плановая прибыль: 20%
Получаем ставку в месяц после суммирования – Rate
Делим на кол-во часов в месяце и получаем – Брутто – Рейт
Проблема: если рейт слишком высокий, то ты никому не нужен, если слишком низкий, то работаешь в убыток.
В первом приближении, стоимость проекта - это трудоемкость проекта умноженная на рейт.
Возможны вариации, когда учитывают разные рейты для разных родов деятельностей и зп.

14. Статистические коэффициенты для оценки различных компонент проекта. Различные методы оценки базовой трудоемкости: экспертная оценка, метод PERT, метод функциональных точек. Формула Боэма для расчета продолжительности проекта. Оценка стоимости проекта на основе брутто-рейтов.

По лекциям ребят:
Что такое оценка трудоемкости - все работы связанные с исполнением проекта.
Что включает:
1. Работы по разработке проекта – база, от неё делаются производные оценки
2. Затраты на управление проектом
3. Затраты на тестирование
4. Встреча с заказчиком
5. Написание требований
6. Написание ТЗ
7. Написание и согласование самого договора
8. Установка софта, средств разработки, налаживание инфраструктуры
9. Демонстрации для заказчика
10. Непредвиденные риски
11. Буфер на внесение изменений и рефакторинг

Как проходит оценка задач? Берется несколько опытных программистов и у них спрашивается. Ищется min и max и берется среднее
Пусть мы получили оценку базы, будем считать проценты от базы
Тестирование 20%-30%
Менеджмент 15%
Риски > 10%
Изменения > 10%
Новый разработчик 1 неделя
Еженедельные митинги 4 часа в неделю
Демонстрация заказчику 1 день подготовка + 4 часа на демонстрацию
Пример:
Пусть 120 чел/мес.
Критерии, влияющие на сроки:
1. Размер команды
2. Связность компонентов проекта (плохо делится)
3. Время сыгрывания команды
4. Динамика ввода новых участников в команду
Время = 2.5 (Трудозатраты ^ (0.333)) – 25% - лучшая оценка времени от привлечения сотрудников. Эмпирическая формула. Смысл формулы может быть записан неверно.
Следующий вопрос - это стоимость.
На основе нормы часа. Как это?
Пусть есть разработчик, который получает 100 условных единиц (у.е.). На руки получает 87.
Затем проценты от зп разработчика

На это начисляются налоги , которые начисляет предприятия 13-30% (страховка, фсс и т.д.)
Накладные расходы предприятия – это аренда, амортизация оборудования, зарплаты бухгалтеров и управленцев, интернет, зарплата персонала: 50-70% в небольшой, в крупной до 300%
Плановая прибыль: 20%
Получаем ставку в месяц после суммирования – Rate
Делим на кол-во часов в месяце и получаем – Брутто – Рейт
Проблема: если рейт слишком высокий, то ты никому не нужен, если слишком низкий, то работаешь в убыток.
В первом приближение, стоимость проекта это трудоемкость проекта умноженная на рейт.
Возможны вариации, когда учитывают разные рейты для разных родов деятельностей и зп.

Один из них PERT. Если есть набор экспертных оценок, как из них получить интегральную оценку, которая лучше усреднения. Считаем что есть набор назависимых мелких задач. Есть оценка из трех чисел:
Mi – наиболее вероятное (medium)
Oi – минимально возможное (optimistic)
Pi – максимально возможное (pessimistic)
Набираем такие тройки. Тогда утверждается в этом методе, что средняя трудоемкость Ei
Ei = (Pi + 4Mi + Oi) / 6
Среднеквадратичное отклонение = (Pi – Oi)/6
Суммарная трудоемкость E = сумма по всем Ei
Суммарное отклонение = sqrt(Сумма по всем среднеквадратичным отклонениям ^ 2)

https://studfiles.net/preview/2438406/page:16/

Метод функциональных точек

Метод функциональных точек используется для оценки времени разработки на ранних стадиях (этапах) проекта, например, на этапе логического и концептуального проектирования. Для использования данного метода необходимо иметь перечень требований к разрабатываемому программному обеспечению. Точность оценки зависит от уровня детализации требований. Данный метод используется для оценки производительности труда разработчиков и объема работы.
При анализе методом функциональных точек надо выполнить следующую последовательность шагов:
– определение типа оценки;
– определение области оценки и границ продукта;
– подсчет функциональных точек, связанных с данными;

– подсчет функциональных точек, связанных с транзакциями;
– определение суммарного количества не выровненных функциональных точек;
– определение значения фактора выравнивания;
– расчет количества выровненных функциональных точек.

https://project.dovidnyk.info/index.php/programnye-proekty/tehnologiyarazrabotkiprogrammnogoobespecheniya/22-upravlenieproektom

В модели Боэма утверждается, прежде всего, что для разных типов приложений длительность и трудозатраты по-разному зависят от размера приложений (отличаются множителем и показателем экспоненты). Например, разработка приложения в 20 000 строк потребует 2,4 x 20 «51 человеко-месяц, если приложение органично, и потребует 3,6 x 20 « 76 человеко-месяцев, если это встроенное приложение. Формула для длительности может быть прямо выражена через количество строк кода:. Длительность = c x Трудозатраты^ = c x (a x KLOC) = c x a x KLOC

15. Понятие моделирования бизнес-процесса. Различные подходы к моделированию бизнес-процессов. Системы визуального моделирования. Нотации EPC, BPMN, IDEF

Моделирование бизнес-процессов – это эффективное средство поиска путей оптимизации деятельности компании, позволяющее определить, как компания работает в целом и как организована деятельность на каждом рабочем месте. Моделирование бизнес-процессов позволяет точно представить цели, исследуемые характеристики и конечные результаты каждого вида деятельности. Бизнес-процессы определяют прохождение потоков работ независимо от иерархии и границ подразделений, которые их выполняют.
Обобщенная модель бизнес-процесса может быть представлена как совокупность:

- сущностей: рабочих объектов, ресурсов, организационных единиц;
- функций (действий);
- событий.

Существуют различные подходы к отображению модели бизнес-процессов, среди которых выделяются функциональный и объектно-ориентированный подходы. В функциональном подходе главным структурообразующим элементом является функция (действие), в объектно-ориентированном подходе - объект.
Наибольшую перспективу представляют комплексные методологии моделирования бизнес-процессов, например ARIS - технология, позволяющие в зависимости от целей анализа бизнес-процессов выбирать адекватные модели.

ARIS eEPC

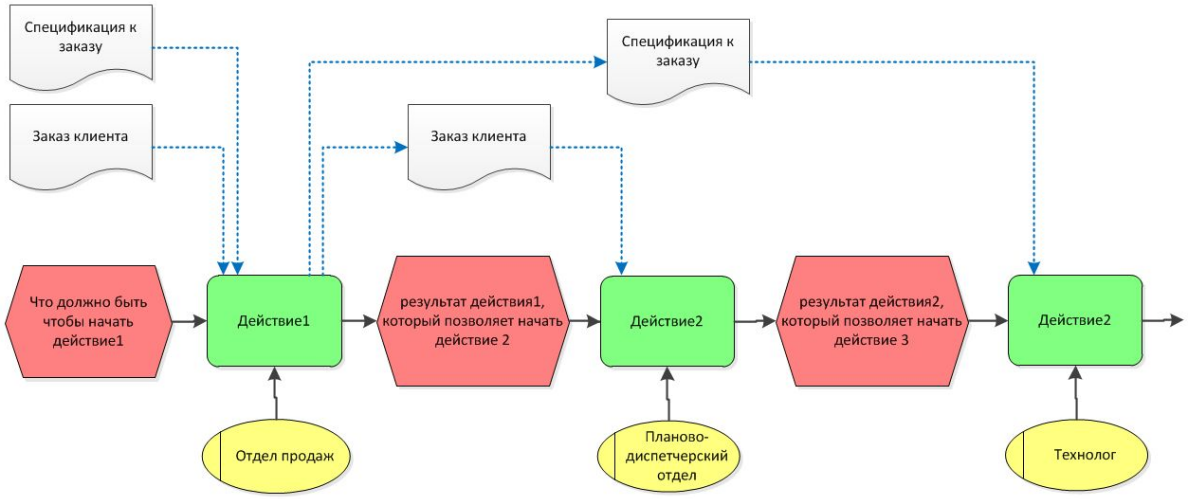
ARIS (Architecture of Integrated Information Systems) - интегрированное средство моделирования. Разработан германской фирмой IDS Scheer.
ARIS поддерживает четыре типа моделей (и множество видов моделей в каждом типе), отражающих различные аспекты исследуемой системы.
Поддерживаемые типы моделей в ARIS:

1. Организационные модели, представляющие структуру системы — иерархию организационных подразделений, должностей и конкретных лиц, связи между ними, а также территориальную привязку структурных подразделений.
2. Функциональные модели, содержащие иерархию целей, стоящих перед аппаратом управления, с совокупностью деревьев функций, необходимых для достижения поставленных целей.
3. Информационные модели, отражающие структуру информации, необходимой для реализации всей совокупности функций системы.
4. Модели управления, представляющие комплексный взгляд на реализацию бизнес-процессов в рамках системы.

Для построения перечисленных типов моделей используются как собственные методы моделирования ARIS, так и различные известные методы и языки моделирования, в частности, UML.
Основная бизнес-модель ARIS - eEPC (extended Event-driven Process Chain, расширенная модель цепочки процессов, управляемых событиями). Нотация ARIS eEPC является расширением нотации IDEF3. Бизнес-процесс в нотации eEPC

представляет собой поток последовательно выполняемых работ (процедур, функций), расположенных в порядке их выполнения.
Основные объекты:

1. Функция. Служит для описания функций (процедур, работ), выполняемых подразделениями/сотрудниками предприятия. Каждая функция должна быть инициирована событием и должна завершаться событием; в каждую функцию не может входить более одной стрелки, "запускающей" выполнение функции, и выходить более одной стрелки, описывающей завершение выполнения функции.
2. Событие. Служит для описания реальных событий, воздействующих на выполнение функций.
3. Организационная единица. Например, управление или отдел.
4. Документ. Отражает реальные носители информации, например, бумажные документы.
5. Прикладная система.
6. Кластер информации. Характеризует набор сущностей и связей между ними.
7. Связь между объектами. Тип отношений между объектами, например, активация выполнения функции некоторым событием.
8. Логический оператор. Оператор "И", "ИЛИ" или исключающее "ИЛИ", позволяет описать ветвление процесса.



BPMN

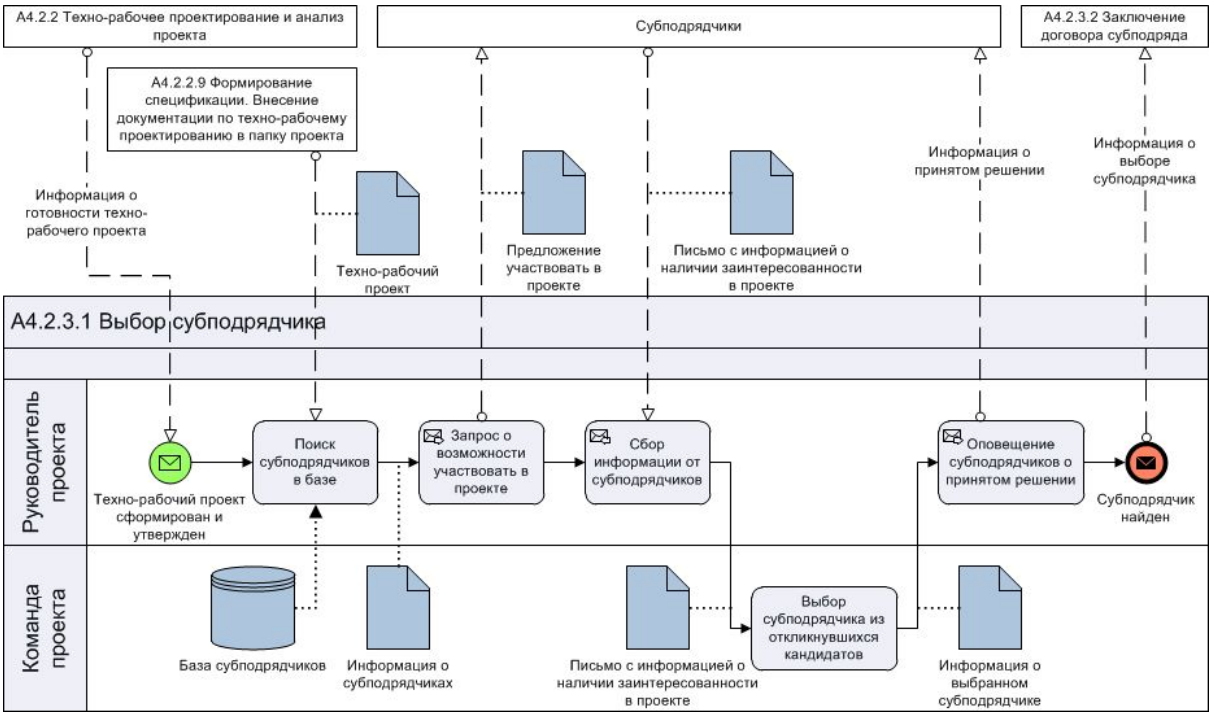
Нотация BPMN (Business Process Model and Notation - модель бизнес-процессов и нотация) используется для описания процессов нижнего уровня.
Диаграмма процесса в нотации BPMN представляет собой алгоритм выполнения процесса.
На диаграмме могут быть определены события, исполнители, материальные и документальные потоки, сопровождающие выполнение процесса.
Каждый процесс может быть декомпозирован на более низкие уровни.
В нотации BPMN выделяют пять основных категорий элементов:

1. элементы потока (события, процессы и шлюзы);
2. данные (объекты данных и базы данных);
3. соединяющие элементы (потоки управления, потоки сообщений и ассоциации);

4. зоны ответственности (пулы и дорожки);
5. артефакты (сноски).

Примеры элементов:

- Процесс (Задача, Подпроцесс) - действие или набор действий, выполняемых над исходным объектом деятельности (документом, ТМЦ и прочим) с целью получения заданного результата.
- Событие - состояние, которое является существенным для целей управления бизнесом и оказывает влияние или контролирует дальнейшее развитие одного или более бизнес-процессов.
- Элементы ветвления (Параллельный шлюз, Эксклюзивный шлюз, Неэксклюзивный шлюз).
- Поток управления. Используется для связи элементов потока BPMN (событий, процессов, шлюзов). Поток управления отображает ход выполнения процесса. При необходимости поток может быть именованным.
- Условный поток управления. По такому потоку будет происходить дальнейшее выполнение процесса только в том случае, если выполнится условие, указанное в названии потока.
- Пул предназначен для отображения потока рассматриваемого процесса. Содержимое пула - это и есть тот процесс, диаграмма которого рассматривается. На диаграмме развернутый пул может быть только один.
- Дорожка предназначена для отображения организационных единиц (должности, подразделения, роли, внешнего субъекта) - исполнителей задач и подпроцессов процесса BPMN. Внутри блока помещается наименование организационной единицы.
- Объект данных. Используется для отображения на диаграмме объектов деятельности, сопровождающих выполнение процесса. Рядом с блоком размещается наименование объекта данных.
- База данных. Используется для отображения на диаграмме базы данных, сопровождающей выполнение процесса. Рядом с элементом размещается наименование базы данных.



IDEF

IDEF - это семейство нотаций. Различаются они по порядковым номерам — IDEF0, IDEF1, IDEF2 и т.д. Каждая нотация имеет свои особенности и используется для описания разных элементов бизнес-системы.

Методология IDEF0 базируется на методе SADT (Structured Analysis and Design Technique) Росса, предназначенном для структурированного представления функций системы и анализа системных требований.

IDEF0-модель состоит из диаграмм и фрагментов текста. На диаграммах все функции системы и их взаимодействия представлены как блоки (функции) и дуги (отношения).

Основные элементы модели:

1. Функциональный блок (Activity) – преобразование (активность);
2. Выходы (Output) – результат преобразования;
3. Входы (Input) — объекты, которые преобразуются в Выходы;
4. Управление (Control) — информация, как происходит преобразование;
5. Механизм (Mechanism) – объекты, осуществляющие преобразование.

Функциональный блок может быть декомпозирован — представлен в виде совокупности других взаимосвязанных блоков, которые детально описывают исходный блок.

Таким образом, IDEF0-модель состоит из набора иерархически связанных диаграмм

На диаграмме блоки соединяются дугами: выходные дуги одних блоков могут являться входами (управлением, механизмом) других.

Дуги с одним свободным концом имеют источник или получатель вне диаграммы. Для обозначения внешних дуг используются буквы:

- I (Input),
- C (Control),
- O (Output) и
- M (Mechanism).

Типы связей между блоками:

1. Выход-вход
2. Выход-управление
3. Выход-механизм
4. Обратная связь по управлению
5. Обратная связь по входу

Методология IDEF3

IDEF3-модели используются для документирования технологических (информационных) процессов, где важна последовательность выполнения процесса
Выделяют четыре элемента IDEF3-модели:

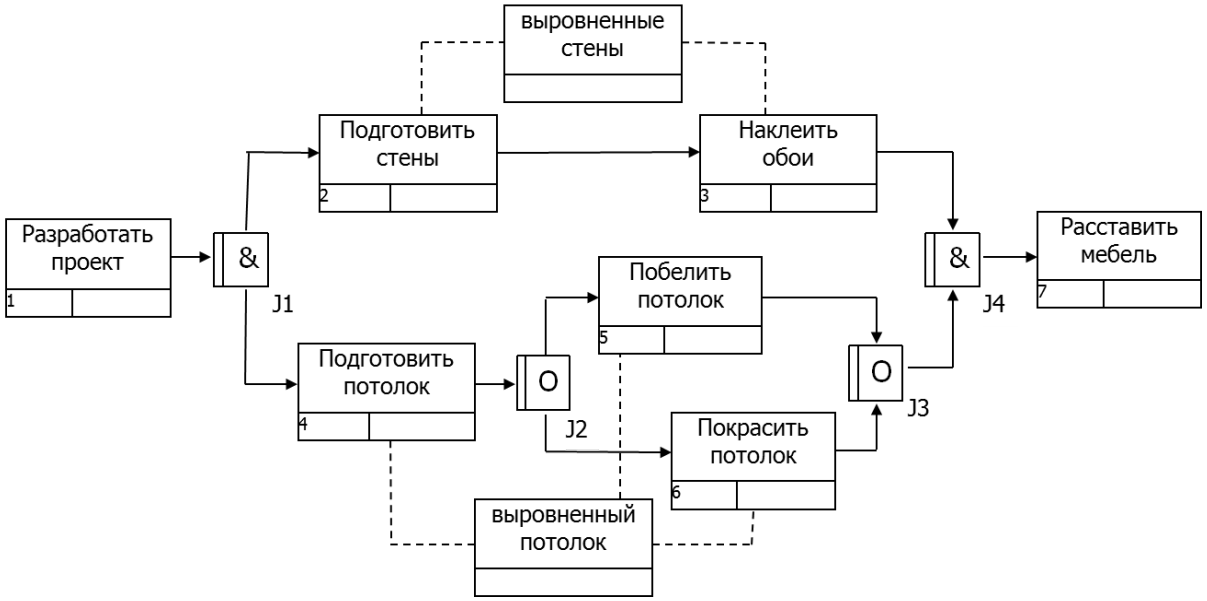
1. Единицы работ (Unit of work) — отображают действия, процессы, события, этапы выполнения работ. Единица работы может иметь только один вход и один выход
2. Ссылки (Referents):
    a. необходимые элементы для выполнения процесса (сырье, материалы);
    b. результат процесса (изделие);
    c. активаторы процесса (клиент, поставщик).
3. Связи (Links), которые бывают двух типов:
    a. передают действия от одной единицы работ к другой
    b. соединяют ссылку с единицей работ (активируют единицу работ)
4. Перекрестки (Junctions) – элементы модели, за счет которых описывается логика и последовательность выполнения этапов процесса. Бывают двух видов:
    a. перекрестки слияния – Fan-in
    b. перекрестки ветвления – Fan-out

Типы перекрестков

1. Асинхронное И (Asynchronous AND)
    a. выходной процесс запустится, если завершились все входные процессы
    b. после завершения входного процесса запустятся все выходные процессы
2. Синхронное И (Synchronous AND)
    a. выходной процесс запустится, если завершились одновременно все входные процессы
    b. после завершения входного процесса запустятся все выходные процессы, причем запустятся одновременно
3. Асинхронное ИЛИ (Asynchronous OR)
    a. выходной процесс запустится, если завершится один или несколько входных процессов
    b. после завершения входного процесса запустятся один или несколько выходных процессов
4. Синхронное ИЛИ (Synchronous OR)
    a. выходной процесс запустится, если завершились один или несколько входных процессов, причем завершились одновременно
    b. после завершения входного процесса запустится один или несколько выходных процессов, причем запустятся одновременно
5. Исключающее ИЛИ (XOR, Exclusive OR)
    a. выходной процесс запустится, если завершился только один входной процесс

b. после завершения входного процесса запустится только один выходной процесс

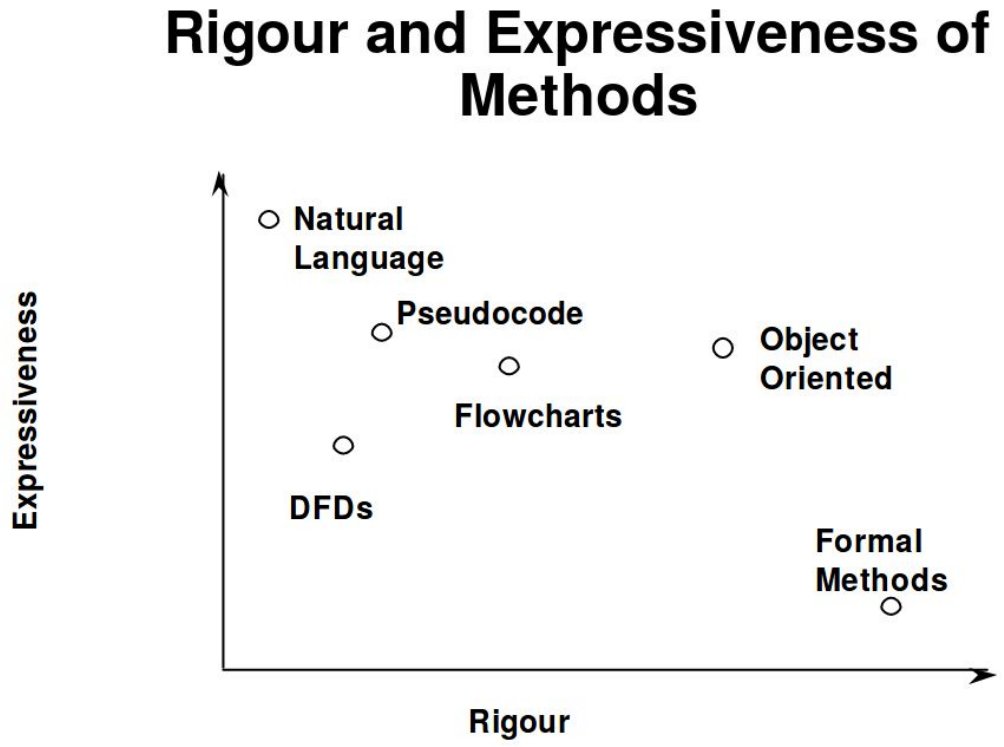
Пример IDEF3



16. Различные формы представления результатов проектирования информационных систем. Описания бизнес-процессов. Списки требований. Архитектурные диаграммы. Используемые нотации. Техническое задание и технический проект.

Различные формы представления результатов проектирования информационных систем

http://docplayer.net/59539265-Systems-concepts-and-modelling.html:



Описания бизнес-процессов.

Вертикальное и горизонтальное описания

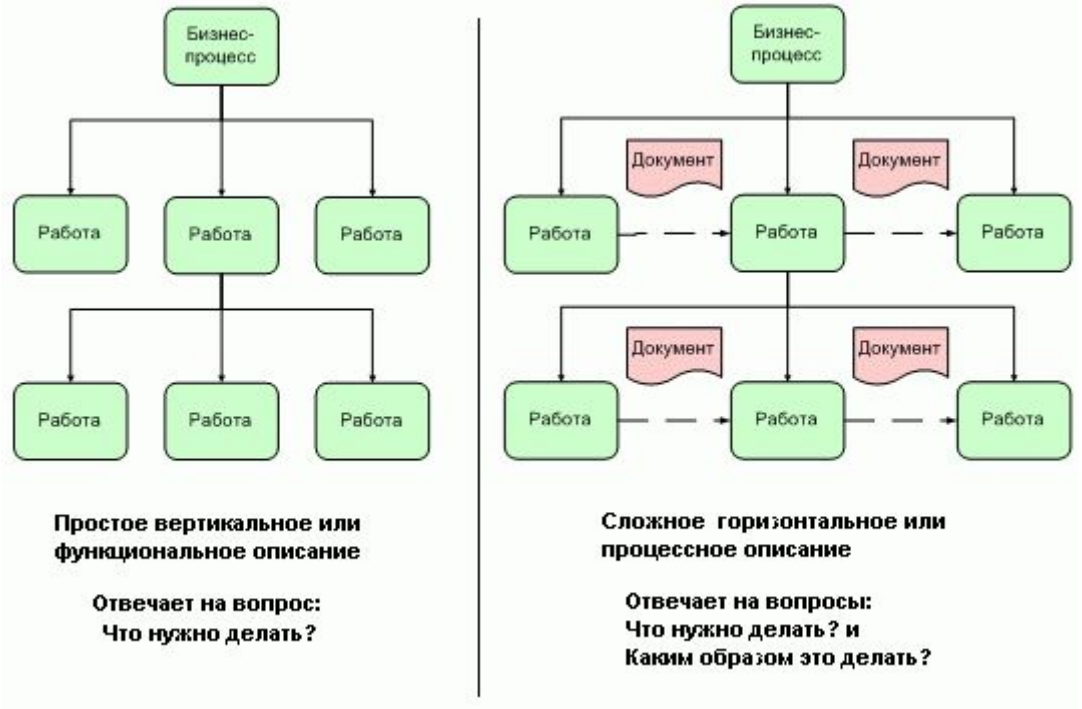
(https://studfiles.net/preview/1484405/page:7/)

Существует два вида инструментов, применяемых при описании бизнес-процессов – вертикальное и горизонтальное описание.

При вертикальном описании показывают только работы и их иерархический порядок в дереве бизнес-процесса. В этом случае имеются только вертикальные связи между родительскими и дочерними работами.

При горизонтальном описании так же показывается, как эти работы между собой взаимосвязаны, в какой последовательности они выполняются, какие информационные и материальные потоки между ними движутся. В этом случае в

модели бизнес-процесса появляются горизонтальные связи между различными работами, которые процесс составляют.



Специалисты по организационному проектированию используют различную терминологию при описании бизнес-процессов. Например, вертикальное описание называют функциональным описанием, горизонтальное – процессным описанием или просто описанием бизнес-процессов.

Формы описания

(http://rzbpm.ru/knowledge/opisanie-biznes-processov-tipy-opisaniya.html)

Описание бизнес процессов можно делать разными способами. Каждый имеет как плюсы, так и минусы. Можно выделить 3 типа описания — текстовый, табличный и графический. Естественно, в чистом виде они встречаются редко. В большинстве случаев мы комбинируем эти методы, в том или ином виде.

Текстовое описание бизнес процессов

Наверное самый простой в реализации и распространенный вариант. Все, что происходит в бизнес процессе описывается словами, т.е. в итоге у нас получается текст. Построение бизнес процессов требует описания довольно таки большого количества элементов и вариантов развития бизнес процесса, текст может получиться весьма громоздким.

Плюсы:

- Очень просто сделать
- Не требует специальных навыков

Минусы:

- Текст сложно обрабатывать
- Затрудняет целостное восприятие процесса
- В принципе сложно для восприятия

- Сложно структурировать и анализировать

Подсказка — используйте структурированные списки

Описание бизнес процессов в виде таблиц

На самом деле, описание бизнес процессов компании в виде связанных таблиц, не такая уж плохая идея. По крайней мере намного лучше текстового описания. Самая большая сложность заключается в том, чтобы подготовить хороший шаблон таблицы, в которую, собственно, потом уже внесут данные.

Плюсы:

- Относительно просто подготовить
- Относительно просто заполнить шаблон
- Наличие структуры
- Удобство обработки цифровых данных

Минусы:

- Не компактно
- Отсутствует необходимая детализация
- Нет целостности восприятия
- Сложно отобразить ветвления
- Требует подготовки

Подсказка — располагайте подпроцессы, операции в строках, а данные в столбцах. Это облегчает восприятие. Вместо одной, большой таблицы, используйте связанные таблицы.

Описание в виде схемы, модели бизнес процесса

Графическое описание бизнес процессов предприятия, в виде схемы, модели — лучший тип описания. Большинство специалистов, уже давно отдают предпочтение именно графическому типу описания.

Плюсы:

- Простота восприятия
- Целостность восприятия
- Необходимая и достаточная детализация
- Наглядное отображение ветвлений и путей развития процесса
- Удобство автоматизации

Минусы:

- Требует специальных навыков
- Относительно больше время на подготовку описания

Инвестиции в графический тип описания окупаются весьма неплохо.

Подсказка — для описания бизнес процессов может быть достаточно 3-5 графических элементов (фигур).

* * *

См. также https://www.u-b-s.ru/publikacii/biznes-processy.html.

Списки требований

Спецификация требований

(https://ru.wikipedia.org/wiki/Анализ_требований#Списки_требований)

Списки требований

Традиционный способ документировать требования — это создание списков требований. Соответствующей метафорой может быть чрезвычайно длинный список покупок в магазине. Такие списки крайне неэффективны в современном анализе, хотя используются и по сей день.

Преимущества

- Обеспечивает контрольный список требований.
- Обеспечивает договор между заказчиками и разработчиками.
- Для большой системы может обеспечить описание высокого уровня.

Недостатки

- Такие списки могут занимать сотни страниц. Фактически невозможно прочитать такие документы в целом и получить чёткое понимание системы.
- Такие списки требований перечисляют отдельные требования абстрактно, оторвано друг от друга и от контекста использования
    - Эта абстракция лишает возможности видеть, как требования связываются между собой или работают вместе.
    - Эта абстракция мешает верно расположить требования по приоритетам; в то время как список, действительно облегчает приоритезацию отдельных пунктов, удаление одного пункта из контекста может сделать весь сценарий использования или деловое требование бесполезным.
    - Эта абстракция увеличивает вероятность неверного трактования требований; поскольку чем больше число людей будет их читать, тем большее будет число (различных) интерпретаций системы.
    - Эта абстракция означает, что чрезвычайно трудно убедиться, что у вас есть все необходимые требования.
- Эти списки создают ложное чувство взаимопонимания между заинтересованными лицами и разработчиками.
- Эти списки дают заинтересованным лицам ложное чувство защищённости, что разработчики должны достигнуть определённых вещей. Однако, из-за природы этих списков, они неизбежно упускают важные требования, которые будут выявлены позже в процессе. Разработчики могут использовать новые требования для пересмотра сроков и условий в их пользу.

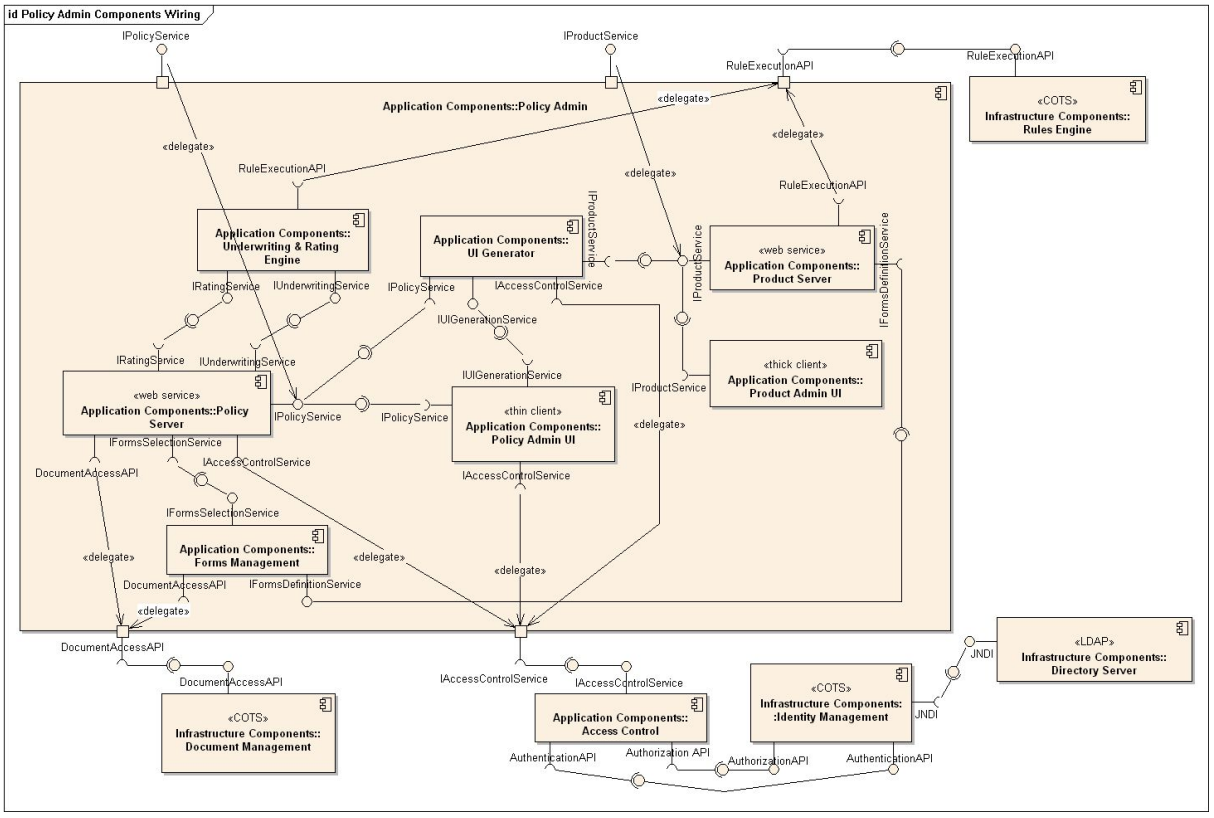
Архитектурные диаграммы

(The Art of Crafting Architectural Diagrams)

Архитектурные диаграммы UML

Диаграмма компонентов

(https://ru.wikipedia.org/wiki/Диаграмма_компонентов)



Диагра́мма компоне́нтов — статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

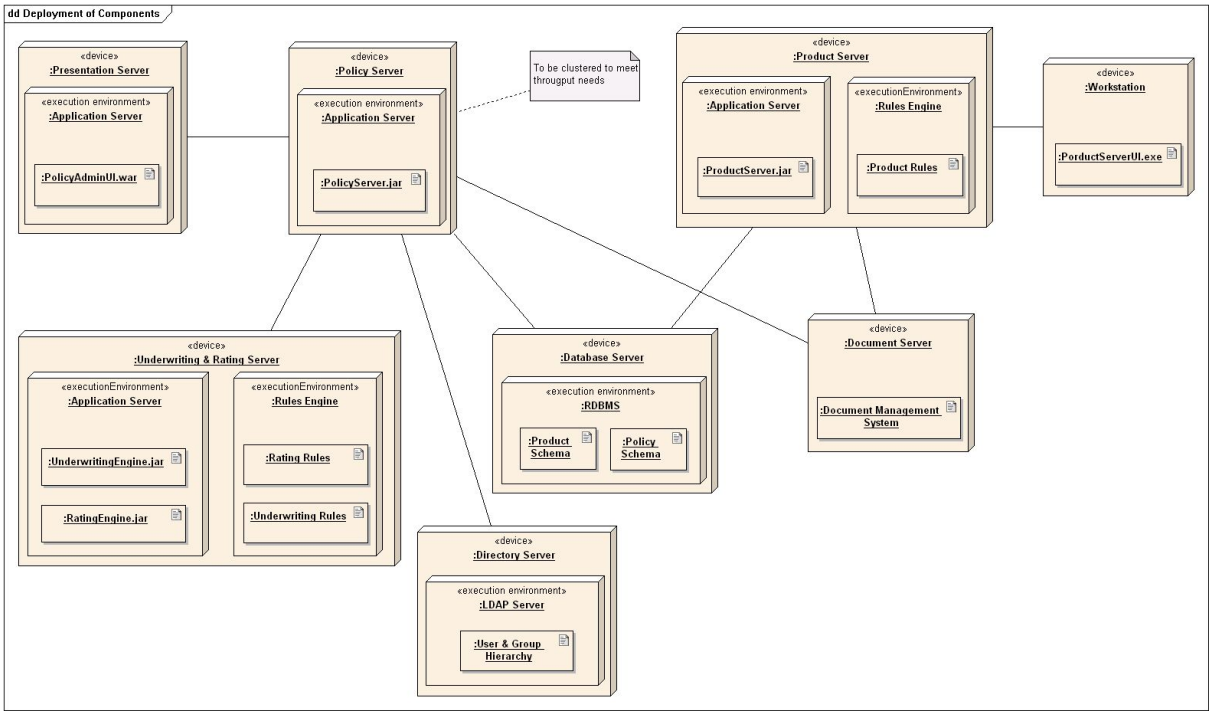
Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывается связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

Диаграмма развёртывания

(https://ru.wikipedia.org/wiki/Диаграмма_развёртывания)



Диагра́мма развёртывания, Deployment diagram в UML моделирует физическое развертывание артефактов на узлах.[1] Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI).

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь подузлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов:

1. Узел устройства
2. Узел среды выполнения

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны. Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

Диаграмма вариантов использования

(https://ru.wikipedia.org/wiki/Диаграмма_прецедентов)

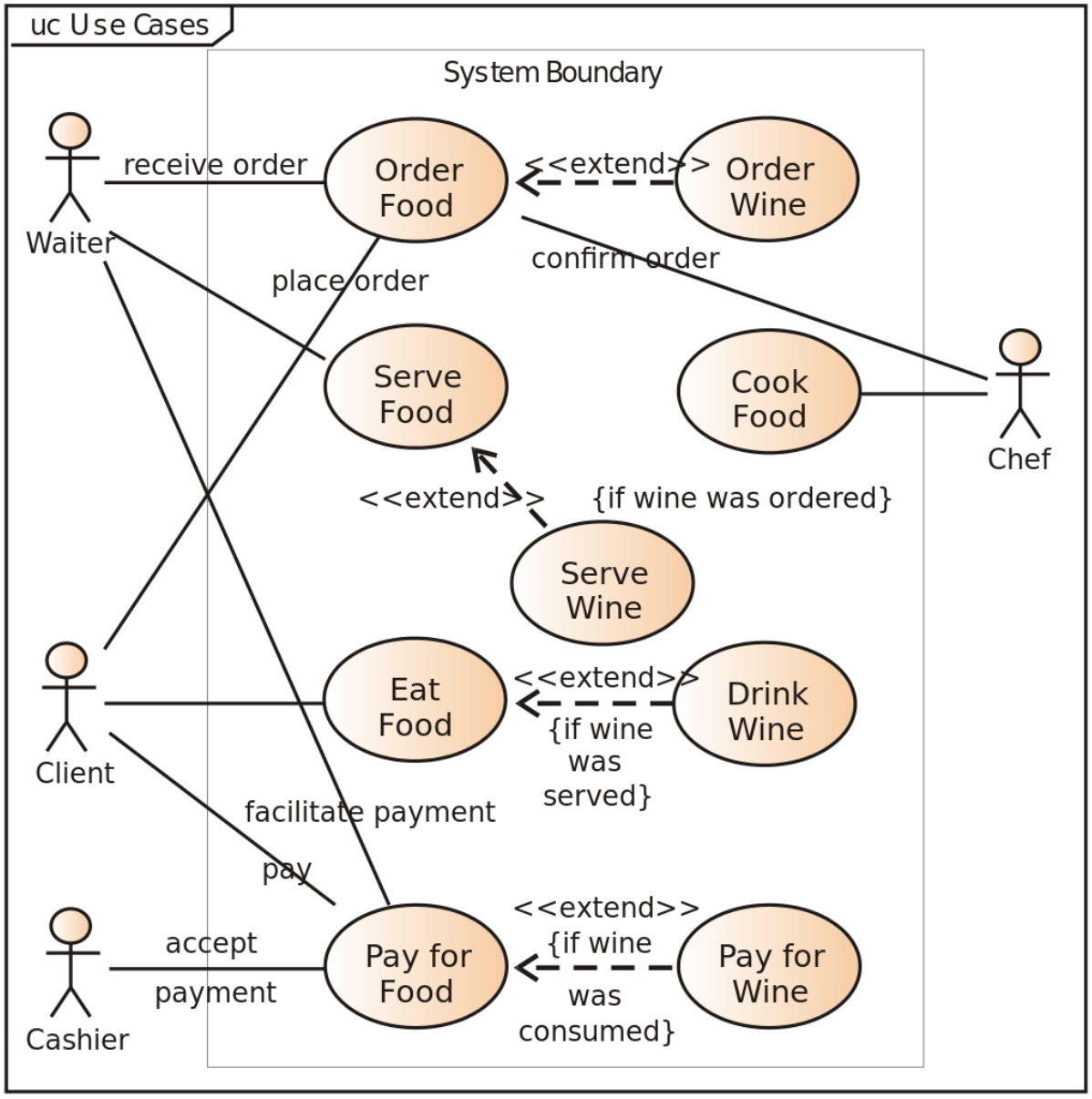


Диаграмма прецедентов (диаграмма вариантов использования) в UML — диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

- чётко отделить систему от её окружения;
- определить действующих лиц (актёров), их взаимодействие с системой и ожидаемую функциональность системы;
- определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть, прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются (для них составляется другой документ).

Техническое задание и технический проект

Техническое задание

Из лекций

Разработка

Нужно понимать, для кого и с какой целью.

ТЗ - результат проектирования, входной документ разработки.

Состав:

- Назначение объекта
- Технические характеристики
- Показатели качества
- Технико-экономические требования
- Требования к документации

Вопросы к ТЗ:

- В каких понятиях / на каком языке
- Степень детализации

ТЗ должно быть в терминах бизнеса, а не технологий или проектирования. Для деталей реализации следует использовать другой документ.

Степень детализации определяется следующими факторами:

- Заказчик может разобраться в ТЗ
- Можно оценить объём разработки

Структура

1. Общие сведения
2. Назначение и цели создания
3. Характеристики объекта автоматизации
4. Требования к системе
5. Состав и содержание работ по созданию системы
6. Порядок контроля и приёмки системы
7. Состав и содержание работ по вводу системы в действие
8. Требования к документированию

9. Источники разработки

https://ru.wikipedia.org/wiki/Техническое_задание

Техни́ческое зада́ние (ТЗ, техзада́ние) — документ, содержащий требования заказчика к объекту закупки, определяющие условия и порядок ее проведения для обеспечения государственных или муниципальных нужд, в соответствии с которым осуществляются поставка товара, выполнение работ, оказание услуг и их приемка. Это исходный документ, который учитывает основное назначение закупки товаров, работ, услуг, их характеристики, задание заказчика, описание первичных данных, целей и задач закупки, сроков поставки, выполнения работ, оказания услуг, требований к товару, работам, услугам, их результатам, к гарантиям, описание объекта закупки, объем закупаемых товаров, работ, услуг, формы отчетности, обоснование требований к товару, работам, услугам, эквивалентные показатели, экономические требования, а также специальные требования.

https://habrahabr.ru/post/300420/

Техническое задание — исходный документ на проектирование технического объекта (изделия). ТЗ устанавливает основное назначение разрабатываемого объекта, его технические характеристики, показатели качества и технико-экономические требования, предписание по выполнению необходимых стадий создания документации (конструкторской, технологической, программной и т. д.) и её состав, а также специальные требования. Техническое задание является юридическим документом — как приложение включается в договор между заказчиком и исполнителем на проведение проектных работ и является его основой: определяет порядок и условия работ, в том числе цель, задачи, принципы, ожидаемые результаты и сроки выполнения. То есть должны быть объективные критерии, по которым можно определить, сделан ли тот или иной пункт работ или нет. Все изменения, дополнения и уточнения формулировок ТЗ обязательно согласуются с заказчиком и им утверждаются. Это необходимо и потому, что в случае обнаружения в процессе решения проектной задачи неточностей или ошибочности исходных данных возникает необходимость определения степени вины каждой из сторон-участниц разработки, распределения понесенных в связи с этим убытков. Техническое задание, как термин в области информационных технологий – это юридически значимый документ, содержащий исчерпывающую информацию, необходимую для постановки задач исполнителям на разработку, внедрение или интеграцию программного продукта, информационной системы, сайта, портала либо прочего ИТ сервиса.

Нормативные документы

(http://tehpis.ru/services/razrabotka_tekhnicheskikh_zadaniy/chto-takoe-tekhnicheskoe-zadanie-i-kak-ego-razrabatyvat/)

Несмотря на всю свою важность, содержание ТЗ мало регламентировано нормативными документами. Требования к содержанию и порядку построения документа зависят от направления разработки и содержатся в следующих документах:

- ГОСТ Р 15.201-2000. Система разработки и постановки продукции на производство (СРПП). Продукция производственно-технического назначения.

Порядок разработки и постановки продукции на производство (приведены общие требования и краткие рекомендации по разработке).

- ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению (кратко изложено содержание ТЗ);
- ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы (достаточно подробно изложены состав и содержание ТЗ);
- ГОСТ 25123-82. Машины вычислительные и системы обработки данных. Техническое задание. Порядок построения, изложения и оформления (приведен порядок построения ТЗ).

Технический проект

(Из лекций.) Технический проект - совокупность технических документов, которые содержат окончательные проектные решения по изделию

https://ru.wikipedia.org/wiki/Технический_проект:

Технический проект — стадия создания автоматизированной системы[2].

В более узком смысле под техническим проектом понимается совокупность технических документов, которые содержат окончательные проектные решения по изделию (системе)[3][4].

Разработку технического проекта на автоматизированные системы осуществляют в соответствии с Комплексом стандартов на автоматизированные системы (ГОСТ 34 серии).

Работы по созданию (развитию) автоматизированной системы, выполняемые на стадии «Технический проект», регламентируются документом ГОСТ 34.601-90 и в общем случае содержат следующие этапы:

1. Разработка проектных решений по системе и её частям.
2. Разработка проектной документации на автоматизированную систему и её части.
3. Разработка и оформление документации на поставку изделий для комплектования автоматизированной системы и (или) технических требований (технических заданий) на их разработку.
4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации.

Перечень документов, создаваемых на стадии «Технический проект», определяется документом ГОСТ 34.201-89.

Требования к содержанию документов технического проекта приведены в руководящем документе по стандартизации РД 50-34.698-90.

17. Цели и предназначение технического задания. Структура технического задания. Основные разделы технического задания в соответствии с ГОСТом. Связь требований ГОСТ с индустриальной практикой

Разработка ТЗ

Делается с определенными целями и для определенных лиц

Цели для создания ТЗ:

1. Для внешнего исполнителя
2. Для организации тендера

ТЗ – (препод ссылается на определение из вики) исходный документ для разработки технического объекта

Что устанавливает ТЗ:

1. Назначение технического объекта (ТО)
2. Технические характеристики ТО
3. Показатели качества
4. Технико-экономические требования
5. Требования к документации
6. Спец. требования для конкретных объектов

Вопросы при составлении ТЗ:

1. В каких понятиях и на каком языке?
2. С какой степенью детализации (пример: надо ли прописывать структуру БД или нет?)

В каких понятиях и на каком языке? - В терминах бизнеса, а не техники.

Необходимо различать Техническое Задание с Техническим Проектом.

Технический проект может содержать техническое описание.

ТЗ – пишется в терминах бизнеса, но может включать дополнение в виде технического проекта. Но без нужды этим не надо грузить заказчика.

ТЗ – по-хорошему пишет аналитик (НЕТ ДЕТАЛЕЙ РЕАЛИЗАЦИИ)

Технический Проект – архитектор (ЗДЕСЬ ДЕТАЛИ РЕАЛИЗАЦИИ)

Степень детализации?

1. Необходимо чтобы было понятно для заказчика, что будет разработано
2. И чтобы была возможность по ТЗ оценить объем разработки

Структура технического задания

1. Общие сведения
2. Назначение и цели создания системы
3. Характеристика объекта автоматизации
4. Требования к системе
5. Состав и содержание работ по созданию системы
6. Порядок контроля и приемки системы
7. Состав и содержание работ по вводу системы в действие
8. Требования к документированию
9. Источники разработки

Общие сведения – какой-то идентификатор, название и т.д. Шифр темы и номер договора. Наименование и реквизиты заказчика и исполнителя. Перечень документов

на основании, которых создается система !полезно!. Плановые сроки начала и окончания. Сведения об источнике и порядке финансирования работ.
Общее отношении к стандарту: если работаем на гос. Структуру то все соблюдаем. Если обычный заказ, то вытаскиваем полезные детали.
Назначение и цели создания системы: система для того-то и того, и !ВАЖНО! в соответствии с требованиями настоящего ТЗ
Цели создания ТЗ - не уверены не пишите, если можете явно написать то пишите.
Характеристика объекта автоматизации – тот контекст, в котором предстоит работать в вашей системе. Включает:

- Краткие сведения об объекте автоматизации (например: инф. система клиники заказчика включает то-то, и персоналу приходится дублировать то-то, то-то) и ссылки на её документацию.
- Сведения об условиях эксплуатации системы !для ИТ почти неактуально!

Требования к системе !главный раздел! - изложение тех требований, на основе которых составляется ТЗ. Включает:

1. Требования к структуре и функционированию системы (функциональные требования)
2. Требования к численности персонала и их квалификации по работе
3. Требования к надежности
4. Требования к безопасности
5. Требования к эргономике и технической эстетики
6. Требования к транспортабельности
7. Требования к эксплуатации и техническому обслуживанию
8. Требования к защите информации от несанкционированных действий
9. Требования к сохранности информации при авариях
10. Требования к защите от влияния внешних воздействий
11. Требования к патентной чистоте
12. Требования к стандартизации и унификации
13. Требования к видам обеспечения:
    a. Математическое
    b. Информационное
    c. Лингвистическое
    d. И т.д.

Состав и содержание работ по созданию системы (сейчас в ТЗ обычно не пишется. Это есть либо в тексте договора, либо в календарном плане).
Пример календарного плана:
Номер Список работ Сроки Стоимость
Порядок контроля и приемки – что и как должно быть сделано, чтобы заказчик убедился, что система сделана. В ТЗ теперь это теперь тоже никогда не делается. Пишется в специальном документе в программе и методике испытания (ПМИ). Там, что необходимо сделать, чтобы систему развернуть, выполнить тест-кейсы и способ проверки, что она работает.
Пункты в ТЗ должны быть такими, чтобы можно было проверить.
Требования к составу и содержанию работ по вводу в действие:

1. Создание соответствующей инфраструктуры !это важно!

2. Приведение поступающей информации в структуру необходимой для системе !это важно и необходимо прописать, это по сути формат данных!
3. Кадровая подготовка

Требования к документированию

1. Объем и форма документации

Источники разработки !не очень понятно, формальный, о нем можно просто забыть!
Общие слова:
Требования должны быть понятными и проверяемыми.
Заказчик будет всегда пытаться манипулировать недосказанными требованиями.
По возможности писать короткими предложениями. Четко и коротко. Цель: избежание многозначности. Сюда же: не использовать многозначные термины. Избегать всяких оценочных слов.