1. Понятие технологии проектирования ИС, понятие информационной модели.

Осуществление проектирования информационных систем предполагает использование проектировщиком некоторых технологий, соответствующих масштабам и особенностям проекта.

Технология проектирования информационных систем (из конспекта) -- совокупность концептуальных методов и средств проектирования, а также методов и средств организации проектирования, т.е. управления процессом создания и модернизации ИС.

ТЗ ==========>| Методы |======> ИС
/Требования\ | |
\ к ИС / | Средства |

/|\
|
методы + средства ---------- управление IT-проектом
организации проектирования

Технология проектирования информационных систем – совокупность технологических операций в их последовательности и взаимосвязи, приводящая к разработке проекта системы, при этом определяется совокупность методов (методология) и средств, направленных не только непосредственно на проектирование ИС, но и на организацию, управление, внедрение и модернизацию проекта ИС, то есть обеспечение всего жизненного цикла системы.

Составные элементы технологии проектирования:

- методология;
- инструментальные средства;
- организация.

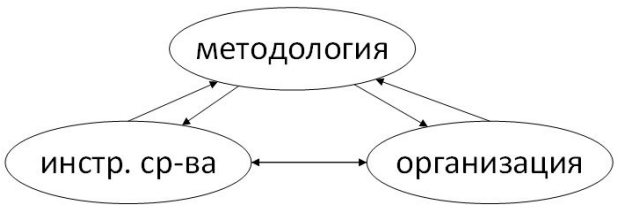
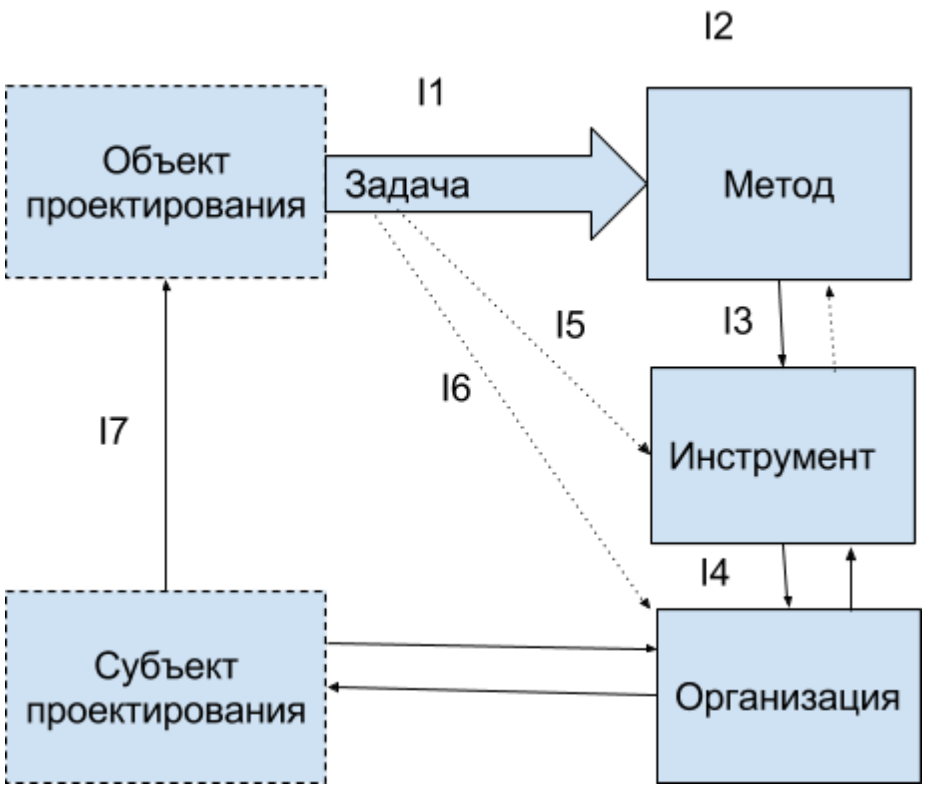
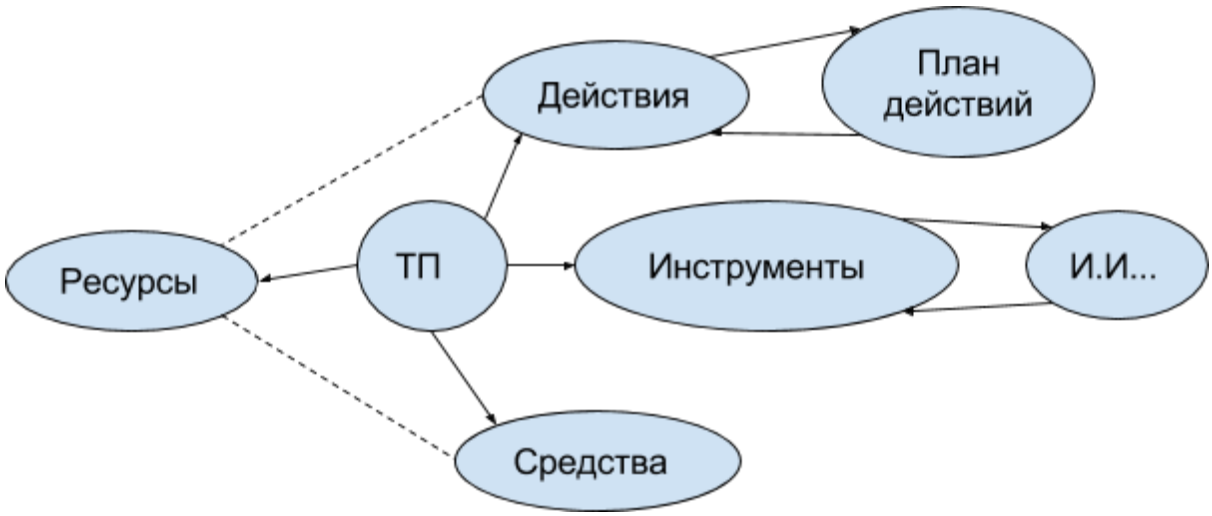


Схема решения задач в процессе проектирования ИС.



I -- исследовательское решение,
пунктиром обозначены вторичные связи.

В основе технологии проектирования лежит технологический процесс, который определяет действия , их последовательность, состав исполнителей, средства, ресурсы, требуемые для выполнения этих действий.



Что такое и.и… хз вообще)

Таким образом технология проектирования является регламентированной последовательностью технологических операций на основе некоторого метода и дает ответы на вопросы “что” и “как”, а далее “ в какой последовательности” и “кем”.

Информационная модель – Модель объекта, в которой представлены информационные аспекты моделируемого объекта. Является основой разработки моделей ИС.

Процесс построения информационных моделей с помощью формальных языков называется формализацией.

Обычно различают реальное (материальное, предметное) и мысленное (идеализированное, концептуально-методологическое) моделирование.

2. Подходы к проектированию и построению ИС

Есть 2 основных подхода: Локальный и Системный.

Сущность **локального подхода** к проектированию состоит в последовательном наращивании задач, решаемых в системе. В этом случае проектирование информационной системы состоит из решения задач, ориентированных на удовлетворение потребностей конкретных подразделений или требований, связанных с реализацией конкретных условий. При этом данные организовывают в отдельные логически структурированные файлы.

Этот подход имеет серьезные недостатки:

- избыточность информации;
- противоречивость;
- недостаточная скорость обработки;
- отсутствие гибкости;
- низкая стандартизация программного обеспечения.

Системный подход, будучи общей методологической базой проектирования информационных систем, основан на концепции интеграции данных, которые описывают все сферы деятельности объекта информатизации. Этот подход предусматривает рассмотрение всех элементов и составляющих процесса проектирования в их взаимосвязи, взаимозависимости и взаимном влиянии в интересах оптимального достижения как отдельных, так и общих целей создания информационной системы.

На современном этапе можно выделить четыре основных подхода к проектированию:

- **структурный подход** (функционально-ориентированное проектирование), который использует структурные методы для построения функциональной, информационной и других моделей информационной системы;
- **блочно-иерархический подход** к проектированию использует идеи декомпозиции сложных описаний объектов и соответственно средств их создания на иерархические уровни и аспекты, вводит понятие стиля проектирования (восходящее, нисходящее, смешанное), устанавливает связь между параметрами соседних иерархических уровней;
- **объектный подход** (объектно-ориентированное проектирование) предлагает набор объектных моделей для описания предметной области;
- **модельный подход** (модельно-ориентированное проектирование) основан на настройке и доработке типовой конфигурации информационной системы в среде специализированных инструментальных систем.

3. Понятие метода проектирования ИС. Классификация методов проектирования ИС

Методы проектирования ИС - использование определенных программных и аппаратных средств, составляющих средства проектирования.

Метод проектирования представляет собой совокупность трех компонент:

- Пошаговой процедуры, определяющей последовательность технологических операций проектирования;
- Критериев и правил, исп. Для оценки результатов выполнения технологических операций
- Нотации (графических и тестовых средств), используемых для описания проектируемой системы.

Классификация методов проектирования ИС:

1) По подходу к автоматизации объектов:

- **сверху вниз** - разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация. При нисходящем проектировании задача анализируется с целью определения возможности разбиения ее на ряд подзадач. Затем каждая из полученных подзадач также анализируется для возможного разбиения на подзадачи;
- **снизу вверх** - разработка подсистем (процедур, функций), в то время когда проработка общей схемы не закончилась, то есть разработка ведется от отдельных задач ко всей системе;
- **принципы «дуализма» и многокомпонентности** – заключается в сбалансированном сочетании двух предыдущих.

2) По степени автоматизации:

- **Ручного проектирования** (на бумаге, программирование на алгоритмических языках);
- **Компьютерного проектирования**(производится генерация или конфигурирование проектных решений на основе исп. Спец. Инструментальных программных средств(CASE-средства)).

3) По степени использования типовых проектных решений:

- **типового** проектирования, предполагающего конфигурацию ИС из готовых типовых проектных решений;
- **оригинального** (индивидуального) проектирования, когда проектные решения разрабатываются «с нуля» в соответствии с требованиями к ИС.

4) По степени адаптивности проектных решений методы:

- **реконструкции** - адаптация проектных решений выполняется путем переработки соответствующих компонентов (перепрограммирования программных модулей);
- **параметризации** - проектные решения настраиваются (перегенерируются) в соответствии с изменяемыми параметрами;
- **реструктуризациm** - перегенерация используемого набора проектных решений в соответствии с изменениями модели предметной области.

4. [НЕ ЗАКОНЧЕНО] Систем автоматизированного проектирования (САПР) технических средств и ИС: сходства, отличия, особенности, примеры.

Организационное обеспечение автоматизированного проектирования - совокупность документов устанавливающих состав проектной организации, межорганизационные связи, функции, формы представления результатов проектирования и порядок их согласования

САПр - комплекс средств автоматизации проектирования взаимосвязанных с необходимыми подразделениями проектирования и специалистами (проектировщиками)

Классификация САПр
ГОСТ 23501.108-85

Фасетный метод классификации XXXX-XXXX

1. Тип объекта (1-8)
2. Вид
3. Сложность (1-5)
4. Уровень автоматизации (1-3)
5. Комплексность автоматизации (1-3)
6. Характер документов (1-5) обычно ставится 4 или 5
7. Количество документов (1-3)
8. Количество уровней в архитектуре технического обслуживания (1-3)

Тип объектов

1. Машиностроение
2. Приборостроение
3. Технологические процессы машиностроения
4. Строительство
5. Технологические процессы в строительстве
6. Программные изделия

7. Организационные системы
8. Прочие

Вид объекта - **нет ничего, у кого что-то есть напишите**

Тут ничего не надо перечислять. ГОСТ говорит, что здесь -- действующий классификатор на объекты.

Сложность

1. Простые объекты (< 100 элементов)
2. Средней сложности (100-1000)
3. Сложные (1000-10000)
4. Высокой сложности(10000-1000000)
5. Сверхсложные(>1000000)

Уровень автоматизации

1. Низко автоматизированные <25%
2. Средне автоматизированные 25-50%
3. Высоко автоматизированные >50%

Комплексность автоматизации

1. Одноэтапно
2. Многоэтапно
3. Комплексно

Количество выпускаемых документов

1. САПр малой производительности <10000
2. САПр средней производительности 10000-1000000
3. САПр большой производительности >1000000

Характер документов

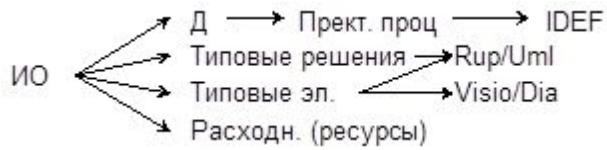
4. Комбинированные документы
5. Прочие

Число уровней в архитектуре

1. Одноуровневая
2. Двухуровневая
3. Трехуровневая

Информационное обеспечение САПр

представляет собой совокупность документов, описывающих стандартные процедуры, типовые решения, комплектующие элементы и другие ресурсы



док -> процесс проектирования
Типовые решения -> RUP/UML
Типовые элементы -> Visio/DIA
Расходное (Ресурсы)

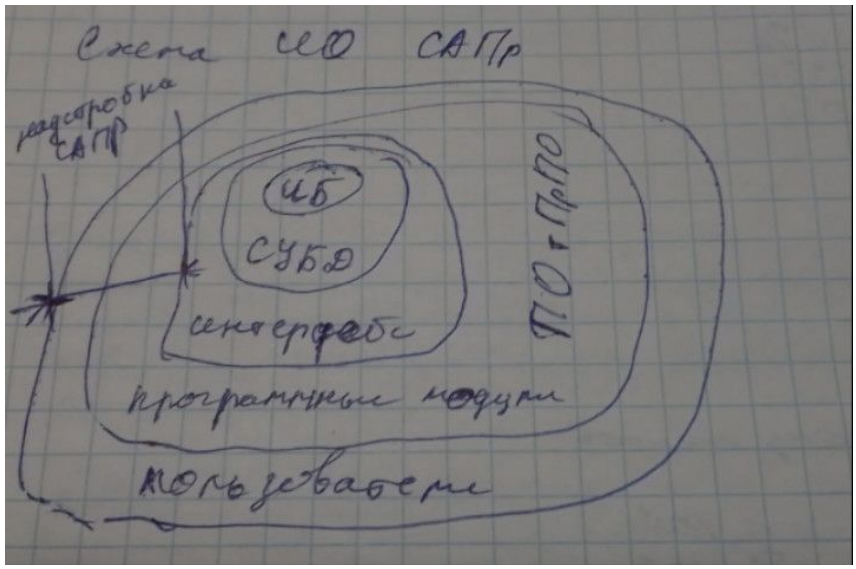
Основные требования к ИО САПр

- Наличие необходимой информации как для ручного так и для автоматизированного проектирования

- Возможности хранения и поиска информации, предоставление результатов процессов проектирования
- Объем хранимой информации и соответствующая его структура
- Быстродействие системы информационного обеспечения
- Возможность быстрого внесения изменений

При создании ИО САПр основная проблема заключается в преобразовании информации или консолидации.

Типовыми группами данных являются классификаторы и таблицы соответствий для них, а также расчетно-проектная информация (класс оперативной информации)

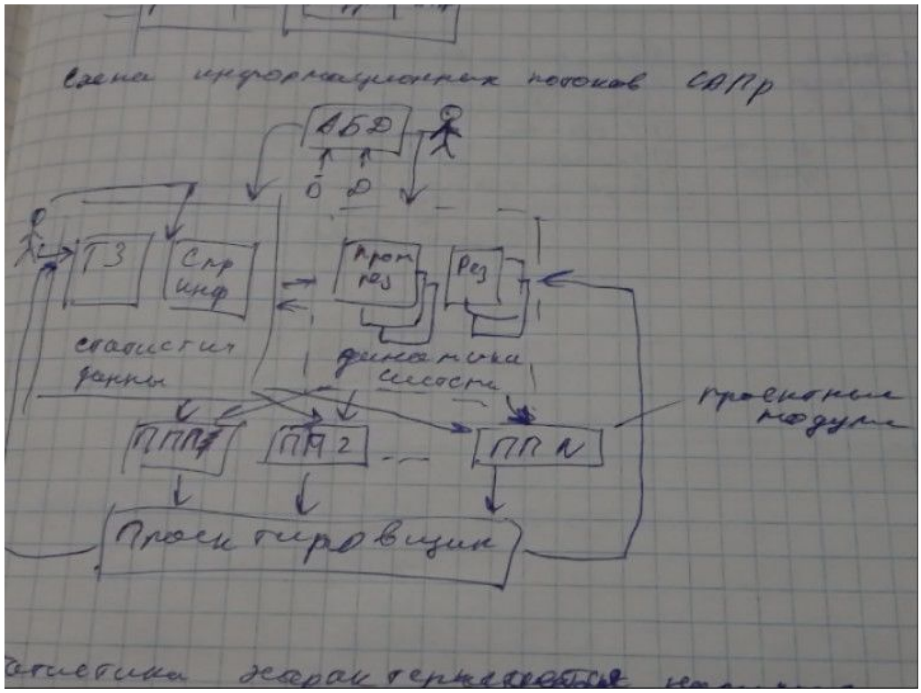


Основной проблемой построения ИО САПр является взаимодействие системы с проектными модулями. Это организуется посредством специального интерфейса.

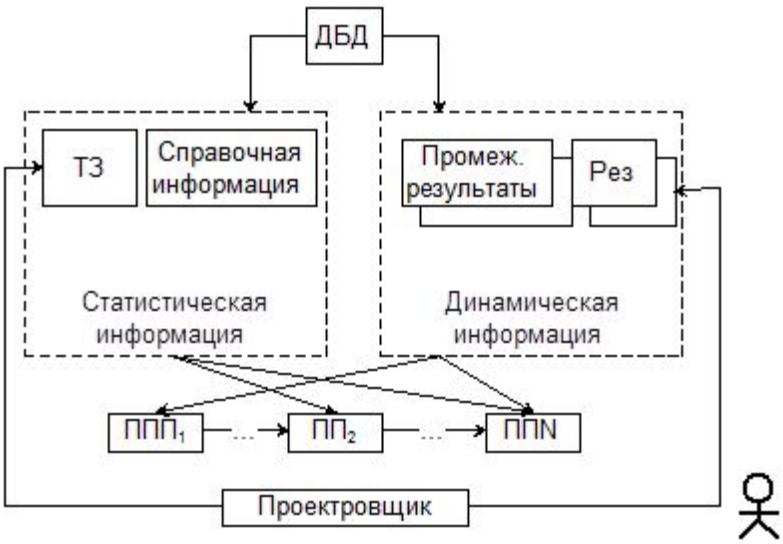
Проблемы:

- Согласование и сопряжение ИО САПр с модулями по форматам записей
- Согласование по программным средствам
- Согласование языков программирования
- Согласование кодов и обозначений данных

Сложность разработки БД для САПр обуславливается тем, что формирование ее структуры возможна только после разработки алгоритмов проектирования



Та норм фотка) только надо добавить,
ППП - проектные модули, прикладные программы
Не ДБД а АБД - администратор баз данных



Стрелочки к пунктирным блокам
Человечек -- составляет тз и справочную информацию
Пунктирные блоки между собой взаимосвязаны
Объяснение схемы из конспекта:

Статистика характеризуется наличием либо редко изменяемой информацией либо аналитической информацией, подстраивающейся под требования справочной информацией.

Администратор поддерживает контакт со службой проектной документации.

Динамическая информация состоит из данных, накапливаемых по мере выполнения операций проектирования.

Промежуточные данные постоянно изменяются, вносить изменения может только конструктор исполнитель и его руководитель.

Информация проект.:

1. Документальная (метаинформация). Например, аналоги объекта проектирования, методики проектирования.
2. Иконографическая.
3. Фотографическая (буквенные, числовые, справочные данные об объектах, а также данные для выполнения расчётов).

Основа БД -- фактографическая информация.

Существует два вида САПР -- банки данных и информационных систем.

5. Требования к технологиям проектирования ИС

Технология проектирования предполагает возможность выбора различных методов и средств на основе вариантного анализа.

1) Обеспечение создания ИС, отвечающей целям и задачам (заказчика)
2) Гарантированное создание системы с заданным качеством в заданное время в рамках заданных ресурсов
3) Поддержка сопровождения, модификации и наращивания системы
4) Обеспечение преемственности разработки
5) Обеспечение роста производительности труда проектировщика
6) Обеспечение надежности программного продукта и простота ведения проектной документации

Проблемы, порождаемые применением технологий проектирования на различных этапах ЖЦ (жизненного цикла)

1) Формирование требований к ИС

Проблема на 1: противоречивость требований

2) Разработка концепции
3) Разработка ТЗ

Проблемы на 2 и 3: отсутствие аналогов, целесообразность привлечения специалистов, необходимость системного подхода.

4) Эскизный проект
5) Технический проект

Проблемы на 4 и 5: целесообразность унификации этапов, стандартизация описания, противоречивость этапов

6) Рабочая документация

Проблема на 6: стандарты, проблема коммуникации между специалистами разных уровней с целью описания отдельных структур

7) Ввод в эксплуатацию(действие)
8) Сопровождение

Проблемы на 7 и 8: целесообразность модифицирования, сокращение количества натурных испытаний

6. Формализация описания технологии проектирования ИС

Технология проектирования в общем виде может быть формализована на основании теории множеств:

$$TD: <D<M^{D}, S^{D}>; <OD<M^{OD}, S^{OD}>; OR<M^{OR}, S^{OR}>>$$

Где D - проектирование (design), OD - организация проектирования, OR-модернизация и реинжениринг

M - методы,

S - средства.

D (проектирование) -> ТПИС

OD (организация проектирования) -> управление IT-проектами

OR (реинжиниринг и модернизация) -> НИР

7. Классификация технологий проектирования ИС. Выбор технологии проектирования ИС

Технология проектирования информационных систем – совокупность технологических операций в их последовательности и взаимосвязи, приводящая к разработке проекта системы, при этом определяется совокупность методов (методология) и средств, направленных не только непосредственно на проектирование ИС, но и на организацию, управление, внедрение и модернизацию проекта ИС, то есть обеспечение всего жизненного цикла системы.

Среди технологий проектирования ИС выделяют два основных класса:

1) каноническая технология;

2) индустриальная технология.

! Из конспекта:

Канонические технологии используются для уникального проектирования:

- На основании известных решений. При этом известное решение развивается, пока не достигнет уровня поставленной задачи.
- Когда решения начинаются с абсурдных постановок задачи.

Индустриальное проектирование ориентируется на использование САПР.

Технология канонического проектирования предполагает использование инструментальных средств универсальной компьютерной поддержки и предназначена для создания индивидуальных (оригинальных) проектов локальных ИС. При этом адаптация проектных решений возможна лишь путем перепрограммирования соответствующих программных модулей.

Организация канонического проектирования ИС ориентирована на использование главным образом каскадной модели жизненного цикла ИС. Стадии и этапы работы описаны в ГОСТ 34.601-90 [18].

Технологии индустриального проектирования используют инструментальные средства специальной компьютерной поддержки для разработки проектов сложных интегрированных (корпоративных) ИС.

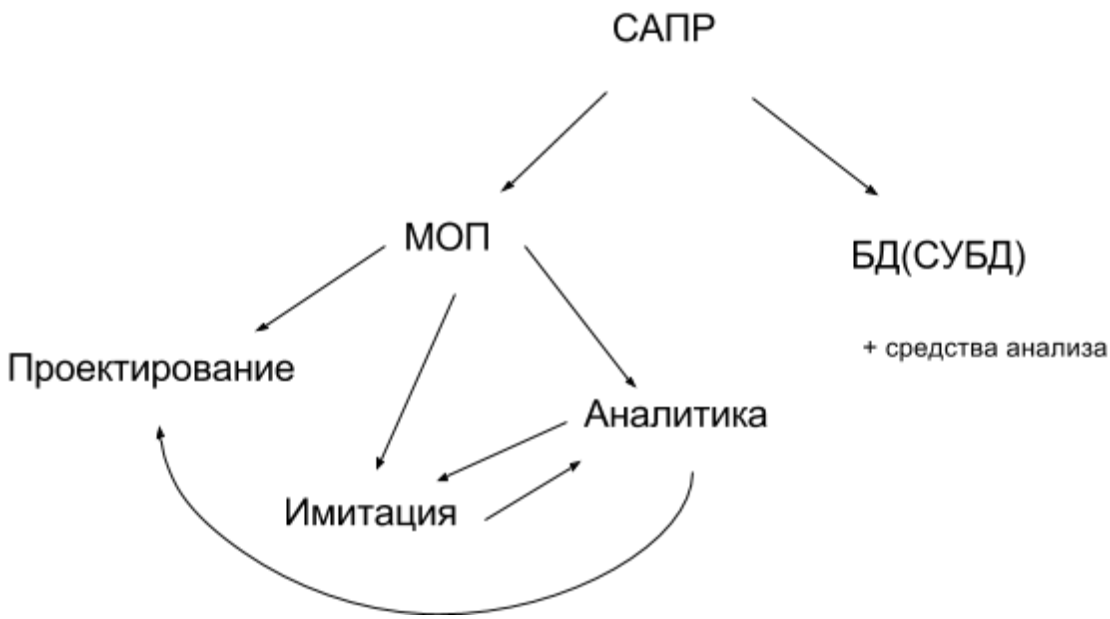
Индустриальная технология проектирования, в свою очередь, разбивается на два подкласса:

- типовое (параметрически-ориентированное или модельно-ориентированное) проектирование.
- автоматизированное проектирование (использование CASE-технологий).

8. Классификация средств проектирования ИС

Тут по идее не только САПР, хотя я не уверен

Картинка из конспекта, возможно из предыдущего вопроса:



Под ***средствами проектирования информационных систем*** (СП ИС) будем понимать комплекс инструментальных средств, обеспечивающих в рамках выбранной методологии проектирования поддержку полного жизненного цикла (ЖЦ) ИС. Каждый этап ЖЦ характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами.

Основные признаки средств проектирования ИС:

- Инвариантность к объекту проектирования
- Охват всех этапов жизненного цикла
- Техническая, программная и информационная совместимость средств

- Простота применения и обучения
- Экономическая целесообразность

Два класса средств проектирования:

- Стандарты, регламентирующие процесс создания, документация
- Компьютерные средства:
  - Проектирование операций обработки информации
    - § Алгоритмические языки
    - § Библиотеки
    - § Отладчики
    - § Тесты
  - Средства проектирования отдельных комплексов ИС
    - § Специальные пакеты мат.статистики
    - § Графические редакторы
    - § Редакторы текста
    - § СУБД
    - § Математического программирования
  - Средства автоматизированной (case) разработки этапов

! Мне кажется что тут нужна классификация из 8 вопроса

А вот это кажется то что надо

Традиционные средства проектирования

Использование технологии канонического проектирования предполагает применение различных средств на традиционных носителях, к которым принято относить:

- нормативно-правовые документы (положения о структурном подразделении, должностных инструкций и т.п.);
- нормативно-технические документы (стандарты, руководящие документы и т.п.);
- системы классификации и кодирования информации;
- системы документации;
- модели входных и выходных потоков информации и методики их анализа;
- др.

Средства автоматизации проектирования (CASE-средства проектирования)

При автоматизированном проектировании наряду с вышеназванными средствами могут быть использованы разнообразные программные средства, которые подразделяют на четыре группы:

- операционные средства (алгоритмические языки, макрогенераторы, генераторы программ типовых операций обработки данных, утилиты, библиотеки стандартных подпрограмм и классов объектов и др.);
- прикладные программные средства общего назначения (СУБД, текстовые и графические редакторы, табличные процессоры, метод ориентированные пакеты прикладных программ, оболочки экспертных систем, интегрированные пакеты прикладных программ);
- функциональные средства проектирования (функциональные пакеты прикладных программ, типовые проекты и проектные решения);
- средства автоматизации проектирования (CASE-средства).

Проектирование ИС с применением компьютерной поддержки, называется CASE – технологии проектирования. CASE – технологии применяются не только для

автоматизации проектирования ИС, но и для разработки моделей бизнес-процессов при проведении бизнес-анализа. CASE – технологии применяются в ситуациях, когда проблематика предметной области отличается большой сложностью.

Классификация по уровню проектирования в жизненном цикле создания ИС:

- средства верхнего уровня (Upper CASE) – анализ предметной области, определение места ИС в контуре бизнес-системы;
- средства среднего уровня (Middle CASE) – разработка архитектуры ИС, создание проектных спецификаций;
- средства нижнего уровня (Lower CASE) – поддержка разработки программного обеспечения.

Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ и включает следующие типы:

- средства анализа (соответствуют Upper CASE), предназначенные для построения и анализа моделей предметной области – (CA ERwin Process Modeler (бывший BPWin, затем AllFusion Process Modeler), Design/IDEF, ARIS, ORACLE Designer);
- средства анализа и проектирования (соответствуют Middle CASE), поддерживающие наиболее распространенные методологии проектирования, которые используются для создания проектных спецификаций, в частности проектных компонентов интерфейсов системы, архитектуры системы, алгоритмов и структур данных (Vantage Team Builder, Designer/2000, Silverran, PRO-4, CASE- аналитик, ARIS);
- средства проектирования баз данных (соответствуют Middle CASE), обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL – Structured Query Language – структурированном языке запросов) для наиболее распространенных СУБД: ER-win, S-Designer/2000, DataBase Designer, ARIS Tooleset;
- средства разработки приложений (соответствуют Lower CASE) -средства 4GL – Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000, (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi(Borland) и др. и генераторы кода, входящие в состав Vantage Team Builder, PRO-IV и частично – в Silverrun;
- средства реинжиниринга, обеспечивающие анализ программных кодов и схем БД и формирование на их основе различных моделей и проектных спецификаций: ER-win, Vantage.Team Builder, Silverran, PRO-4, ARIS Tooleset, Designer/2000 и т.д. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программы на языке C++: Rational Rose, Object Team.

Классификация по степени интегрированности выделяет:

- локальные CASE-средства – применяются для анализа системы и разработки автоматизированных рабочих мест, поддерживают 1, 2 типа моделей и методов (CASE-аналитик, Design/IDEF);
- малые интегрированные CASE-средства, используются для создания небольших ИС, поддерживают несколько типов моделей и методов (ER-win, BP-win, , Silverran);
- средние интегрированные CASE-средства – поддерживают от 4-15 моделей и методов. В этой категории Rational Rose, Designer/2000;
- крупные интегрированные CASE-средства, поддерживаются свыше 15 типов моделей и методов (семейство программных продуктов ARIS).

На сегодняшний день российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;
- Silverrun;
- ERwin+BPwin;
- Design/IDEF;
- S-Designor;
- CASE.Аналитик.

9. Основные стадии жизненного цикла проектирования ИС

Понятие жизненного цикла (ЖЦ) является одним из базовых понятий методологии проектирования информационных систем.

Модель ЖЦ, как правило, представляется в виде диаграммы ЖЦ, на которой изображаются пути перехода из некоторого начального состояния в конечное состояние и события, инициирующие изменения состояния. Модель ЖЦ может быть также представлена в виде текстового описания.

Структура ЖЦ по стандарту ГОСТ Р ИСО/МЭК 12207 базируется на трех группах процессов (каждый процесс включает ряд действий. Каждое действие включает ряд задач.):

- основные процессы ЖЦ ПО (приобретение, поставка, разработка, эксплуатация, сопровождение);
- вспомогательные процессы, обеспечивающих выполнение основных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем);
- организационные процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого ЖЦ, обучение).

Структура ЖЦ по стандарту ГОСТ Р ИСО/МЭК 15288-2005 базируется на пяти группах процессов:

- договорные процессы: приобретение (внутренние решения или решения внешнего поставщика); поставка (внутренние решения или решения внешнего поставщика);
- процессы предприятия: управление окружающей средой предприятия; инвестиционное управление; управление ЖЦ ИС; управление ресурсами; управление качеством;
- проектные процессы: планирование проекта; оценка проекта; контроль проекта; управление рисками; управление конфигурацией; управление информационными потоками; принятие решений;
- технические процессы, включающие: определение требований; анализ требований; разработка архитектуры; внедрение; интеграция; верификация; переход; аттестация; эксплуатация; сопровождение; утилизация;

- специальные процессы: определение и установка взаимосвязей исходя из задач и целей.

Структура ЖЦ по стандарту ГОСТ 34.601-90 предусматривает следующие стадии и этапы создания автоматизированной системы (АС) (8 групп процессов):

- **формирование требований к АС:** обследование объекта и обоснование необходимости создания АС; формирование требований пользователя к АС; оформление отчета о выполнении работ и заявки на разработку АС;
- **разработка концепции АС:** изучение объекта; проведение необходимых научно-исследовательских работ; разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователей; оформление отчета о проделанной работе;
- **техническое задание:** разработка и утверждение технического задания на создание АС;
- **эскизный проект:** разработка предварительных проектных решений по системе и ее частям; разработка документации на АС и ее части;
- **технический проект:** разработка проектных решений по системе и ее частям; разработка документации на АС и ее части; разработка и оформление документации на поставку комплектующих изделий; разработка заданий на проектирование в смежных частях проекта;
- **рабочая документация:** разработка рабочей документации на АС и ее части; разработка и адаптация программ;
- **ввод в действие:** подготовка объекта автоматизации; подготовка персонала; комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями); строительно-монтажные работы; пусконаладочные работы; проведение предварительных испытаний; проведение опытной эксплуатации; проведение приемочных испытаний;
- **сопровождение АС**: выполнение работ в соответствии с гарантийными обязательствами; послегарантийное обслуживание; эскизный, технический проекты и рабочая документация – это последовательное построение все более точных проектных решений.

Допускается исключать стадию «Эскизный проект» и отдельные этапы работ на всех стадиях, объединять стадии «Технический проект» и «Рабочая документация» в «Технорабочий проект», параллельно выполнять различные этапы и работы, включать дополнительные.

Замечание: *стандарт ГОСТ 34.601-90 не вполне подходит для проведения разработок в настоящее время: многие процессы отражены недостаточно, а некоторые положения устарели.*

10. Модели жизненного цикла ИС: основные и модифицированные

Технология проектирования задаёт последовательность действий, необходимых для получения проекта ИС. Очевидно, что выполнение каждого из таких действий означает переход ИС из состояния в состояние.
Два подхода моделирования:

- Стохастический (дискретные модели Маркова, непрерывные модели Маркова)
- Встроенный (предположительно в жизненный цикл)

Из модели жизненного цикла можно определить задачи, состав и последовательность работ, получаемые результаты на каждом этапе, методы и средства, наиболее эффективные на каждом этапе, роли и ответственность субъектов. Всё это будет означать описание выбранной комплексной технологии проектирования.

1) Основные

- **Модель кодирования и устранения ошибок** («проб и ошибок»)

Эта модель не актуальна при профессиональной разработке программного обеспечения. Модель Build-and-fix хороша для небольших проектов, которые не требуют сопровождения, но абсолютно непригодна для корпоративных или вообще нетривиальных проектов объемом более 1000 строк.

- **Каскадная модель** («водопад» – waterfall)

Каскадная стратегия подразумевает линейную последовательность прохождения стадий создания информационной системы. Другими словами, переход с одной стадии на следующую, происходит только после того, как будет полностью завершена работа на текущей.

Данная модель применяется при разработке информационных систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования.

***Достоинства* модели:**

− на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;

− выполняемые в четкой последовательности стадии позволяют уверенно планировать сроки выполнения работ и соответствующие ресурсы.

***Недостатки* модели:**

− реальный процесс разработки информационной системы редко полностью укладывается в такую жесткую схему. Особенно это относится к разработке нетиповых и новаторских систем;

− жизненный цикл основан на точной формулировке исходных требований к информационной системе. Реально в начале проекта требования заказчика определены лишь частично;

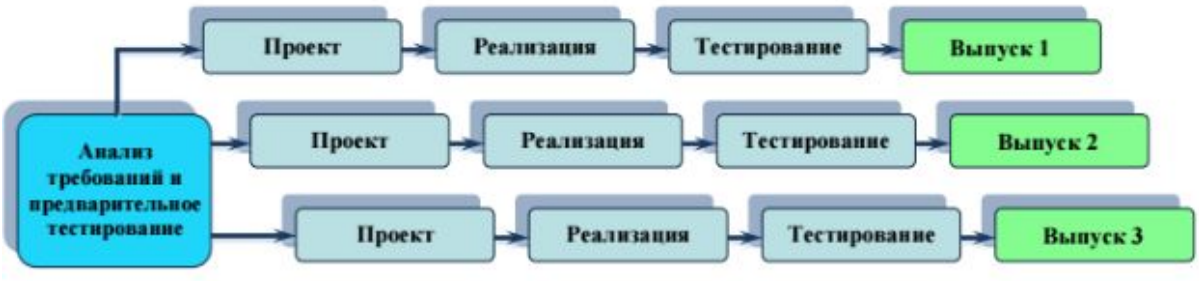
− основной недостаток – результаты разработки доступны заказчику только в конце проекта. В случае неточного изложения требований или их изменения в течение длительного периода создания ИС заказчик получает систему, не удовлетворяющую его потребностям.

- **V-образная (шарнирная) модель**

V-образная модель дала возможность значительно повысить качество ПО за счет своей ориентации на тестирование, а также во многом разрешила проблему соответствия созданного продукта выдвигаемым требованиям благодаря процедурам верификации и аттестации на ранних стадиях разработки (пунктирные линии на рисунке указывают на зависимость этапов).
Однако в целом V-образная модель является всего лишь модификацией каскадной и обладает многими ее недостатками. В частности, и та и другая слабо приспособлены к возможным изменениям требований заказчика. Если процесс разработки занимает продолжительное время, то полученный в результате продукт может оказаться фактически ненужным заказчику, поскольку его потребности существенно изменились.

- **Инкрементная модель** (поступательная, многопроходная)

Инкрементная разработка представляет собой процесс частичной реализации всей системы и постепенного наращивания функциональных возможностей. В начале работы над проектом определяются все основные требования к системе, после чего выполняется ее разработка в виде последовательности версий. При этом каждая версия является законченным и работоспособным продуктом. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т.д., пока не будет получена полная система.

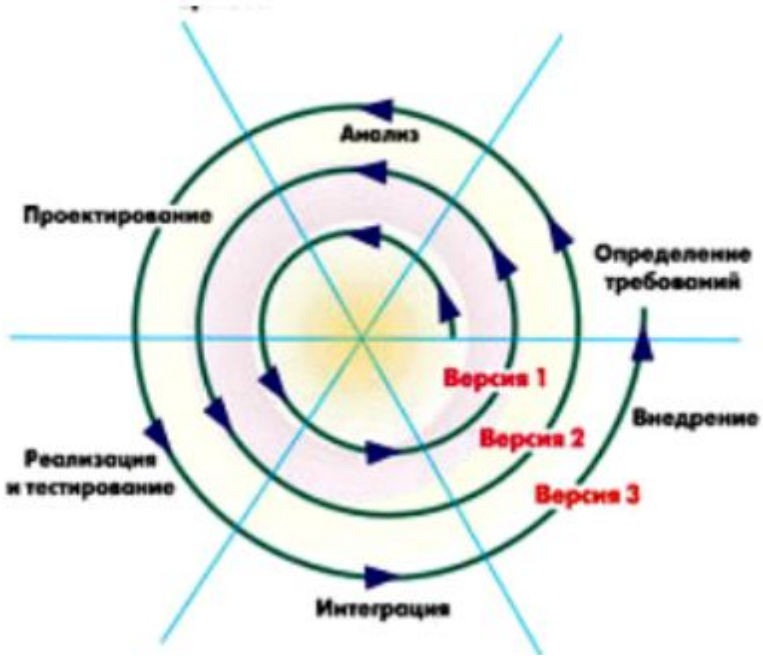


Данная модель жизненного цикла характерна при разработке сложных и комплексных систем, для которых имеется четкое видение того, что собой должен представлять конечный результат. Разработка версиями ведется в силу разного рода причин:
− отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;
− отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;
− требований поэтапного внедрения и освоения продукта конечными пользователями. Внедрение всей системы сразу может вызвать у ее пользователей неприятие и только «затормозить» процесс перехода на новые технологии.
Достоинства и недостатки этой стратегии такие же, как и у классической. Но в отличие от классической стратегии заказчик может раньше увидеть результаты. Уже по результатам разработки и внедрения первой версии он может незначительно изменить требования к разработке, отказаться от нее или предложить разработку более совершенного продукта с заключением нового договора.

- **Спиральная (эволюционная, итерационная) модель**

Спиральная стратегия подразумевает разработку в виде последовательности версий, но в начале проекта определены не все требования. Требования уточняются в результате разработки версий.
В начале работы над проектом у заказчика и разработчика нет четкого видения итогового продукта (требования не могут быть четко определены) или стопроцентной уверенности в успешной реализации проекта (риски очень велики). В связи с этим

принимается решение разработки системы по частям с возможностью изменения требований или отказа от ее дальнейшего развития.



Достоинства модели:

− позволяет быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований;

− допускает изменение требований при разработке информационной системы, что характерно для большинства разработок, в том числе и типовых;

− обеспечивает большую гибкость в управлении проектом;

− позволяет получить более надежную и устойчивую систему. По мере развития системы ошибки и слабые места обнаруживаются и исправляются на каждой итерации;

− позволяет совершенствовать процесс разработки – анализ, проводимый в каждой итерации, позволяет проводить оценку того, что должно быть изменено в организации разработки, и улучшить ее на следующей итерации;

− уменьшаются риски заказчика. Заказчик может с минимальными для себя финансовыми потерями завершить развитие неперспективного проекта.

Недостатки модели:

− увеличивается неопределенность у разработчика в перспективах развития проекта. Этот недостаток вытекает из предыдущего достоинства модели;

− затруднены операции временного и ресурсного планирования всего проекта в целом. Для решения этой проблемы необходимо ввести временные ограничения на каждую из стадий жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа выполнена. План составляется на основе статистических данных, полученных в предыдущих проектах и личного опыта разработчиков.

2) Модифицированные

- Поэтапная модель с промежуточным контролем («водоворот»)

Это доработка каскадной модели с учетом взаимозависимости этапов и необходимости возврата на предыдущие ступени, что может быть вызвано, например, неполнотой требований или ошибками в формировании задания.

Результатом каждой фазы является один или несколько документов, которые утверждаются. Следующая фаза не должна начинаться до того, пока предыдущая не завершена. У фаз имеются определенные совпадения, и информация передается от одной фазы к другой.

Недостаток: модель больше подходит для традиционных видов инженерной деятельности, чем для разработки ПО. В частности, одной из самых больших проблем оказалась ее «предрасположенность» к возможным несоответствиям полученного в результате продукта и требований, которые к нему предъявлялись. Основная причина этого заключается в том, что полностью сформированный продукт появляется лишь на поздних этапах разработки, но так как работу на разных этапах обычно выполняли различные специалисты и проект передавался от одной группы к другой, то по принципу испорченного телефона оказывалось так, что на выходе получалось не совсем то, что предполагалось вначале.

- Структурная эволюционная модель быстрого прототипирования (Rapid Prototype Model)

Прототипирование – это процесс построения рабочей модели системы. Прототип – это первоначальная версия системы, которая используются для апробирования возможностей дизайна и демонстрирования идей. Прототипы можно использовать на различных фазах разработки.

Жизненный цикл разработки программного продукта начинается с разработки плана проекта, затем выполняется быстрый анализ, после чего создаются база данных, пользовательский интерфейс и выполняется разработка необходимых функций. В результате этой работы получается документ, содержащий частичную спецификацию требований к программному продукту. Данный документ в дальнейшем является основой для итерационного цикла быстрого прототипирования.

Достоинства модели:
− пользователь может получить прототип системы;
− сведение к минимуму количества неточностей в требованиях;
− нечувствительна к изменениям требований;
− формальная спецификация в рабочей модели;
− гибкое проектирование и разработка;
− минимизация разногласий с заказчиком;
− управление рисками;
− участие заказчика в проектировании и реализации.
Недостатки модели:
− неадекватность прототипов;
− недостаток документации;
− недостаточное внимание качеству;
− заказчик может оставить себе прототип;
− зацикливание прототипа;
− может быть большое количество итераций;
− процесс без надлежащего контроля может быть длительным;
− не используются методы SADT – минимум документации.

- Модель быстрой разработки приложений RAD (Rapid Application Development)

Является одним из подходов к разработке ПО в рамках спиральной модели ЖЦ. Данная модель включает в себя три составляющие:
− 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.

Рис.1.9. Модель быстрой разработки приложений RAD.

Жизненный цикл ПО по методологии RAD состоит из четырёх фаз:

− анализа и планирования требований. Пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь. Формулирование требований к системе выполняется пользователями под руководством специалистов-разработчиков. Ограничивается масштаб проекта, устанавливаются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации проекта в заданных размерах финансирования, на имеющихся аппаратных средствах и т.п.

− проектирования. - Часть пользователей принимают участие в техническом проектировании системы под руководством специалистов разработчиков, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. При необходимости корректируется функциональная модель, создаются частичные прототипы: экранов, отчетов, устраняющие неясности или неоднозначности. Устанавливаются требования разграничения доступа к данным. На этой же фазе происходит определение необходимой документации. Оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении системы на подсистемы.

− построения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется в процессе разработки.

− внедрения. Производятся обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой.

Хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Она неприменима для разработки операционных систем; сложных расчетных программ с большим объемом программного кода.

Основные принципы технологии RAD:

− разработка приложений итерациями;

− необязательность полного завершения работ на каждом этапе ЖЦ;

− обязательное вовлечение пользователей на этапе разработки;

− использование прототипирования, позволяющего выяснить и удовлетворить все требования конечного пользователя;
− тестирование и развитие проекта одновременно с разработкой;
− грамотное руководство разработкой, четкое планирование и контроль выполнения работ.

- Спиральная модель «Win-Win»

Спиральная модель «Win-Win» содержит в себе больше фаз, в которых внимание сконцентрировано на участии заказчика в процессе разработки.
В этом методе, основанном на постоянном согласовании, циклы состоят из следующих **фаз или стадий:**
− определение участников следующего уровня;
− определение условий, необходимых для одержания участниками победы;
− согласование «победных» условий;
− формулирование целей, ограничений и альтернативных вариантов следующего уровня;
− оценка альтернативных вариантов на уровне продукта и процесса, разрешение рисков;
− определение следующего уровня продукта и процесса, включая сегментацию;
− обоснование определений продукта и процесса;
− обзор и комментарии.
Важной стадией является последующее планирование следующего цикла и обновление плана жизненного цикла, включая разделение системы на подсистемы. Эта стадия может включать в себя план прекращения проекта, если продолжение работы является слишком рискованным или невозможным.
Спиральная модель «Win-Win» имеет следующие **преимущества**:
− более быстрая разработка ПО благодаря содействию, оказываемому участниками проекта;
− уменьшение стоимости программ благодаря уменьшению объема переработок и текущего сопровождения;
− более высокий уровень удовлетворения со стороны участников проекта, достигаемого до разработки самого продукта;
− более высокое качество ПО благодаря использованию компромиссных качественно-атрибутивных моделей на уровне архитектуры;
− исследование большого количества вариантов построения архитектуры на ранних этапах разработки.

- Модель ROP (Rational Objectory Process) методологии RUP (Rational Unified Process)

Rational Objectory Process – итеративный процесс, в течение которого происходит последовательное уточнение результатов. Направлен именно на создание моделей, а не на разработку каких-либо других элементов проекта (например, текстовых документов).
RUP – одна из спиральных методологий разработки программного обеспечения.
ROP разбит на циклы, каждый из которых, в свою очередь, состоит из четырех фаз:
− начальная стадия (Inception);
− разработка (Elaboration);
− конструирование (Construction);
− ввод в эксплуатацию (Transition).

Результатом работы каждого такого цикла является своя версия программной системы.
В этом смысле RUP является реализацией спиральной модели, хотя довольно часто изображается в виде графика-таблицы. Диаграмма имеет два измерения: горизонтальная ось представляет время и показывает временные аспекты жизненного цикла процесса; вертикальная ось представляет дисциплины, которые определяют физическую структуру процесса. Видно, как с течением времени изменяются акценты в проекте (на ранних итерациях больше времени отводится требованиям).

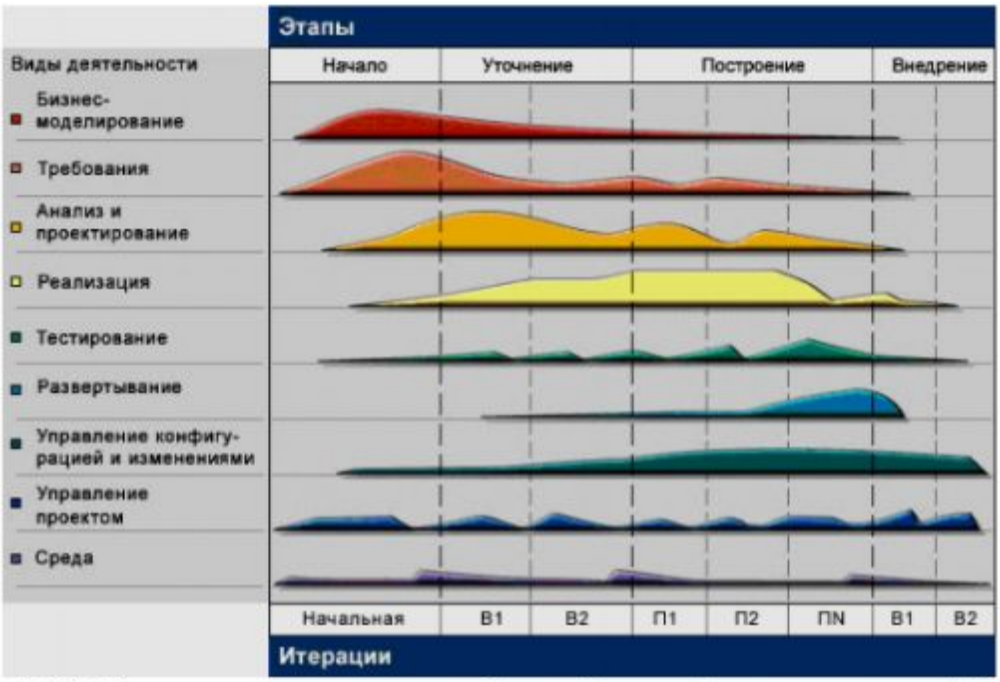


Рис.1.10. Модель жизненного цикла быстрой разработки приложений RUP.

11. [хз, оно ли, честно спи… позаимствовано у ГИС] Соответствие информационных процессов, информационных технологий и средств их проектирования. Принципы проектирования базовых информационных технологий.

В настоящее время уже давно сформировались идеология и практика применения различных средств извлечения, передачи, хранения, обработки и представления информации. Однако разрозненное их применение или использование в ограниченной совокупности не позволяло до сих пор получить значительный системный эффект. Необходим подход к информационным технологиям как к системе. Такой подход является обоснованным ввиду того, что информационная технология обладает единой целью, а именно: необходимостью формирования информационного ресурса в обществе, имеет сопрягаемые взаимодействующие средства ее реализации, характеризуется тенденцией развития в связи с интенсивным обновлением средств вычислительной техники и техники связи. Анализ информационных технологий как системы должен выполняться на основе дескриптивного определения, разработка информационных технологий должна

базироваться на конструктивном подходе. Такой подход предполагает необходимость возникновения проблемной ситуации для разработки системы. Можно считать, что возникающая проблема порождает будущую систему.

Структура системы описывается на концептуальном, логическом и физическом уровнях:

– концептуальный уровень позволяет качественно определить основные подсистемы, элементы и связи между ними;

– на логическом уровне могут быть сформированы модели, описывающие структуру отдельных подсистем и взаимодействия между ними;

– физический уровень означает реализацию структуры на известных программно-аппаратных средствах.

При использовании информационных технологий в системном аспекте необходимо соблюдать следующие принципы.

1. Наличие сформулированной единой цели у информационных технологий в рамках разрабатываемой системы.

2. Согласование информационных технологий по входам и выходам с окружающей средой.

3. Типизация структур информационных технологий.

4. Стандартизация и взаимная увязка средств информационной технологии.

5. Открытость информационных технологий как системы.

Базовые принципы разработки информационных технологий

В качестве базовых принципов выделим следующие:

Принцип иерархического упорядочивания - является принципом решения трудных проблем путем разбиения их на множество меньших независимых задач и рассмотрении их в виде компонентов единой системы. Процесс описания проблемы, в этом случае, представляет собой разработку древовидной иерархической структуры, каждый новый уровень которой детализирует задачи, описанные на предыдущем уровне. Например, проблема получения цветного фотоснимка может быть представлена в виде иерархии подзадач.

Принцип абстрагирования - заключается в выделении существенных с некоторых позиций аспектов проблемы и отвлечение от несущественных с цел

Принцип формализации - заключается в необходимости строгого методического подхода к решению проблемы.

Принцип концептуальной общности - заключается в следовании единой системе обозначений и общих принципов решения проблемы на всех стадия проектирования и эксплуатации технологии.ью представления системы в простом общем вид

Принцип полноты - заключается в контроле избыточности и исключении лишних элементов из технологической системы.

Принцип непротиворечивости - заключается в обоснованности и согласованности элементов.

Принцип логической независимости - заключается в независимости логических информационных структур от технической реализации системы.

12. Классификация стандартов на проектирование и разработку информационных систем.

Реальное применение любой технологии проектирования, разработки и сопровождения ИС в конкретной организации и конкретном проекте происходит при соблюдении всеми участниками проекта правил и соглашений, касающихся:

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

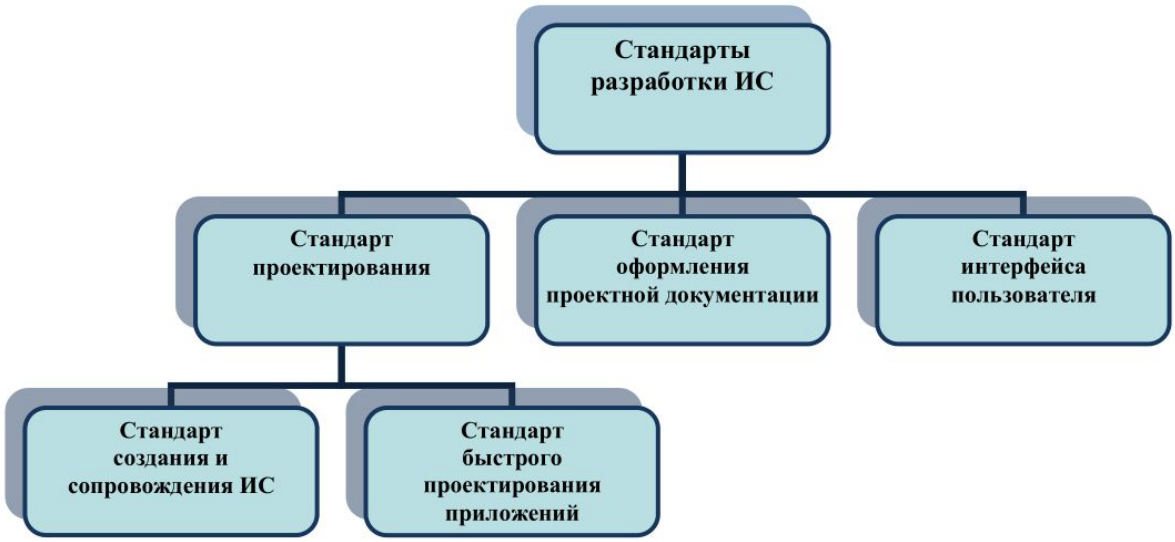


Рисунок – Стандарты проектирования информационных систем

Стандарт проектирования должен устанавливать:

- набор необходимых моделей (диаграмм) на каждой стадии проектирования и степень их детализации;
- правила фиксации проектных решений на диаграммах, в том числе: правила именования объектов (включая соглашения по терминологии), набор атрибутов для всех объектов и правила их заполнения на каждой стадии, правила оформления диаграмм, включая требования к форме и размерам объектов, и т.д.;
- требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы, настройки CASE-средств, общие настройки проекта и т.д.;
- механизм обеспечения совместной работы над проектом, в том числе: правила интеграции подсистем проекта, правила поддержания проекта в одинаковом для всех разработчиков состоянии (регламент обмена проектной информацией,

механизм фиксации общих объектов и т.д.), правила проверки проектных решений на непротиворечивость и т.д.

Стандарт оформления проектной документации должен устанавливать:

- комплектность, состав и структуру документации на каждой стадии проектирования;
- требования к ее оформлению (включая требования к содержанию разделов, подразделов, пунктов, таблиц и т.д.);
- правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков для каждой стадии;
- требования к настройке издательской системы, используемой в качестве встроенного средства подготовки документации;
- требования к настройке CASE-средств для обеспечения подготовки документации в соответствии с установленными требованиями.

Стандарты в области информационных технологий можно классифицировать:

- в зависимости от организации, утверждающей стандарты;
- по уровню утверждающей организации;
- по предмету стандартизации;
- по используемым методическим источникам.

В зависимости от *организации, утверждающей стандарты*:

- официальные международные, официальные национальные или национальные ведомственные стандарты (например, стандарты ISO1, IEC2, ГОСТ);
- стандарты международных консорциумов и комитетов по стандартизации (например, стандарты OMG – Object Management Group (Группа по управлению объектами));
- стандарты «де-факто» – официально никем не утвержденные, но фактически действующие (например, стандартом «де-факто» долгое время были язык взаимодействия с реляционными базами данных SQL4);
- фирменные стандарты (например, Microsoft ODBC5, методика Oracle CDM).

Объектами стандартизации при проектировании информационных систем являются:

- процесс (что делать?): ISO12207, ISO15288, IEEE1220;
- практика (как следует делать?): ISO15504;
- источник (какие данные использовать): ISO10303;
- описание: IDEF, UML, IEEE 1471.

Стандартизация информационных технологий и систем повышает их прибыльность за счет снижения затрат на создание и особенно модификацию.

13. Международные стандарты ISO/IEC 12207:1995-08-01 (ГОСТ Р ИСО/МЭК 12207), стандарт ISO/IEC 15288:2002 (ГОСТ Р ИСО/МЭК 15288-2005), комплекс стандартов ГОСТ 34.

Международный стандарт: ISO/IEC 12207 Information technology - Software life cycle processes (Информационные технологии. Процессы жизненного цикла программного обеспечения).

Российский аналог: ГОСТ Р ИСО/МЭК 12207 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств».

Базируясь на процессном подходе, ISO 12207 определяет необходимость документирования основных результатов процесса, но не ограничивает их содержание и тем более последовательность, а также не противоречит применению итераций в разработке.

Эта картинка ни под один ГОСТ не подходит(см вопрос 9)

В РФ был разработан и принят идентичный ISO 12207 стандарт **ГОСТ Р ИСО/МЭК 12207**. Стандарт предполагает, что процессы состоят из работ, для которых определены задачи (а также цели и результаты). Тем не менее, допускается адаптация процессов к особенностям организации (например, при больших масштабах проекта изменять состав определенных задач или работ). Как правило, это возможно сделать в рамках существующих на предприятии процессов.

ISO/IEC 15288

Международный стандарт: ISO/IEC 15288 Systems engineering. System life cycle processes (Системотехника. Процессы жизненного цикла системы).

Российский аналог: ГОСТ Р ИСО/МЭК 15288 Информационная технология. Системная инженерия. Процессы жизненного цикла систем.

Достаточно «молодой» стандарт системной инженерии (впервые представленный в 2002 году), ISO/IEC 15288 фокусируется на вопросах жизненного цикла системного уровня, в особенности тейлоринге (tailoring) - по сути, настройке и адаптации ЖЦ к конкретным требованиям и ограничениям.

В отличие от рассмотренного ранее стандарта, ISO 15288 распространяется на системы в целом, охватывая такие их элементы, как: «технические средства, программные средства, люди, процессы (например, процесс оценки), процедуры (например, инструкции оператора), основные средства и природные ресурсы (например, вода, объекты живой природы, минералы)»

Согласно данному стандарту, любой процесс ЖЦ может быть начат в любой момент, без ограничения порядка использования и последовательности (в том числе, параллельном выполнении нескольких процессов).

Важно также отметить высокий уровень абстракции ISO 15288 в сравнении с ISO 12207, так как данный стандарт не приводит ролей, конечных результатов в виде списка выходных документов, либо же состава работ, лишь оставаясь на уровне концепции.

Гост 34

Группа 0	ГОСТ 34.003-90	Автоматизированные системы. Термины и определения
Группа 2	ГОСТ 34.201-89	Виды, комплектность, обозначения документов при создании АС
Группа 6	ГОСТ 34.601-90	Стадии создания АС
	ГОСТ 34.602-89	ТЗ на создание АС
Руководящий документ	РД 50-34.698-90	Требования к содержанию документов

По ГОСТ 34.601-90

1.ФТ - Формирование требований к АС.	1.1. Обследование объекта и обоснование необходимости создания АС 1.2. Формирование требований пользователя к АС 1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания)
2. РК - Разработка концепции АС.	2.1. Изучение объекта 2.2. Проведение необходимых научно-исследовательских работ 2.3. Разработка вариантов концепции АС удовлетворяющей требованиям пользователя 2.4. Оформление отчета о выполненной работе
3. ТЗ - Техническое задание АС.	3.1. Разработка и утверждение технического задания на задание
4. ЭП - Эскизный проект.	4.1. Разработка предварительных проектных решений по системе и ее частям 4.2. Разработка документации на АС и ее части
5. ТП - Технический проект.	5.1. Разработка проектных решений по системе и ее частям 5.2. Разработка документации на АС и ее части 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и/или технических требований (технических заданий) на их разработку 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации
6. РД - Рабочая документация.	6.1. Разработка рабочей документации на систему и ее части 6.2. Разработка или адаптация программ
7. ВД - Ввод в действие.	7.1. Подготовка объекта автоматизации к вводу АС в действие; 7.2.Подготовка персонала 7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)7.4.Строительно-монтажные работы 7.5.Пуско-наладочные работы 7.6.Проведение предварительных испытаний7.7.Проведение опытной эксплуатации 7.8. Проведение приемочных испытаний
8.Сп - Сопровождение АС.	8.1. Выполнение работ в соответствии с гарантийными обязательствами; 8.2. Послегарантийное обслуживание.

14. Фирменные стандарты (Oracle CDM, MSF и др.)

Методология Oracle (Oracle Method) – комплекс методов, охватывающий большинство процессов ЖЦ ПО [40]. В состав комплекса входят:
− CDM (Custom Development Method) – разработка прикладного ПО;
− PJM (Project Management Method) – управление проектом;
− AIM (Application Implementation Method) – внедрение прикладного ПО;
− BPR (Business Process Reengineering) – реинжиниринг бизнес-процессов;
− OCM (Organizational Change Management) – управление изменениями, и др.
Custom Development Method (методика Oracle) по разработке прикладных информационных систем – технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle.
В соответствии с этими факторами в CDM выделяются два основных подхода к разработке:
− классический подход (Classic);
− подход быстрой разработки (Fast Track).
Классический подход (Classic) применяется для наиболее сложных и масштабных проектов, он предусматривает последовательный и детерминированный порядок выполнения задач. Для таких проектов характерно большое количество реализуемых бизнес-правил, распределенная архитектура, критичность приложения. Применение классического подхода также рекомендуется при нехватке опыта у разработчиков, неподготовленности пользователей, нечетко определенной задаче. Продолжительность таких проектов от 8 до 36 месяцев.
Подход быстрой разработки (Fast Track) в отличие от каскадного классического, является итерационным и основан на методе DSDM (Dynamic Systems Development Method). В этом подходе четыре этапа – стратегия, моделирование требований, проектирование и генерация системы и внедрение в эксплуатацию. Подход используется для реализации небольших и средних проектов с несложной архитектурой системы, гибкими сроками и четкой постановкой задач. Продолжительность проекта от 4 до 16 месяцев.

15. Методология моделирования функциональной структуры объектов – SADT. Методология моделирования данных – ERD (Entity-Relationship Diagrams) (case-метод Баркера).

Методология SADT (Structured Analysis and Design Technique – Технология структурного анализа и проектирования), разработана Дугласом Т. Россом в 1969-73 годах. В основе методологии SADT лежат два основных принципа:

- SA-блоки на основе которых создается иерархическая многоуровневая модульная система, каждый уровень которой представляет собой законченную систему (блок), поддерживаемую и контролируемую системой (блоком), находящейся над ней.
- Декомпозиция. Процесс декомпозиции проводится до достижения нужного уровня подробности описания. Диаграмма ограничивается 3-6 блоками для того, чтобы детализация осуществлялась постепенно, рис.1.4.

Рисунок 1.4 – Функциональный блок и интерфейсные дуги
SADT-диаграммы

В программе интегрированной компьютеризации производства (ICAM) Министерства обороны США была признана полезность SADT, что привело в 1993 году к стандартизации и публикации ее части, называемой IDEF0 в качестве федерального стандарта в США, а в 2000 году – в качестве руководящего документа по стандартизации в Российской Федерации

Наиболее распространенной методологией моделирования данных являются диаграммы «сущность-связь» – ERD (Entity-Relationship Diagrams). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Нотация ERD была впервые введена П.Ченом (Chen) и получила дальнейшее развитие в работах Баркера.

ERD-диаграмма, рис.1.9, позволяет рассмотреть систему целиком и выяснить требования, необходимые для ее разработки, касающиеся хранения информации.

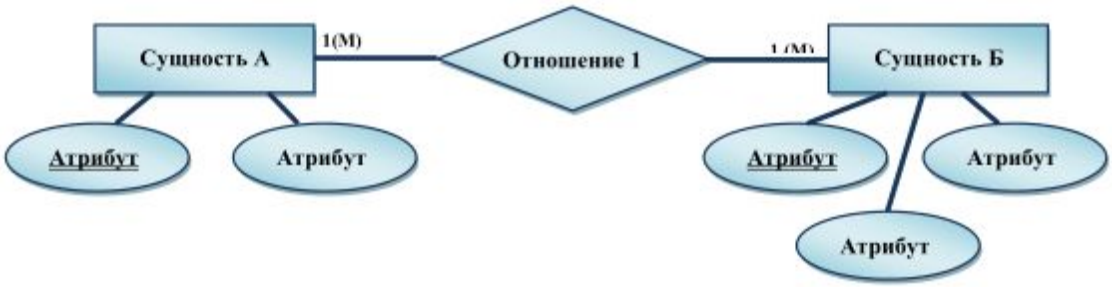


Рисунок 1.9 – Пример ER-диаграммы в нотации П.Чена

Существуют и другие нотации для представления ER-диаграмм: нотация Мартина, нотация IDEF1X (см.п.1.8.6), нотация Баркера.

16. Методология моделирования работы в реальном времени – STD.

Методология STD (State Transition Diagram) предназначена для моделирования аспектов функционирования системы, зависящих от времени или реакции на события (так называемая работа в реальном времени)], рис.1.10.

Рисунок 1.10 – Пример STD узлов

Состояние – рассматривается как устойчивое значение некоторого свойства в течение определенного времени. Начальное состояние – узел STD, являющийся стартовой точкой для начального системного перехода.

Переход определяет перемещение моделируемой системы из одного состояния в другое. При этом имя перехода идентифицирует событие, являющееся причиной перехода и управляющее им.

Управляющий поток – это «трубопровод», через который проходит управляющая информация.

Имеются следующие типы управляющих потоков:

- T-поток (trigger flow) – поток управления процессом, который может вызвать выполнение процесса. При этом процесс включается одной короткой операцией;
- A-поток (activator flow) – поток управления процессом, который может изменять выполнение отдельного процесса. Используется для обеспечения непрерывности выполнения процесса до тех пор, пока поток «включен», с «выключением» потока выполнение процесса завершается;
- E/D-поток (enable/disable flow) –поток управления процессом, который может переключать выполнение отдельного процесса. Течение по E-линии вызывает выполнение процесса, которое продолжается до тех пор, пока не возбуждается течение по D-линии. Можно использовать 3 типа таких потоков: E-поток, D-поток, E/D-поток.

Условие представляет собой событие (или события), вызывающее переход и идентифицируемое именем перехода. Если в условии участвует входной управляющий поток управляющего процесса-предка, то имя потока должно быть заключено в кавычки, например. Кроме условия с переходом может связываться действие или ряд действий, выполняющихся, когда переход имеет место.

Действие – это операция, которая может иметь место при выполнении перехода. Если действие необходимо для выбора выходного управляющего потока, то имя этого потока должно заключаться в кавычки.

На STD **состояния** представляются узлами, а переходы – дугами. Условия (по-другому называемые стимулирующими событиями) идентифицируются именем перехода и возбуждают выполнение перехода. **Действия** или отклики на события привязываются к переходам и записываются под соответствующим усло-

вием.

17. Модель быстрой разработки приложений RAD (Rapid Application Development)

Технология проектирования ИС на основе методологии использования средств быстрой разработки приложений, получила широкое распространение и приобрела название методологии быстрой разработки приложений – RAD (Rapid

Application Development).

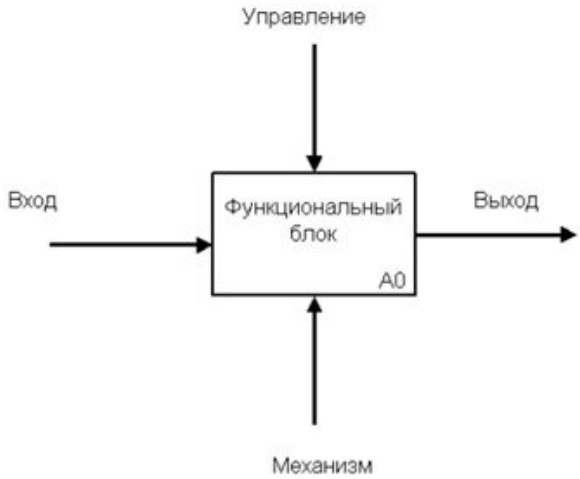
RAD – это комплекс специальных инструментальных средств быстрой разработки прикладных информационных систем, позволяющих оперировать с определенным набором графических объектов, функционально отображающих отдельные информационные компоненты приложений.

Основные принципы методологии RAD можно свести к следующему:

- используется итерационная (спиральная) модель разработки;
- полное завершение работ на каждом из этапов жизненного цикла не обязательно;
- в процессе разработки информационной системы необходимо тесное взаимодействие с заказчиком и будущими пользователями;
- необходимо применение CASE-средств и средств быстрой разработки приложений;
- необходимо применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
- необходимо использование прототипов, позволяющее полнее выяснить и реализовать потребности конечного пользователя;
- тестирование и развитие проекта осуществляются одновременно с разработкой;
- разработка ведется немногочисленной и хорошо управляемой командой профессионалов;
- необходимы грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

18. Методология функционального моделирования процессов – IDEF0. Характеристика диаграмм. Типы взаимосвязей между блоками.

IDEF0 представляет процесс в виде иерархической структуры функциональных блоков



Функциональный блок представляет собой функцию (например, «Производить услуги», «Разработать план производства», «Выставить счет»), к которой относятся интерфейсные дуги четырех типов (Вход, Выход, Управление, Механизм). Входом и выходом функции могут являться как различные реальные объекты (детали, материалы), так и потоки информации (данные о заказах, аналитика продаж). Управление определяет все стандарты, методологии, распоряжения и прочие регламентирующие документы, а Механизм–поддерживающие выполнение функции информационные системы, сотрудники и ресурсы. IDEF0 в обязательном порядке требует указания как минимум одной управляющей и одной исходящей интерфейсной дуги, что объясняется с точки зрения необходимости для каждой функции иметь определенные правила выполнения и определенный результат, иначе смысла в функциональном блоке просто не будет.

IDEF0 используется не только для иллюстрации процессов предприятия, но и в целом для описания схемы его функционирования, с какими ресурсами компания работает, от чего зависит и какой итоговый результат производит. Однако в некоторой степени содержание функциональных блоков IDEF0 остается «черным ящиком», в связи с чем и проводится построение дополнительны схем процессов, уже в нотации IDEF3

Результатом моделирования процессов является модель, которая относится к одному из трех типов:

- модель AS-IS (как есть) - модель текущей организации процессов;
- модель TO-BE (как будет) - модель идеальной организации процессов;
- модель SHOULD-BE(как должно бы быть) - идеализированная модель, не отражающая реальную организацию процессов.

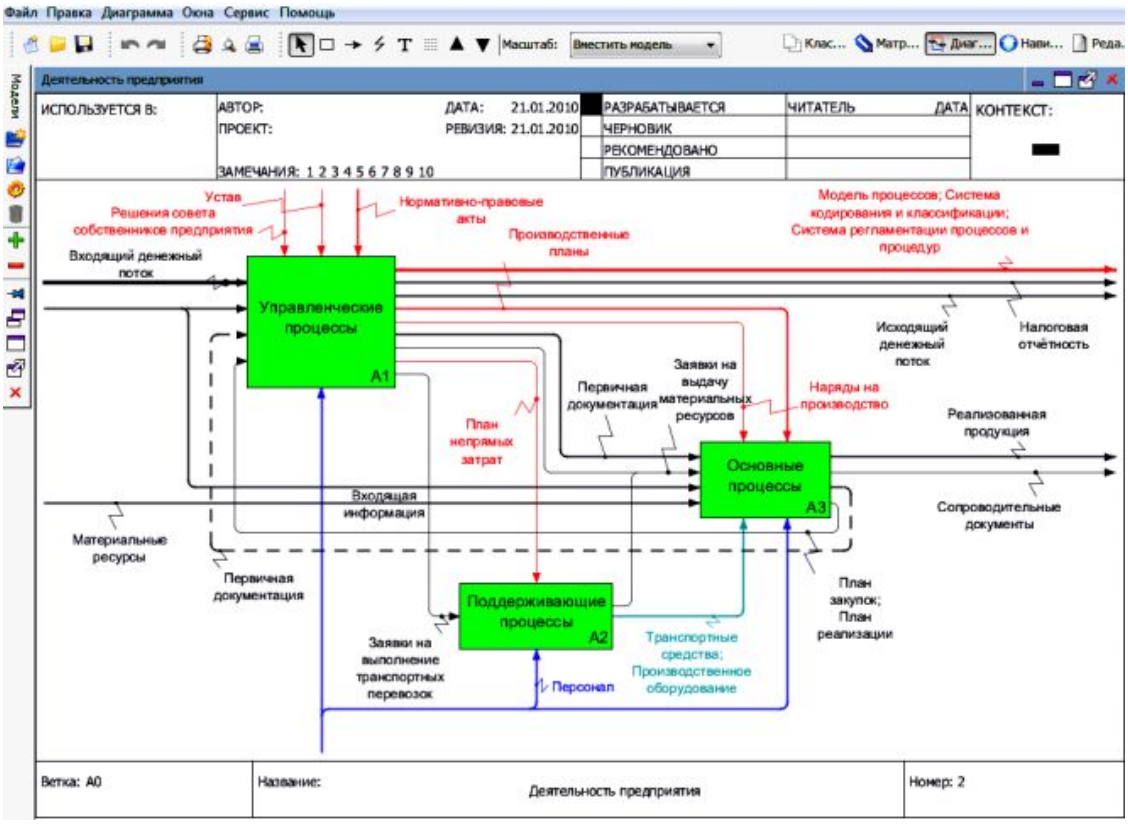
Модель (AS-IS, TO-BE или SHOULD-BE) может содержать 4 типа диаграмм:

- контекстную диаграмму;
- диаграммы декомпозиции;
- диаграммы дерева узлов;
- презентационные диаграммы (диаграммы только для экспозиции) (for exposition only, FEO).

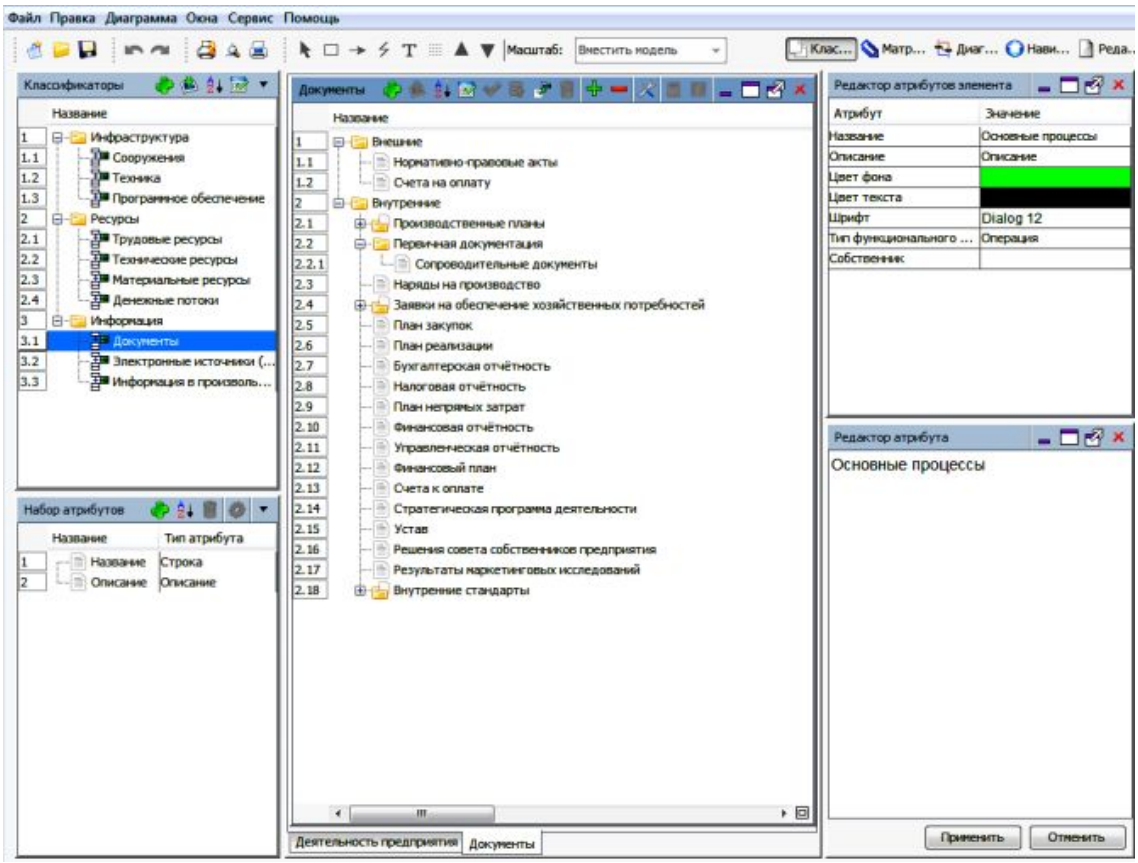
19. [Было бы неплохо дополнить] Кроссплатформенная система моделирования и анализа бизнес-процессов Ramus Educational: возможности, особенности применения при проектировании ИС.

Ramus Education может быть использован для создания диаграмм в формате IDEF0 и DFD. Ramus Education использует формат файлов полностью совместимый с форматом файла коммерческой версии Ramus. Как и Ramus, Ramus Educational поддерживает импорт/экспорт файлов в формат IDL, таким образом реализуя частичную совместимость с подобными программами (например, с CA Erwin Process Modeler). Лицензия Ramus Educational запрещает его использование в коммерческих целях.

Также имеется возможность импорта/экспорта файлов в формат IDL BPWin. Таким образом, обеспечивается частичная совместимость с CA ERwin Process Modeler (в части графических моделей IDEF0).



Область построения диаграмм Ramus Educational

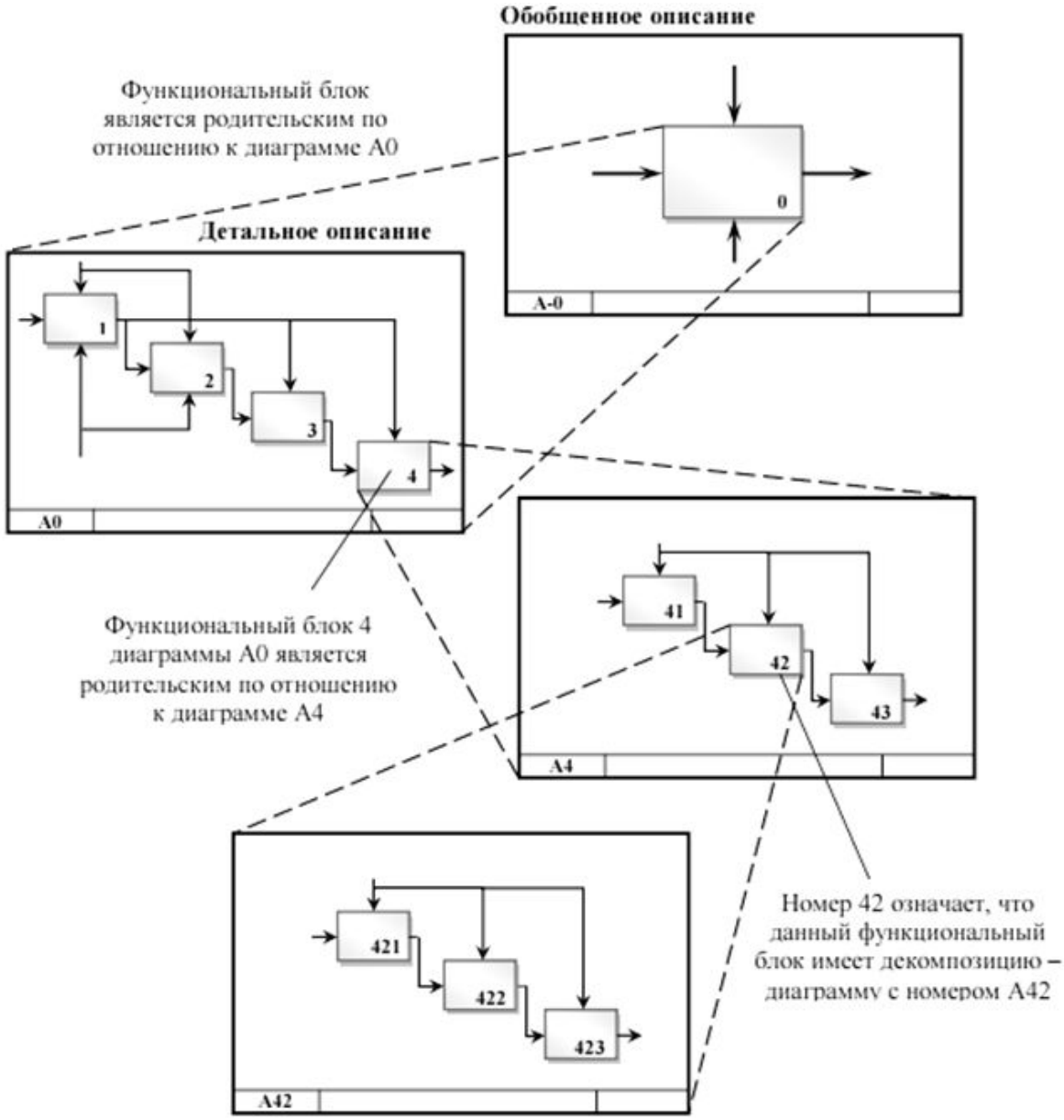


Область документирования процессов в Ramus Educational.

20. Методология функционального моделирования процессов – IDEF0. Последовательность создания функциональных моделей.

Первый шаг построения IDEF0 – построение основной контекстной диаграммы, то есть с представления всей системы в виде простейшей компоненты (контекстной диаграммы). Данная диаграмма отображает назначение (основную функцию) системы, необходимые входные и выходные данные, управляющую и регламентирующую информацию, а также механизмы.

То есть модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока, с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с называется контекстной диаграммой, и обозначается идентификатором «A-0»



Декомпозиция функциональных блоков.

На втором шаге контекстная диаграмма детализируется с помощью диаграммы декомпозиции первого уровня. На этой диаграмме отображаются функции системы, которые должны быть реализованы в рамках основной функции.

Как правило, при построении диаграммы декомпозиции исходная функция (декомпозируемая) разбивается на 3–8 подфункций (блоков).

После построения диаграммы декомпозиции первого уровня для указанных на ней функций строятся отдельные диаграммы (диаграммы декомпозиции второго уровня). Затем процесс декомпозиции (построения диаграмм) продолжается до тех пор, пока дальнейшая детализация функций не теряет смысла. Для каждой атомарной функции, описывающей элементарную операцию (т.е. функции, не имеющей диаграмму декомпозиции), составляется подробная спецификация, определяющая ее особенности и алгоритм реализации.

Стрелки, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются теми же самыми, что и стрелки, входящие в диаграмму нижнего уровня и выходящие из нее.

После определения состава функций и взаимосвязей между ними, возникает вопрос о правильной их композиции (объединении) в модули (подсистемы). При этом подразумевается, что каждая отдельная функция должна решать одну, строго определенную задачу. В противном случае необходима дальнейшая декомпозиция или разделение функций.

При объединении функций в подсистемы необходимо стремиться, чтобы внутренняя связность (между функциями внутри модуля) была как можно сильнее, а внешняя (между функциями, входящими в разные модули), как можно слабее.

- иерархическая связь имеет место между функцией и подфункциями, из которых она состоит;
- регламентирующая связь отражает зависимость одной функции от другой, когда выход одной работы направляется на управление другой.
- функциональная связь имеет место, когда выход одной функции служит входными данными для следующей функции. С точки зрения потока материальных объектов данная связь показывает технологию (последовательность работ) обработки этих объектов.
- потребительская связь имеет место, когда выход одной функции служит механизмом для следующей функции.

Из приведенных выше типов связей наиболее сильной является иерархическая связь, которая, по сути, и определяет объединение функций в модули (подсистемы). Несколько слабее являются регламентирующие, функциональные и потребительские связи. Функции с этими связями обычно реализуются в одной подсистеме.

Третий шаг построение диаграммы дерева узлов. Это обзорная диаграмма, показывающая структуру всей модели. Обычно вершина дерева соответствует контекстному блоку, под вершиной выстраивается вся иерархия блоков модели. Обзор модели с использованием дерева помогает сконцентрироваться на функциональной декомпозиции модели.

Четвёртый шаг построение презентационных FEO-диаграмм для иллюстрации других точек зрения или деталей, выходящих за рамки традиционного синтаксиса IDEF0. Диаграммы FEO допускают нарушение любых правил построения диаграмм IDEF0 в целях выделения важных с точки зрения аналитика частей модели.

Один из способов использования FEO-диаграмм состоит в отделении функционального блока от его окружения посредством создания диаграммы с единственным блоком и всеми относящимися к нему стрелками наподобие контекстной диаграммы

21. Технология анализа взаимосвязей между информационными потоками – IDEF1 (IDEF1X).

Методология IDEF1 разработанная Т.Рэмеем, основана на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

Методология IDEF1 позволяет на основе наглядных графических представлений моделировать информационные взаимосвязи и различия между:

- реальными объектами;
- физическими и абстрактными зависимостями, существующими среди реальных объектов;
- информацией о реальных объектах;
- структурой данных, используемой для приобретения, накопления и управления информацией.

Центральным понятием IDEF1 является понятие «сущность», рис.1.11.

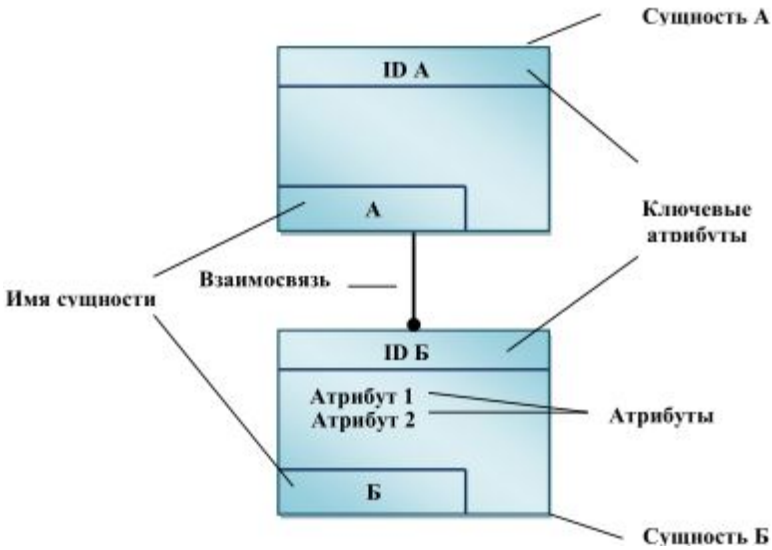


Рисунок 1.11 – Стандарт IDEF1

· разных концов.

Методология IDEF1X – язык для семантического моделирования данных, основанный на концепции «сущность-связь» (ERD). Графическая нотация IDEF1 применяется для построения информационной модели, эквивалентной реляционной модели в третьей нормальной форме. На основе совершенствования метода IDEF1 создана его новая версия – метод IDEF1X, разработанный с учетом таких требований, как простота для изучения и возможность автоматизации.

В IDEF1X все сущности делятся на зависимые и независимые от идентификаторов. Сущность является независимой от идентификаторов, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае – не идентифицирующей

Идентифицирующая связь между сущностью-родителем и сущностью потомком изображается сплошной линией. Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью.

Связь может дополнительно определяться с помощью указания степени или мощности

- каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

Сущности могут иметь также внешние ключи (Foreign Key), которые могут использоваться в качестве части или целого первичного ключа или не ключевого атрибута.

В качестве первичных ключей могут быть использованы несколько атрибутов или групп атрибутов. Атрибуты, которые могут быть выбраны первичными ключами, называются кандидатами в ключевые атрибуты (потенциальные атрибуты). Кандидаты в ключи должны уникально идентифицировать каждую запись сущности.

22. [Думаю надо бы дополнить] Средства CA ERwin Data Modeler Community Edition: возможности, особенности применения.

CA ERwin Data Modeler Community Edition – это бесплатное базовое средство моделирования, включающее в себя подмножество функций флагманского продукта CA ERwin Data Modeler Standard Edition. Оно идеально подходит для начала работы с ведущим в отрасли инструментом моделирования данных. Это решение предоставляет множество базовых функций моделирования данных с ограничением до 25 объектов моделей, в том числе для проектирования и создания баз данных, сравнения моделей, определения стандартов и др., рис.4.6.

Решение CA ERwin Data Modeler Community Edition помогает управлять сложной инфраструктурой данных с помощью следующих основных возможностей:

- визуализация сложных структур данных. Модели данных создаются автоматически, что дает простое графическое представление для визуализации сложных структур баз данных. Ограничение до 25 объектов;
- создание проектов баз данных. Создавайте проекты баз данных непосредственно на основе визуальных моделей, что позволяет повысить эффективность и уменьшить число ошибок. Ограничение до 25 объектов;
- определение стандартов. Повторно используемые стандарты, такие как шаблоны моделей, домены, стандарты именования, улучшают качество и эффективность;
- сравнение моделей и баз данных. Функция Complete Compare осуществляет сравнение моделей, скриптов и баз данных, выводя все различия на экран (в версии Community Edition данные доступны только для чтения);

- отчетность и публикация. Простой, интуитивно понятный интерфейс Report Designer позволяет создавать отчеты в виде текстовых и HTML-файлов, которые могут содержать диаграммы и метаданные.

23. Методология описания (документирования) и моделирования процессов – IDEF3.

Нотация IDEF3, вторая важнейшая нотация (после IDEF0), предназначена для описания потоков работ (Work Flow Modeling).

С помощью диаграмм IDEF3 можно анализировать сценарии из реальной жизни, например, как закрывать магазин в экстренных случаях или какие действия должны выполнить менеджер и продавец при закрытии.

Существуют два типа диаграмм в стандарте IDEF3, представляющие описание одного и того же сценария процесса в разных ракурсах:

- диаграммы Описания Последовательности Этапов Процесса – PFDD (Process Flow Description Diagrams).
- диаграммамы Состояния Объекта и его Трансформаций в Процессе – OSTN (Object State Transition Network).

Диаграмма PFDD, рис.1.12 является графическим отображение сценария процесса.

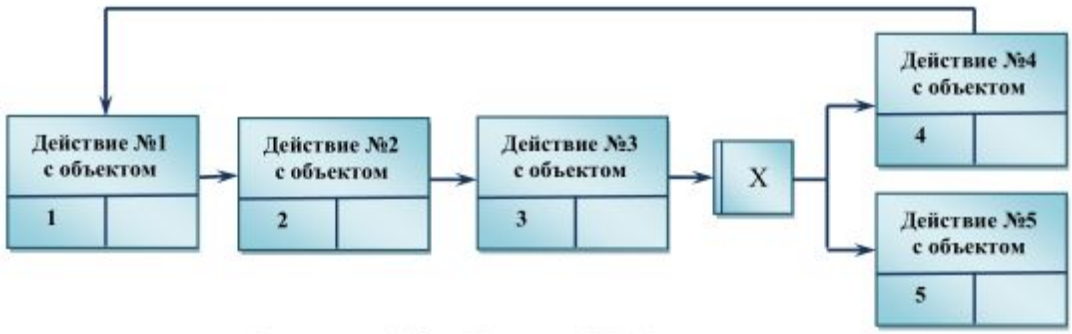


Рисунок 1.12 – Пример PFDD диаграммы

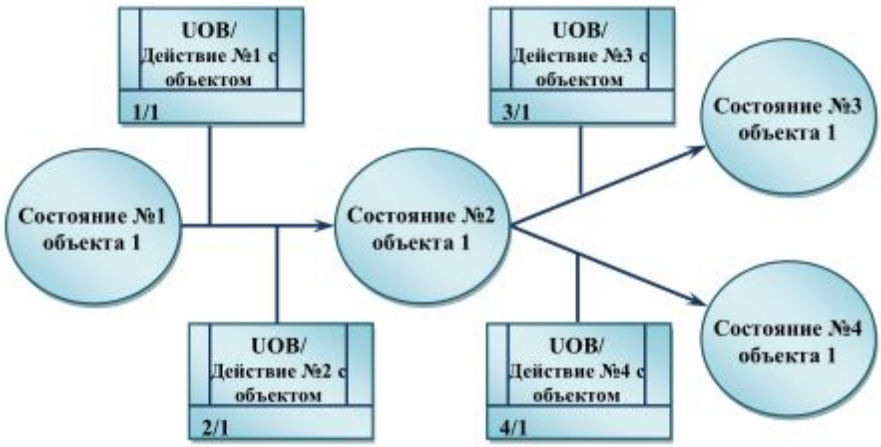
Прямоугольники на диаграмме PFDD называются функциональными элементами или элементами поведения – UOB (Unit of Behavior,) и обозначают событие, стадию процесса или принятие решения.

Если диаграммы PFDD технологический процесс «С точки зрения наблюдателя», то другой класс диаграмм IDEF3 OSTN позволяет рассматривать тот же самый процесс «С точки зрения объекта». На рис.1.13 представлено пример изображения OSTN диаграммы.

Состояния объекта и Изменение состояния являются ключевыми понятиями OSTN диаграммы. Состояния объекта отображаются окружностями, а их изменения направленными линиями. Каждая линия имеет ссылку на соответствующий функциональный блок UOB, в результате которого произошло отображаемое ею изменение состояния объекта.

Нотацию IDEF3 целесообразно применять в случае относительно простых процессов на нижнем уровне декомпозиции, т.е. процессов уровня рабочих мест. В

этом случае схема процесса может служить основой для создания документов, регламентирующих работу исполнителей.



24. Диаграммы PFDD (Process Flow Description Diagrams), OSTN (Object State Transition Network).

Существуют два типа диаграмм в стандарте IDEF3, представляющие описание одного и того же сценария процесса в разных ракурсах:

- диаграммы Описания Последовательности Этапов Процесса – PFDD (Process Flow Description Diagrams).
- диаграммамы Состояния Объекта и его Трансформаций в Процессе – OSTN (Object State Transition Network).

Диаграмма PFDD, рис.1.12 является графическим отображение сценария процесса.

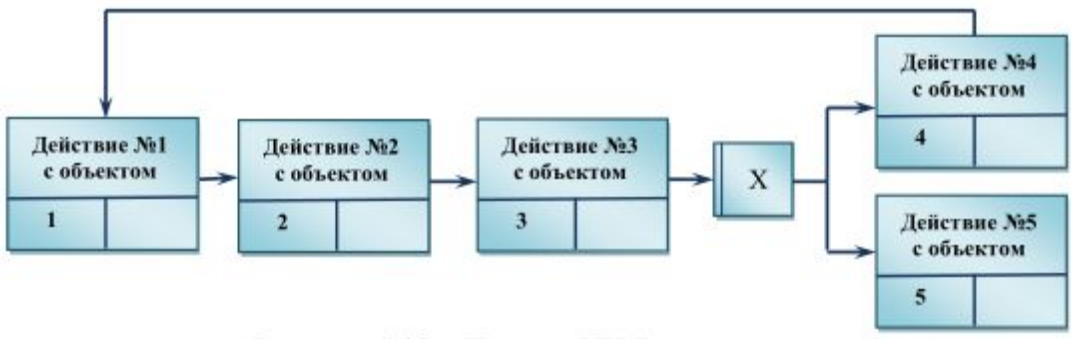


Рисунок 1.12 – Пример PFDD диаграммы

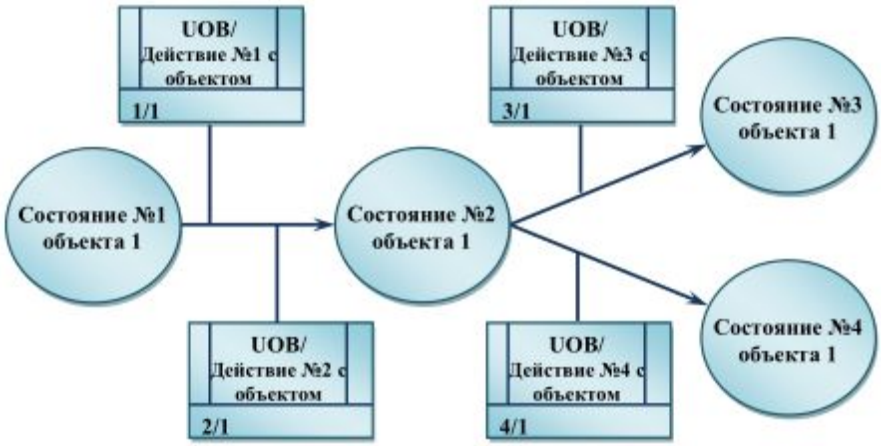
Прямоугольники на диаграмме PFDD называются функциональными элементами или элементами поведения – UOB (Unit of Behavior,) и обозначают событие, стадию процесса или принятие решения.

Если диаграммы PFDD технологический процесс «С точки зрения наблюдателя», то другой класс диаграмм IDEF3 OSTN позволяет рассматривать тот же самый процесс «С точки зрения объекта». На рис.1.13 представлено пример изображения OSTN диаграммы.

Состояния объекта и Изменение состояния являются ключевыми понятиями OSTN диаграммы. Состояния объекта отображаются окружностями, а их изменения направленными линиями. Каждая линия имеет ссылку на соответствующий

функциональный блок UOB, в результате которого произошло отображаемое ею изменение состояния объекта.

Нотацию IDEF3 целесообразно применять в случае относительно простых процессов на нижнем уровне декомпозиции, т.е. процессов уровня рабочих мест. В этом случае схема процесса может служить основой для создания документов, регламентирующих работу исполнителей.



25. Методология рационального унифицированного процесса RUP (Rational Unified Process).

Еще одной ведущей методологией, в которой инструментально поддерживаются все этапы жизненного цикла разработки ПО, является методология Rational Unified Process (RUP). В основе методологии RUP, как и многих других программных методологий, объединяющих инженерные методы создания ПО, лежит «пошаговый подход». Он определяет этапы жизненного цикла, контрольные точки, правила работ для каждого этапа и, тем самым, упорядочивает проектирование и разработку ПО. Для каждого этапа жизненного цикла методология задает: состав и последовательность работ, а также правила их выполнения; распределение полномочий среди участников проекта (роли); состав и шаблоны формируемых промежуточных и итоговых документов; порядок контроля и проверки качества. Модели позволяют рассмотреть будущую систему, ее объекты и их взаимодействие еще до вкладывания значительных средств в разработку, позволяют увидеть ее глазами будущих пользователей снаружи и разработчиков изнутри еще до создания первой строки исходного кода. Большинство моделей представляются UML диаграммами [39]. Определение требований. Унифицированный процесс – это процесс, управляемый прецедентами, которые отражают сценарии взаимодействия пользователей. Фактически, это взгляд пользователей на программную систему снаружи. Таким образом, одним из важнейших этапов разработки, согласно RUP, будет этап определения требований, который заключается в сборе всех возможных пожеланий к работе системы, которые только могут прийти в голову пользователям и аналитикам. Для облегчения этого процесса аналитики используют Диаграммы Прецедентов (Вариантов использования) (Use Case Diagram), рис.1.19. На диаграмме отображаются варианты использования (1) и действующие лица (2), между которыми устанавливаются следующие основные

типы отношений: ассоциация между действующим лицом и вариантом использования (3); обобщение между действующими лицами (4); обобщение между вариантами использования (5); зависимости (различных типов) между вариантами использования (6).

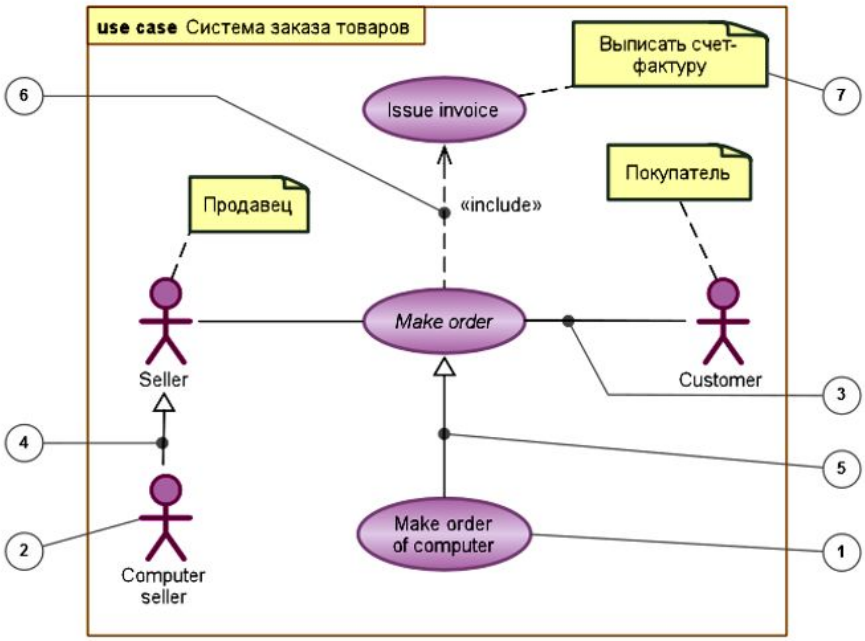


Рисунок 1.19 – Диаграммы Прецедентов (Вариантов использования) (Use Case Diagram)

Для детализации конкретного прецедента используется Диаграмма Активности (Activity Diagram), пример которой дан на рис.1.20. На диаграмме применяют один основной тип сущностей − действие (1), и один тип отношений − переходы (2) (передачи управления и данных). Также используются такие конструкции как развилки, слияния, соединения, ветвления (3), которые похожи на сущности, но таковыми на самом деле не являются, а представляют собой графический способ изображения некоторых частных случаев многоместных отношений.

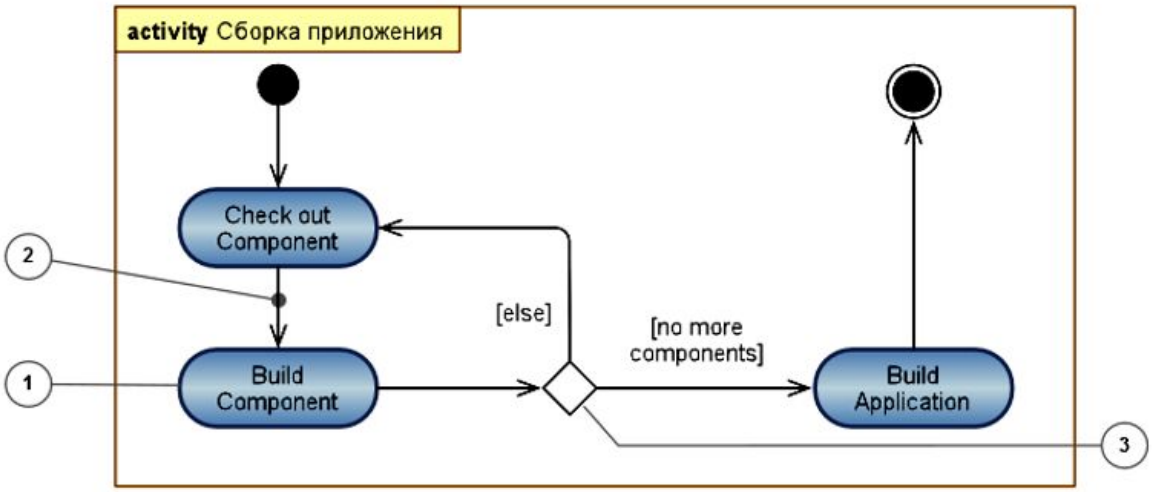


Рисунок 1.20 – Диаграмма Активности (Activity Diagram)

Для создания модели предметной области используется обычная Диаграмма Классов (Class Diagram), рис.1.21, на которой отображаются: основной тип сущностей: классы (1), между которыми устанавливаются следующие основные типы отношений: ассоциация между классами (2); обобщение между классами (3); зависимости (различных типов) между классами 4 и между классами и интерфейсами.).

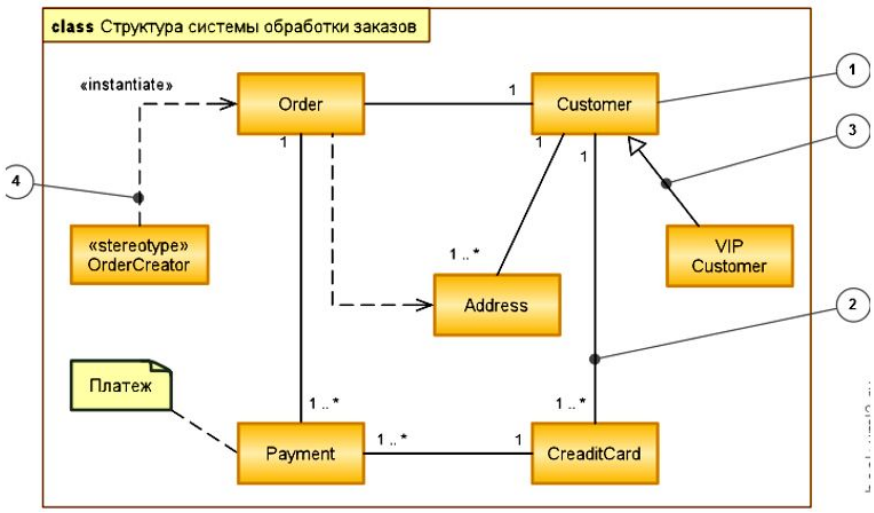


Рисунок 1.21 – Диаграмма Классов (Class Diagram)

Анализ. После определения требований и контекста, в котором будет работать система, наступает черед анализа полученных данных. Аналитическая модель – это взгляд на систему изнутри, в отличие от модели прецедентов, которая показывает, как система будет выглядеть снаружи. Для отображения модели анализа при помощи UML используется Диаграмма Классов со стереотипами (образцами поведения) «граничный класс», «сущность», «управление», а для детализации используются Диаграммы Сотрудничества (Collaboration), рис.1.22.

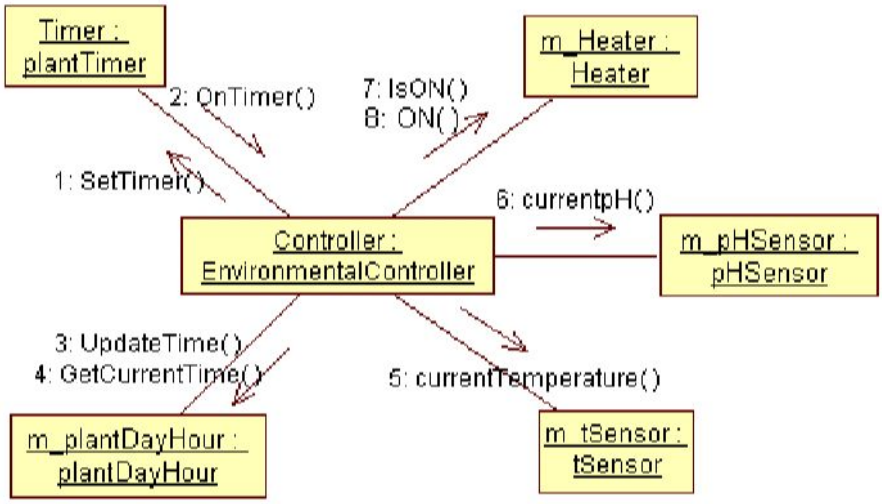


Рисунок 1.22 – Диаграммы Сотрудничества (Collaboration)

Если акцентировать внимание на порядке взаимодействия, то другим его представлением будет Диаграмма Последовательности (Sequence Diagram), рис.1.23.

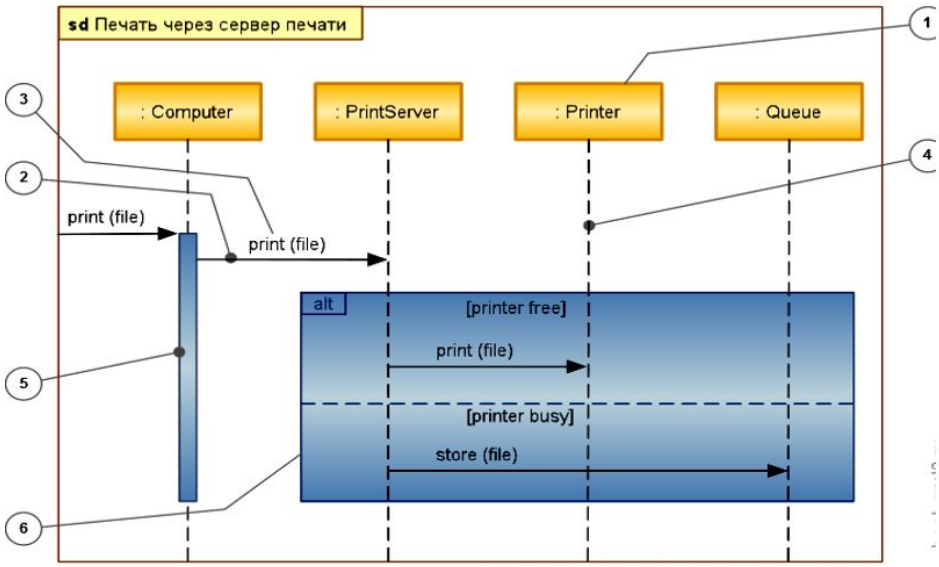


Рисунок 1.23 – Диаграмма Последовательности (Sequence Diagram)

Эта диаграмма позволяет взглянуть на обмен сообщениями во времени, наглядно отобразить последовательность процесса. На диаграмме применяют один основной тип сущностей − экземпляры взаимодействующих классификаторов (1) (в основном классов, компонентов и действующих лиц), и один тип отношений − связи (2), по которым происходит обмен сообщениями (3). Проектирование в ходе которого на основании моделей, созданных ранее, создается модель проектирования. Для создания модели проектирования используются целый набор UML диаграмм: Диаграммы Активности, рис.1.20, Диаграммы Классов, рис.1.21, Диаграммы Коммуникации, рис.1.24, Диаграммы Взаимодействия, рис.1.25.

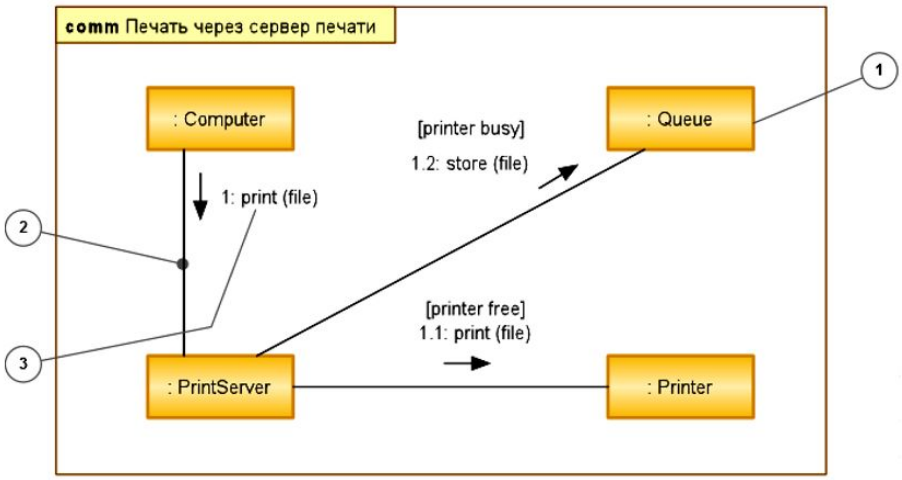


Рисунок 1.24 – Диаграммы Коммуникации (Communication Diagram)

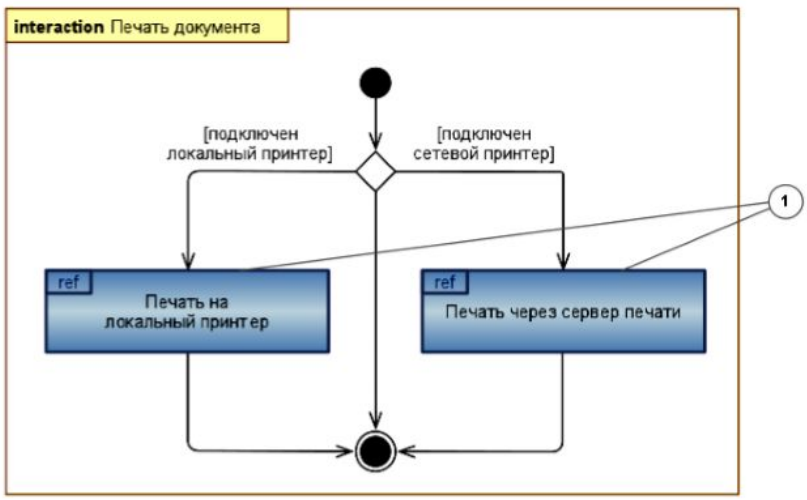


Рисунок 1.25 – Диаграммы Взаимодействия (Interaction diagrams)

Дополнительно в этом рабочем процессе может создаваться модель развертывания, которая реализуется на основе Диаграммы Развертывания (Deployment Diagram), рис.1.26. Это самый простой тип диаграмм, предназначенный для моделирования распределения устройств в сети. На диаграмме присутствует два типа сущностей: артефакт (1), который является реализацией компонента (2) и узел (3) (может быть как классификатор, описывающий тип узла, так и конкретный экземпляр), а также отношение ассоциации между узлами (4), показывающее, что узлы физически связаны во время выполнения. Если одна сущность является частью другой, применяется либо отношение зависимости «deploy» (5), либо фигура одной сущности помещается внутрь фигуры другой сущности (6).

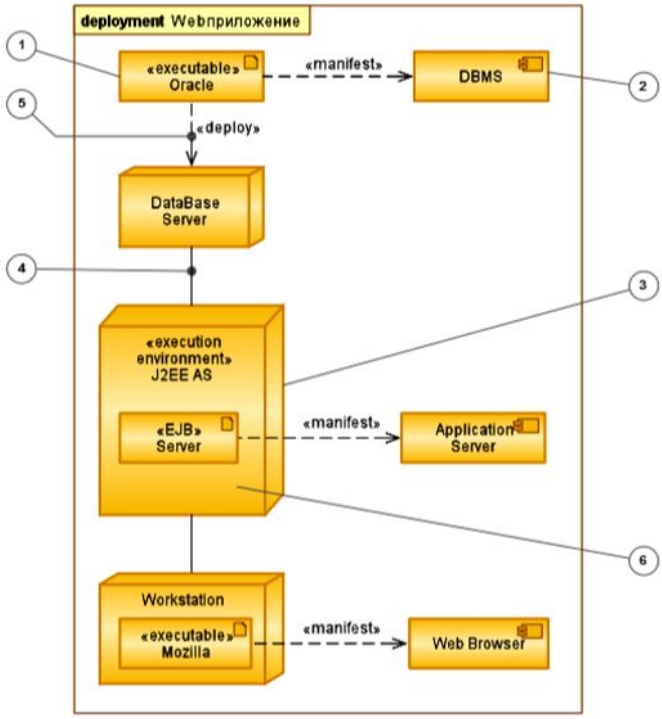


Рисунок 1.26 – Диаграммы Развертывания (Deployment Diagram)

Реализация. Основная задача процесса реализации – создание системы в виде компонентов – исходных текстов программ, сценариев, двоичных файлов, исполняемых модулей и т.д. Данная модель описывает способ организации этих компонентов в соответствии с механизмами структурирования и разбиения на модули, принятыми в выбранной среде программирования и представляется Диаграммой Компонентов (Component Diagram), рис.1.27.

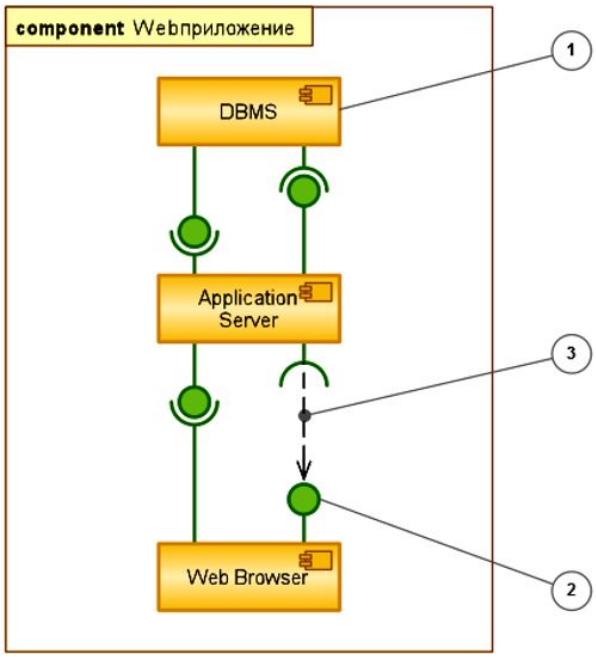


Рисунок 1.27 – Диаграммы Компонентов
(Component Diagram)

Основной тип сущностей на диаграмме компонентов − это сами компоненты (1), а также интерфейсы (2), посредством которых указывается взаимосвязь между компонентами. На диаграмме компонентов применяются следующие отношения: реализации между компонентами и интерфейсами (компонент реализует интерфейс); зависимости между компонентами и интерфейсами (компонент использует интерфейс) (3). Тестирование. В процессе тестирования проверяются результаты реализации. Для данного процесса создается модель тестирования, которая состоит из тестовых примеров, процедур тестирования, тестовых компонентов, однако не имеет отображения на UML диаграммы. RUP довольно обширен, и содержит рекомендации по ведению различных программных проектов, от создания программ группой разработчиков в несколько человек, до распределенных программных проектов, объединяющих тысячи человек на разных континентах.

26. [Добавить сходства, отличия] Диаграмма Последовательности (Sequence Diagram) и диаграмма Ганта: сходства, отличия, назначение.

Это есть в конспекте по Управлению IT проектами, у кого есть добавьте

Диаграмма последовательности - диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определенного объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие акторов (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов). Используется в языке UML.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

На данной диаграмме объекты располагаются слева направо.

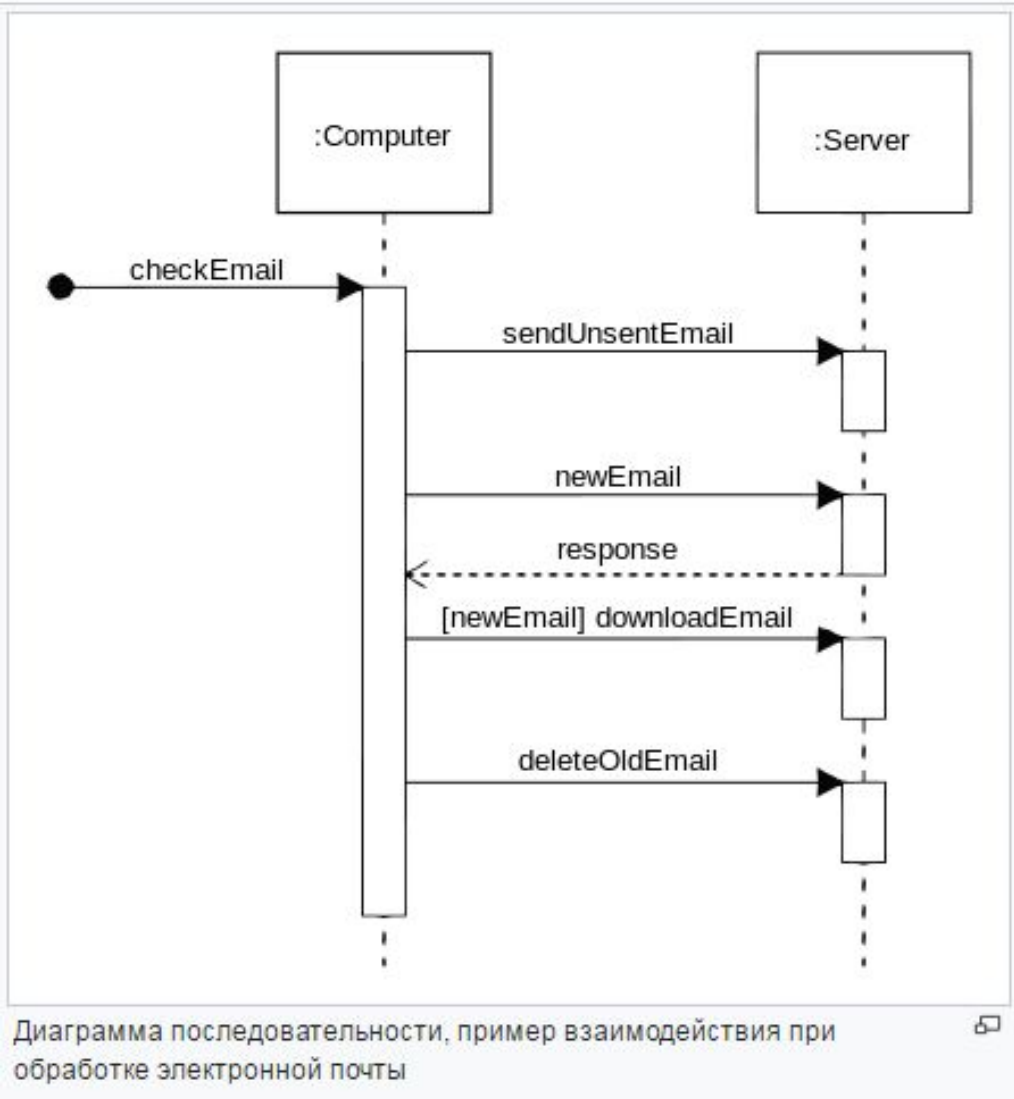


Диаграмма последовательности, пример взаимодействия при обработке электронной почты

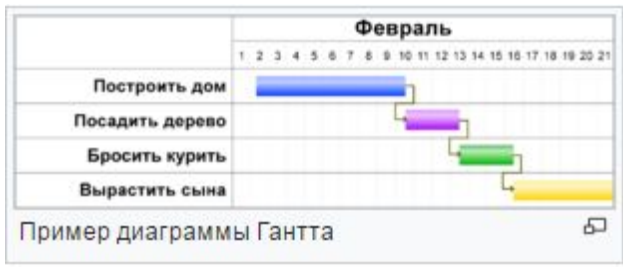
Как было сказано выше, взаимодействие между акторами отображается при помощи специальных стрелок, передающих управление от отправителя (от кого идёт стрелка) к получателю (тот, к кому направлена стрелка). Стрелки демонстрируют ход сценария и те события, которые происходят во время анализируемого прецедента. Всего существует 5 видов стрелок:

- **Синхронное сообщение** - актор-отправитель передаёт ход управления актору-получателю, которому необходимо провести в прецеденте некоторое действие. Пока проводимое актором-получателем действие не будет завершено (соответственно, не будет получено ответное сообщение), актор-отправитель теряет возможность производить какие-либо действия. Графически изображается как стрелка с закрашенным треугольником, после которой идёт прямоугольник, отражающий деятельность объекта,в конце которого находится ответное сообщение.
- **Ответное сообщение** - данное сообщение является ответом на синхронное сообщение. Обычно, содержит какое-либо возвращаемое изначальному актору-отправителю значение, также возвращающее ему управление (возможность действовать).

- **Асинхронное сообщение** - актор-отправитель передаёт ход управления актору-получателю, которому необходимо провести в прецеденте некоторое действие. Основное отличие от синхронного сообщения состоит в том, что актор-отправитель не теряет возможность совершать другие действия.
- **Потерянное сообщение** - сообщение без адресата (есть отправитель, нет получателя).
- **Найденное сообщение** - сообщение без отправителя.

Последние два вида стрелок (взаимодействий) используются крайне редко. В основном, они используются для демонстрации взаимодействия имеющихся объектов в данном прецеденте с внешними системами.

Диагра́мма Га́нтта — это популярный тип столбчатых диаграмм (гистограмм), который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Используется в приложениях по управлению проектами.



Пример диаграммы Гантта

По сути, диаграмма Гантта состоит из полос, ориентированных вдоль оси времени. Каждая полоса на диаграмме представляет отдельную задачу в составе проекта (вид работы), её концы — моменты начала и завершения работы, её протяженность — длительность работы. Вертикальной осью диаграммы служит перечень задач. Кроме того, на диаграмме могут быть отмечены совокупные задачи, проценты завершения, указатели последовательности и зависимости работ, метки ключевых моментов (вехи), метка текущего момента времени «Сегодня» и др.

Ключевым понятием диаграммы Гантта является «Веха» — метка значимого момента в ходе выполнения работ, общая граница двух или более задач. Вехи позволяют наглядно отобразить необходимость синхронизации, последовательности в выполнении различных работ. Вехи, как и другие границы на диаграмме, не являются календарными датами. Сдвиг вехи приводит к сдвигу всего проекта. Поэтому диаграмма Гантта не является, строго говоря, графиком работ. Кроме того, диаграмма Гантта не отображает значимости или ресурсоемкости работ, не отображает сущности работ (области действия). Для крупных проектов диаграмма Гантта становится чрезмерно тяжеловесной и теряет всякую наглядность.

Указанные выше недостатки и ограничения серьёзно ограничивают область применения диаграммы. Тем не менее, в настоящее время диаграмма Гантта является стандартом де-факто в теории и практике управления проектами, по крайней мере, для отображения *Структуры перечня работ* по проекту.

27. [Дописать технологии] Технологии, основанные на моделировании, анализе бизнес-процессов BPMN.

Модель и нотация бизнес-процессов (BPMN, Business Process Model and Notation) – методология моделирования, анализа и реорганизации бизнес-процессов.

Разработана Business Process Management Initiative (BPMI), с 2005 г. поддерживается и развивается Object Management Group (OMG). В отличие от других методологий бизнес-моделирования, имеющих статус «фирменного» (EPC) или «национального» (IDEF0) стандарта, BPMN получила «международный» статус – Международная организация по стандартизации опубликовала стандарт «ISO/IEC 19510:2013. Information technology – Object Management Group.Business Process Model and Notation».

Диаграмма процесса в нотации BPMN представляет собой алгоритм выполнения процесса. В нотации BPMN выделяют пять основных категорий элементов:

- элементы потока (Flow Objects) (события, процессы и шлюзы);
- данные (объекты данных и базы данных) (товарно-материальные ценности (ТМЦ) или информация);
- соединяющие элементы (Connecting Objects) (потоки управления, потоки сообщений и ассоциации);
- зоны ответственности (Swimlanes) (пулы и дорожки);
- артефакты (сноски) (Artifacts).

Элементы потока являются важнейшими графическими элементами, определяющими ход бизнес-процесса. Элементы потока, в свою очередь, делятся на: события (Events); действия (Activities); шлюзы (Gateways).

Артефакты используются для добавления дополнительной информации о Процессе. Текущий перечень Артефактов включает в себя следующие элементы:

объект данных (Data object); группа (Gruop); аннотация (Annotation).

Разновидности диаграмм (типы процессов) BPMN:

диаграмма процессов (Process Diagram):

- частный (внутренний) бизнес-процесс (Private (internal) Business Process), рис.1.14;
- публичный (открытый) процесс (Public Process), рис.1.15;
- диаграмма хореографии (Choreography Diagram), рис.1.16;
- диаграмма взаимодействия (Collaboration Diagram):
- процессов (Process), рис.1.17;
- посредством обмена сообщениями (A view of Conversations), рис.1.18.

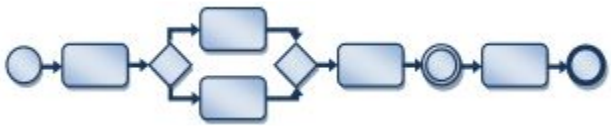


Рисунок 1.14 – Примерный вид BPMN диаграммы частного (внутреннего) бизнес-процесса

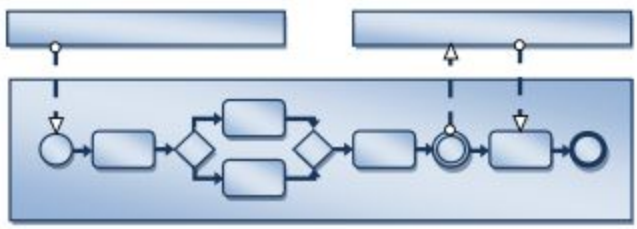


Рисунок 1.15 – Примерный вид BPMN диаграммы публичного (открытого) бизнес-процесса

Рисунок 1.16 – Примерный вид BPMN диаграммы хореографии

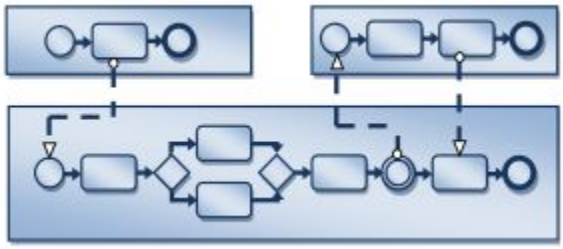


Рисунок 1.17 – Примерный вид BPMN диаграммы взаимодействия процессов

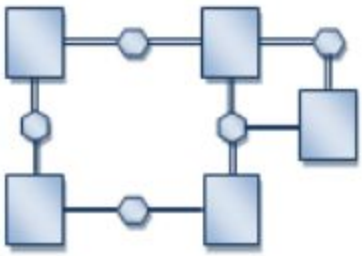


Рисунок 1.18 – Примерный вид BPMN диаграммы посредством обмена сообщениями

28. Инструмент для моделирования бизнес-процессов ARIS Express: типы диаграмм, возможности.

Концепция Архитектуры Интегрированных Информационных Систем - ARIS (ARchitecture of Integrated Information Systems) разработана профессором А.В. Шеером (Scheer).

Эта концепция имеет два основных преимущества:

- позволяет выбрать методы и интегрировать их, опираясь на основные особенности моделируемого объекта;
- служит базой для управления сложными проектами, поскольку благодаря структурным элементам содержит встроенные модели процедур для разработки интегрированных информационных систем.

Архитектура ARIS явилась основой ARIS Toolset – инструментальной среды, разработанной компанией IDS Scheer AG.

В методологии ARIS для описания различных подсистем организации используется более ста типов моделей, отражающих различные аспекты деятельности и реализующих различные методы моделирования (в том числе событийная цепочка процесса EPC (Event driven Process Chain):

- модель «сущность-связь» ERM (Entity Relationship Model);
- модели методики объектно-ориентированного моделирования OMT (Object Modeling Technique);

- модели BSC (Balanced Scorecard) – система сбалансированных показателей;
- модели UML и многие другие.

Все многообразие типов моделей ARIS подразделяется на пять видов описания в соответствии с основными подсистемами предприятия:

- организационной;
- функциональной,
- подсистемой данных;
- подсистемой процессов;
- подсистемой продуктов/услуг.

Остальные подсистемы могут моделироваться с использованием типов объектов, входящих в перечисленные виды описания.

29. Реинжиниринг бизнес-процессов в ИС (реорганизации бизнес-процессов – BPMN) при проектировании ИС.

Модель и нотация бизнес-процессов (BPMN, Business Process Model and Notation) – методология моделирования, анализа и реорганизации бизнес-процессов. В отличие от других методологий бизнес-моделирования, имеющих статус «фирменного» (EPC) или «национального» (IDEF0) стандарта, BPMN получила «международный» статус – Международная организация по стандартизации опубликовала стандарт «ISO/IEC 19510:2013. Information technology – Object Management Group. Business Process Model and Notation» [36, 37].

Основной целью BPMN является обеспечение доступной нотацией описания бизнес-процессов всех пользователей: от аналитиков, создающих схемы процессов, и разработчиков, ответственных за внедрение технологий выполнения бизнес-процессов, до руководителей и обычных пользователей, управляющих этими бизнес-процессами и отслеживающих их выполнение. Таким образом, BPMN нацелена на устранение расхождения между моделями бизнес-процессов и их реализацией.

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. Декомпозиция может производиться в нотациях BPMN или EPC. При декомпозиции процесса BPMN, расположенного на диаграмме SADT, стрелки с диаграммы SADT на диаграмму BPMN не переносятся.

В нотации BPMN выделяют пять основных категорий элементов:

− элементы потока (Flow Objects) (события, процессы и шлюзы);

− данные (объекты данных и базы данных) (товарно-материальные ценности (ТМЦ) или информация);

− соединяющие элементы (Connecting Objects) (потоки управления, потоки сообщений и ассоциации);

− зоны ответственности (Swimlanes) (пулы и дорожки);

− артефакты (сноски) (Artifacts).

Элементы потока являются важнейшими графическими элементами, определяющими ход бизнес-процесса. Элементы потока, в свою очередь, делятся на:

− события (Events);

− действия (Activities);

− шлюзы (Gateways).

Выделяют три вида соединяющих элемента потока, связывающихся друг с другом и с другими элементами:

− поток операций (Sequence Flow);

− поток сообщений (Message Flow);

− ассоциация (Association).

Существуют два способа группировки основных элементов моделирования с помощью зон ответственности:

− группировка с помощью Пула (Pool);

− группировка с помощью Дорожки (Lane).

Артефакты используются для добавления дополнительной информации о Процессе.

Выделяют три типовых Артефакта, что, однако, не запрещает разработчикам моделей бизнес-процессов либо программам моделирования добавлять необходимое количество Артефактов. Для широкого круга пользователей, а также для вертикальных рынков существует возможность стандартизации более полного перечня Артефактов. Таким образом, текущий перечень Артефактов включает в себя следующие элементы:

− объект данных (Data object);

− группа (Gruop);

− аннотация (Annotation).

Все многообразие процессов и способов взаимодействия между их участниками в BPMN поделено на типы (sub-model). Каждому из типов соответствует своя семантика и набор отображаемых элементов.

Разновидности диаграмм (типы процессов) BPMN:

− диаграмма процессов (Process Diagram):

− частный (внутренний) бизнес-процесс (Private (internal) Business Process), рис.1.24;

− публичный (открытый) процесс (Public Process), рис.1.25;

− диаграмма хореографии (Choreography Diagram), рис.1.26;

− диаграмма взаимодействия (Collaboration Diagram):

− процессов (Process), рис.1.27;

− посредством обмена сообщениями (A view of Conversations), рис. 1.28.

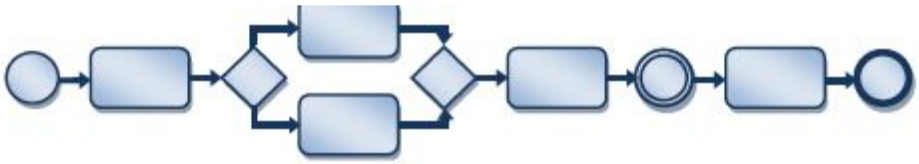


Рис. 1.24. Примерный вид BPMN диаграммы частного (внутреннего) бизнес-процесса.

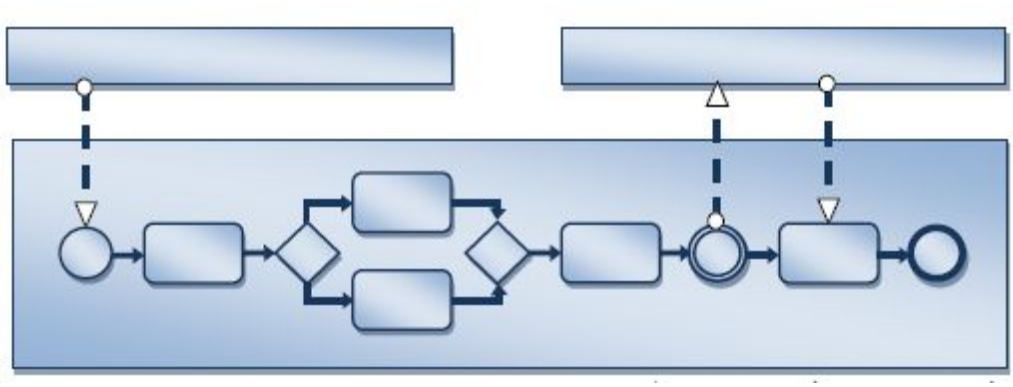


Рис.1.25. Примерный вид BPMN диаграммы публичного (открытого) процесса.

Процесс моделирования процессов с помощью BPMN подчиняется классическим принципам моделирования: декомпозиции и иерархического упорядочивания. Декомпозиция, с отображением на отдельных диаграммах, выполняется для участников (пулов) и отдельных подпроцессов, подобно работам на диаграммах IDEF0 или предопределенным процессам на блок-схемах.

По заявлению разработчиков стандарта BPMN, он вобрал в себя лучшие идеи, что имеются в следующих нотациях и методологиях моделирования:

− UML (Unified Modeling Language, унифицированный язык моделирования):

− Activity Diagram (диаграмма деятельности);

− EDOC (Enterprise Distributed Object Computing, корпоративная распределенная обработка объектов) – Business Processes (бизнес- процессы);

− IDEF (SADT);

− ebXML (Electronic Business eXtensible Markup Language), расширяемый язык разметки для электронного бизнеса) BPSS (Business Process Specification Schema), схемы спецификации бизнес-процессов;

− ADF (Activity-Decision Flow, поток «деятельность-результат») Diagram;

− RosettaNet;

− LOVEM (Line of Visibility Engineering Methodology, визуальная методология проектирования);

− EPC.

30. [из инета, так что хз / вообще-то тут должен быть текст из слайдов] Технологии объектно-ориентированного подхода (анализа) при проектировании ИС.

Нифига тут не текст из слайдов должен быть, этот вопрос был на мисписите, а значит Пелипас нам его читать не мог.

Объектно-ориентированный анализ и проектирование (ООАП, Object-Oriented Analysis/Design) - технология разработки программных систем, в основу которых положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов .

Методология ООАП тесно связана с концепцией автоматизированной разработки программного обеспечения (Computer Aided Software Engineering, CASE).

В рамках ООАП исторически рассматривались три графических нотации:

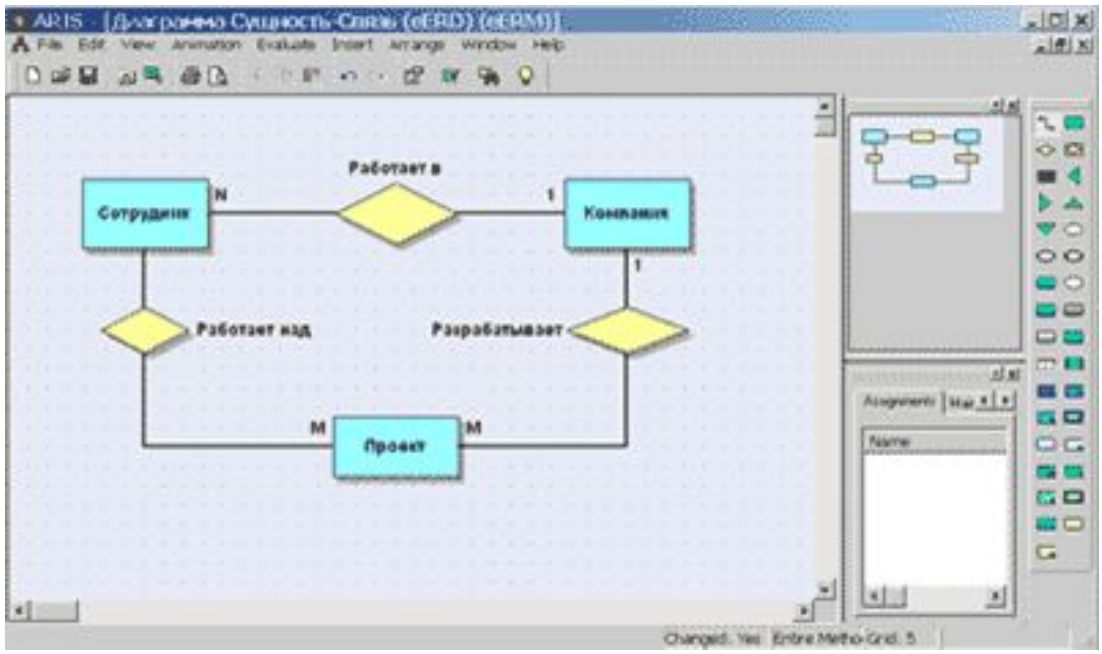
- диаграммы "сущность-связь" (Entity-Relationship Diagrams, ERD),
- диаграммы функционального моделирования (Structured Analysis and Design

Technique, SADT),

- диаграммы потоков данных (Data Flow Diagrams, DFD).

Диаграммы "сущность-связь" (ERD) предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают набор стандартных обозначений для определения данных и отношений между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними.

Основными понятиями данной нотации являются понятия сущности и связи. При этом под сущностью (entity) понимается произвольное множество реальных или абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. Связь (relationship) определяется как отношение или ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения, в частности "отец-сын" или производственные - "начальник-подчиненный". Другой тип связей задается отношениями "иметь в собственности" или "обладать свойством". Различные типы связей графически изображаются в форме ромба с соответствующим именем данной связи.



Нотация диаграмм (ERD) реализована в различных программных средствах. Пример диаграммы ERD, разработанной с помощью средства моделирования бизнес-процессов ARIS. Ограниченность диаграмм ERD проявляется при конкретизации концептуальной модели в более детальное представление моделируемой программной системы, которое кроме статических связей должно содержать информацию о поведении или функционировании отдельных ее компонентов.

В рамках диаграмм функционального моделирования было разработано несколько графических языков моделирования, которые получили следующие названия:

- Нотация IDEF0 - для документирования процессов производства и отображения информации об использовании ресурсов на каждом из этапов проектирования систем
- Нотация IDEF1 - для документирования информации о производственном окружении систем

- Нотация IDEF2 - для документирования поведения системы во времени

Нотация IDEF2 никогда не была полностью реализована.

Процесс моделирования IDEF представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели системы какой-либо предметной области. Функциональная модель IDEF отображает структуру процессов функционирования системы и ее отдельных подсистем, то есть, выполняемые ими действия и связи между этими действиями. Для этой цели строятся специальные модели, которые позволяют в наглядной форме представить последовательность определенных действий. Исходными строительными блоками любой модели нотации IDEF0 процесса являются деятельность (activity) и стрелки (arrows).

Одна из наиболее важных особенностей нотации IDEF0 - постепенное введение все более детальных представлений модели системы по мере разработки отдельных диаграмм. Построение модели IDEF0 начинается с представления всей системы в виде простейшей диаграммы, состоящей из одного блока процесса и стрелок ICOM, служащих для изображения основных видов взаимодействия с объектами вне системы. Поскольку исходный процесс представляет всю систему как единое целое, данное представление является наиболее общим и подлежит дальнейшей декомпозиции.

В конечном итоге модель IDEF0 представляет собой набор иерархически взаимосвязанных диаграмм с сопроводительной документацией, которая разбивает исходное представление сложной системы на отдельные составные части. Детали каждого основного процесса представляются в виде более подробных процессов на других диаграммах. В этом случае каждая диаграмма нижнего уровня является декомпозицией процесса из более общей диаграммы. Поэтому на каждом шаге декомпозиции более общая диаграмма конкретизируется на ряд детальных диаграмм.

Основной недостаток данной методологии связан с отсутствием явных средств для объектно-ориентированного представления моделей сложных систем. Некоторые аналитики отмечают важность знания и применения нотации IDEF0, однако отсутствие возможности реализации соответствующих графических моделей в объектно-ориентированном программном коде существенно сужают диапазон решаемых с ее помощью задач.

В основе графического моделирования информационных систем с помощью диаграмм потоков данных лежит специальная технология построения диаграмм потоков данных DFD. В разработке методологии DFD приняли участие многие аналитики, среди которых следует отметить Э. Йордона. Он автор одной из первых графических нотаций DFD.

Недостаток рассмотренных нотаций связан с отсутствием явных средств для объектно-ориентированного представления моделей сложных систем, а также сложных алгоритмов обработки данных. Поскольку на рассмотренных типах диаграмм не указываются характеристики времени выполнения отдельных процессов и передачи данных между процессами, то модели систем, реализующих синхронную обработку данных, не могут быть адекватно представлены в этих нотациях. Все эти особенности методов структурного системного анализа ограничили возможности широкого применения соответствующих нотаций и послужили основой для разработки

унифицированного языка моделирования UML.

31. Технология объектно-ориентированного подхода с использованием паттернов.

•Общие Принципы Распределения Обязанностей — принципы, используемые в объектно-ориентированном проектировании для решения общих задач по назначению обязанностей классам и объектам.

•Принципы GRASP:

–Information Expert (Информационный эксперт)

–Creator (Создатель)

–Controller (Контроллер)

–Low Coupling (Слабое зацепление)

–High Cohesion (Сильная связность)

–Protected Variations (Сокрытие реализации)

–Polymorphism (Полиморфизм)

–Pure Fabrication (Чистая выдумка)

–Indirection (Посредник)

Шаблон **Information Expert** определяет базовый принцип назначения обязанностей. Он утверждает, что обязанности должны быть назначены объекту, который владеет максимумом необходимой информации для выполнения обязанности. Такой объект называется информационным экспертом.

Возможно, этот шаблон является самым очевидным из девяти, но вместе с тем и самым важным.

Если дизайн не удовлетворяет этому принципу, то при программировании получается спагетти-код, в котором очень трудно разбираться. Локализация обязанностей позволяет повысить уровень инкапсуляции и уменьшить уровень зацепления. Кроме читабельности кода повышается уровень готовности компонента к повторному использованию.

Перекликается с SRP (Принцип единственной обязанности)

•Шаблон **Creator** решает, кто должен создавать объект.

•Фактически, это применение шаблона Information Expert к проблеме создания объектов.

•Более конкретно, нужно назначить классу B обязанность создавать экземпляры класса A, если выполняется как можно больше из следующих условий:

–Класс B содержит или агрегирует объекты A.

–Класс B регистрирует экземпляры объектов A.

–Класс B активно использует объекты A

–Класс B обладает данными инициализации для объектов A.

•Альтернативой создателю является шаблон проектирования Фабрика. В этом случае создание объектов концентрируется в отдельном классе.

•**Контроллер** берёт на себя ответственность за выполнение операций, инициируемых пользователем и часто выполняет сценарий одного или нескольких вариантов использования (например, один контроллер может обрабатывать сценарии создания и удаления пользователя).

•Контроллер не относится к интерфейсу пользователя

•Как правило, контроллер не выполняет работу самостоятельно, а делегирует обязанности компетентным объектам.

Иногда класс-контроллер представляет всю систему в целом, корневой объект, устройство или важную подсистему (внешний контроллер).)

•**Слабое зацепление** — это принцип, который позволяет распределить обязанности между объектами таким образом, чтобы степень зацепления между объектами оставалась низкой.

•Степень зацепления — это мера, определяющая, насколько жестко один элемент связан с другими элементами, либо каким количеством данных о других элементах он обладает.

•Элемент с низкой степенью зацепления (или слабым зацеплением) зависит от не очень большого числа других элементов и имеет следующие свойства:

–Малое число зависимостей между классами (подсистемами).

–Слабая зависимость одного класса (подсистемы) от изменений в другом классе (подсистеме).

–Высокая степень повторного использования подсистем.

•**Сильная связность** — это принцип, который задаёт свойство сильной связности внутри подсистемы. Классы (подсистемы) таким образом получаются сфокусированными, управляемыми и понятными.

•Связность (cohesion) (или более точно, функциональная связность) — это мера связности и сфокусированности обязанностей класса. Считается, что объект (подсистема) обладает высокой степенью связности, если его обязанности хорошо согласованы между собой и он не выполняет огромных объемов работы.

•Класс с низкой степенью связности выполняет много разнородных функций или несвязанных между собой обязанностей. Такие классы создавать нежелательно, поскольку они приводят к возникновению следующих проблем:

–Трудность понимания.

–Сложность при повторном использовании.

–Сложность поддержки.

–Ненадежность, постоянная подверженность изменениям.

–Классы с низкой степенью связности, как правило, являются слишком «абстрактными» или выполняют обязанности, которые можно легко распределить между другими объектами.

•Шаблон **Сокрытие реализации** защищает элементы от изменения других элементов (объектов или подсистем) с помощью вынесения взаимодействия в фиксированный интерфейс.

•Всё взаимодействие между элементами должно происходить через него.

•Поведение может варьироваться лишь с помощью создания другой реализации интерфейса.

•**Полиморфизм** позволяет обрабатывать альтернативные варианты поведения на основе типа и заменять подключаемые компоненты системы.

•Обязанности распределяются для различных вариантов поведения с помощью полиморфных операций для этого класса.

•Все альтернативные реализации приводятся к общему интерфейсу.

•**Чистая выдумка** — это класс, не отражающий никакого реального объекта предметной области, но специально придуманный для усиления связности, ослабления зацепления или увеличения степени повторного использования.

•Чистая выдумка отражает концепцию сервисов в модели проблемно-ориентированного проектирования.

•Пример:

–Есть класс Client, который нужно записать в базу данных.

–Мы не можем позволить классу изнутри лезть в базу данных, так как это приведёт к слабой сцепленности внутри класса.

–И мы создаём новый класс с именем «ClientSaver», который будет иметь сильную сцепленность внутри и единую ответственность записывать объект класса в базу данных.

•Indirection поддерживает слабую связность между двумя элементами (и возможность повторного использования) путём назначения обязанности посредника между ними промежуточному объекту.

•Пример:

–В предыдущем примере добавленный класс «ClientSaver» будет являться промежуточным звеном между БД и нашим основным классом.

–Примерами специализированных вариантов Indirection являются шаблоны Adapter, Facade, Observer (шаблоны Gang of Four, о которых речь будет идти далее).

•Целью введения промежуточного звена является обеспечение слабой связности за счет отделения друг от друга различных компонентов.

SOLID

•S - Single responsibility principle

На каждый класс должна быть возложена единственная обязанность.

•O - Open/closed principle

Программные сущности должны быть открыты для расширения, но закрыты для изменения.

•L - Liskov substitution principle

Функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом.

•I - Interface segregation principle

Много специализированных интерфейсов лучше, чем один универсальный.

•D - Dependency inversion principle

Зависимости внутри системы строятся на основе абстракций. Модули верхнего уровня не зависят от модулей нижнего уровня. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

•Принцип **единственной обязанности** - каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.

•Пример – генератор отчетов. Имеет две обязанности – выборка данных для отчета и формирование отчета. Должен быть разделен на два класса.

•Причина – большая устойчивость к изменениям. Меняем формат отчета – не трогаем выборку.

•Принцип **открытости/закрытости** – «программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения» - это означает, что такие сущности могут позволять менять свое поведение без изменения их исходного кода.

•Пример – любой случай расширяющего возможности наследования (например Заказ и ЗаказПоТелефону)

•Причина - код, подчиняющийся данному принципу, не изменяется при расширении и поэтому не требует дополнительных трудозатрат.

•Принцип **подстановки Барбары Лисков**: «Пусть q(x) является свойством, верным относительно объектов x некоторого типа T. Тогда q(y) также должно быть верным для объектов y типа S, где S является подтипом типа T»

•«Функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом» – т.е. поведение наследуемых классов не должно противоречить поведению, заданному базовым классом, то есть поведение наследуемых классов должно быть ожидаемым для кода, использующего переменную базового типа.

•Причина - код, использующий иерархию классов с нарушениями принципа Лисков, помимо оперирования ссылкой на базовый класс, оказывается также вынужден знать и о подклассе. Подобная функция нарушает принцип открытости/закрытости, поскольку она требует модификации в случае появления в системе новых производных классов.

•

•Принцип **разделения интерфейса**: «Клиенты не должны зависеть от методов, которые они не используют.»

•Принцип разделения интерфейсов говорит о том, что слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы клиенты маленьких интерфейсов знали только о методах, которые необходимы им в работе. В итоге, при изменении метода интерфейса не должны меняться клиенты, которые этот метод не используют.

•Причина – устойчивость к изменениям.

•Принцип и**нверсии (обращения) зависимостей**:

–Модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций.

–Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

•Причина - позволяет уменьшить зацепление (coupling) модулей, упростив их модификацию в будущем.

32. Средства имитационного моделирования ИС на основе технологии модельно-ориентированного подхода.

Модельно-ориентированное проектирование (МОП) – эффективный и экономически выгодный способ разработки систем управления и создания встраиваемых систем. Вместо физических прототипов и текстовых спецификаций в модельно-ориентированном проектировании применяется исполняемая модель.

Эта модель используется во всех этапах разработки. При таком подходе можно разрабатывать и проводить имитационное моделирование как всей системы целиком, так и ее компонентов. Автоматическая генерация программного кода позволяет избежать большинства ошибок связанных с человеческим фактором и уменьшить время разработки более чем в два раза.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации.

Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретной Системы.

Типовая ИС в специальной базе метаинформации (информация о способах и методах переработки информации или о том, где найти информацию) – репозитории – содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование программного обеспечения. Таким образом, модельно-ориентированное проектирование ИС предполагает, прежде всего, построение модели объекта автоматизации с использованием специального программного инструментария (например, SAP Business Engineering Workbench (BEW), BAAN Enterprise Modeler). Возможно также создание системы на базе типовой модели ИС из репозитория, который поставляется вместе с программным продуктом и расширяется по мере накопления опыта проектирования информационных систем для различных отраслей и типов производства.

Репозиторий содержит базовую (ссылочную) модель ИС, типовые (референтные) модели определенных классов ИС, модели конкретных ИС предприятий.

Базовая модель ИС в репозитории содержит описание бизнес-функций, бизнес-процессов, бизнес-объектов, бизнес-правил, организационной структуры, которые поддерживаются программными модулями типовой ИС.

Типовые модели описывают конфигурации информационной системы для определенных отраслей или типов производства.

Модель конкретной системы строится либо путем выбора фрагментов основной или типовой модели в соответствии со специфическими особенностями объекта автоматизации (BAAN Enterprise Modeler), либо путем автоматизированной

адаптации этих моделей в результате экспертного опроса (SAP Business Engineering Workbench).
Построенная модель в виде метаописания хранится в репозитории и при необходимости может быть откорректирована. На основе этой модели автоматически осуществляется конфигурирование и настройка информационной системы.

Принципы МОП существенно отличаются от традиционной методологии проектирования. Вместо создания сложных программных кодов разработчики могут применять МОП для улучшения характеристик модели, используя стандартные функциональные блоки с непрерывным и дискретным временем. Построенные таким образом модели вместе с использованием инструментов для моделирования, могут привести к созданию прототипа системы управления, тестированию и верификации программного обеспечения. В некоторых случаях аппаратно-программное моделирование может быть использовано в качестве инструмента проектирования для более быстрого и эффективного тестирования динамических воздействий на систему, в отличие от традиционного метода проектирования.
Преимущества МОП перед традиционным подходом проектирования:
− МОП предоставляет общую среду разработки, что способствует взаимодействию группы разработчиков в процессе анализа данных и проверки системы;
− инженеры могут найти и исправить ошибки на ранних стадиях проектирования системы, когда затраты времени и финансовые последствия изменения системы сводятся к минимуму;
− МОП способствует повторному использованию моделей для улучшения системы и создания производных систем с расширенными возможностями.

Модельно-ориентированное проектирование (МОП) способствует представлению проекта на основании требований, а также улучшенной степени интеграции и повторному использованию на этапах концептуального и детализированного моделирования и проектирования [42].

Составления требований. Составление спецификации по требованиям – это процесс анализа и документирования требований и ограничений, которым должна удовлетворять разрабатываемая система.
Проведение валидации требований перед переходом к рабочему проекту

большое количество времени следует потратить на получение, анализ и верификацию требований. В случае новых или комплексных приложений верификация будет включать в себя моделирование и быстрое прототипирование для проверки правильности и полноты требований.

Этап проектирования. Проектирование – это процесс определения архитектуры и интерфейсов программного обеспечения и разработки детализированных функций и операций, удовлетворяющих требованиям.

Проектирование концептуальной модели. По ранее собранным требованиям инженер-проектировщик конструирует исполняемую версию проекта. Simulink дает возможность инженерам создавать эти алгоритмические модели в интуитивно понятной графической среде.

Трассируемость требований в концептуальной модели. В рабочем процессе при использовании МОП все элементы концептуального проекта должны трассироваться к требованиям, которые они удовлетворяют.

Верификация концептуальной модели. Концептуальный проект должен анализироваться для верификации выполнения заданных требований.

Этап реализации. Реализация – это процесс перевода разработки во встроенное программное обеспечение, которое можно запускать на целевом аппаратном обеспечении. В МОП перевод из разработки во встроенное программное обеспечение автоматизирован за счет генерации кода, которая значительно снижает количество ошибок и экономит время на написание кода.

Этап верификации и валидации. При МОП модели и имитационное моделирование используются для ранней и непрерывной верификации и валидации разработок на протяжении всего процесса разработки.

При использовании МОП, разработка представляется в виде модели и в завершение процесса демонстрируется, что модель соответствует требованиям, и код, сгенерированный из модели (реализация) также удовлетворяет требованиям.

Верификация действительно нужных элементов. Первый шаг по внедрению верификации и валидации в проект с использованием модельно-ориентированного проектирования состоит в том, чтобы оценить, какие действия по верификации и валидации необходимы для одновременного удовлетворения требований и временной экономии за счет МОП.

33. Основные подходы в имитационном моделировании ИС и соответствующие технологии.

В современной теории имитационного моделирования существуют четыре основных подхода [62]:

− моделирование динамических систем (системы имитационного моделирования: MATLAB Simulink, VinSim и др.) похоже это то чем мы занимались у Безуглой на СГМ;

− дискретно-событийное моделирование (GPSS, Arena, eMPlant, AutoMod, PROMODEL, Enterprise Dynamics, FlexSim и др.); - а это с Кротовым, ну жпсс же

− системная динамика (СИМ: VenSim, PowerSim, iThink, и др.);

− агентное моделирование (системы имитационного моделирования AnyLogic, Swarm, Repast и др.).

Системная динамика и дискретно-событийное (процессное) моделирование, под которым мы понимаем любое развитие идей GPSS — это традиционные устоявшиеся подходы, агентное моделирование — относительно новый. Системная динамика оперирует в основном с непрерывными во времени процессами, тогда как дискретно-событийное и агентное моделирование — с дискретными.

Системная динамика и дискретно-событийное моделирование исторически преподаются совершенно разным группам студентов. СД чаще преподается студентам из области менеджмента, ДС — инженерам по организации производства и инженерам-разработчикам систем управления. В результате возникли два практически не пересекающихся сообщества, которые почти никак не общаются друг с другом.

Агентное моделирование до недавнего времени было строго академическим направлением. Однако, растущий спрос на глобальную оптимизацию со стороны бизнеса заставил ведущих аналитиков обратить внимание именно на агентное моделирование и его объединение с традиционными подходами с целью получения более полной картины взаимодействия сложных процессов различной природы. Так родился спрос на программные платформы, позволяющие интегрировать различные подходы.[5]

Теперь рассмотрим подходы имитационного моделирования на шкале уровня абстракции. Системная динамика, заменяя индивидуальные объекты их агрегатами, предполагает наивысший уровень абстракции. Дискретно-событийное моделирование работает в низком и среднем диапазоне. Что же касается агентного моделирования, то оно может применяться практически на любом уровне и в любых масштабах. Агенты могут представлять пешеходов, автомобили или роботов в физическом пространстве.

Для всех трех видов - AnyLogic, для дискретно-событийного - GPSS, Simulink тоже вроде для всего.

· Если вы располагаете данными о индивидуальных объектах – используйте агентное моделирование

· Если система может быть описана как процесс – используйте дискретно-событийное моделирование

· Если у вас есть информация только о глобальных зависимостях – используйте системную динамику

· Если системе присущи все эти особенности – комбинируйте различные методы.

34. Средства модельно-ориентированного подхода (анализа): MathLab Sumulink, VenSim, AnyLogic.

Модельно-ориентированное проектирование (МОП) – эффективный и экономически выгодный способ разработки систем управления и создания встраиваемых систем. Вместо физических прототипов и текстовых спецификаций в модельно-ориентированном проектировании применяется исполняемая модель.

Эта модель используется во всех этапах разработки. При таком подходе можно разрабатывать и проводить имитационное моделирование как всей системы целиком, так и ее компонентов. Автоматическая генерация программного кода позволяет избежать большинства ошибок связанных с человеческим фактором и уменьшить время разработки более чем в два раза.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации.

AnyLogic – гибкий инструмент имитационного моделирования, который предоставляет множество путей для решения Вашей задачи.

· Если вы располагаете данными о индивидуальных объектах – используйте агентное моделирование

· Если система может быть описана как процесс – используйте дискретно-событийное моделирование

· Если у вас есть информация только о глобальных зависимостях – используйте системную динамику

· Если системе присущи все эти особенности – комбинируйте различные методы.

Лидеры в своих отраслях выбирают AnyLogic за простоту в работе, визуальную среду разработки моделей, гибкость и возможности 3D-анимации. Строить модели в AnyLogic легко благодаря специализированным библиотекам для различных отраслей применения, drag-n-drop интерфейсу, использованию языка Java для неограниченного расширения функциональности моделей, а также возможности создавать агентные, дискретно-событийные, системно-динамические и гибридные модели в одном ПО.

Графическая среда моделирования AnyLogic включает в себя следующие элементы:

- **Stock & Flow Diagrams** (диаграмма потоков и накопителей) применяется при разработке моделей, используя метод системной динамики.
- **Statecharts** (карты состояний) в основном используется в агентных моделях для определения поведения агентов. Но также часто используется в дискретно-событийном моделировании, например для симуляции машинных сбоев.
- **Action charts** (блок-схемы) используется для построения алгоритмов. Применяется в дискретно-событийном моделировании (маршрутизация звонков) и агентном моделировании (для логики решений агента).
- **Process flowcharts** (процессные диаграммы) основная конструкция, используемая для определения процессов в дискретно-событийном моделировании.

Среда моделирования также включает в себя: низкоуровневые конструкции моделирования (переменные, уравнения, параметры, события и т.п), формы

представления (линии, квадраты, овалы и т.п), элементы анализа (базы данных, гистограммы, графики), стандартные картинки и формы экспериментов.

Среда моделирования AnyLogic поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов с моделью, включая различные виды анализа — от анализа чувствительности до оптимизации параметров модели относительно некоторого критерия.

Simulink – это графическая среда имитационного моделирования, позволяющая при помощи блок-диаграмм в виде направленных графов, строить динамические модели, включая дискретные, непрерывные и гибридные, нелинейные и разрывные системы.

С помощью него на стадии концептуального проектирования создаются модели всей системы, включающие алгоритмы и внешнюю среду. Эти модели можно мимур=лировать и анализировать на протяжении всего процесса проектирования для обеспечения соответствия алгоритмов и спецификаций, на основании нот. Разрабатываются алгоритмы. Плючы подхода:

- Раньше находятся ошибки, это дешевле чем сделать, а потом понять что не то
- Результаты проектирования, тесты и анализ можно повторно использовать в течение всего процесса разработки.

Дополнительные пакеты расширения Simulink позволяют решать весь спектр задач от разработки концепции модели до тестирования, проверки, генерации кода и аппаратной реализации. Simulink интегрирован в среду MATLAB, что позволяет использовать встроенные математические алгоритмы, мощные средства обработки данных и научную графику.

Ключевые особенности:

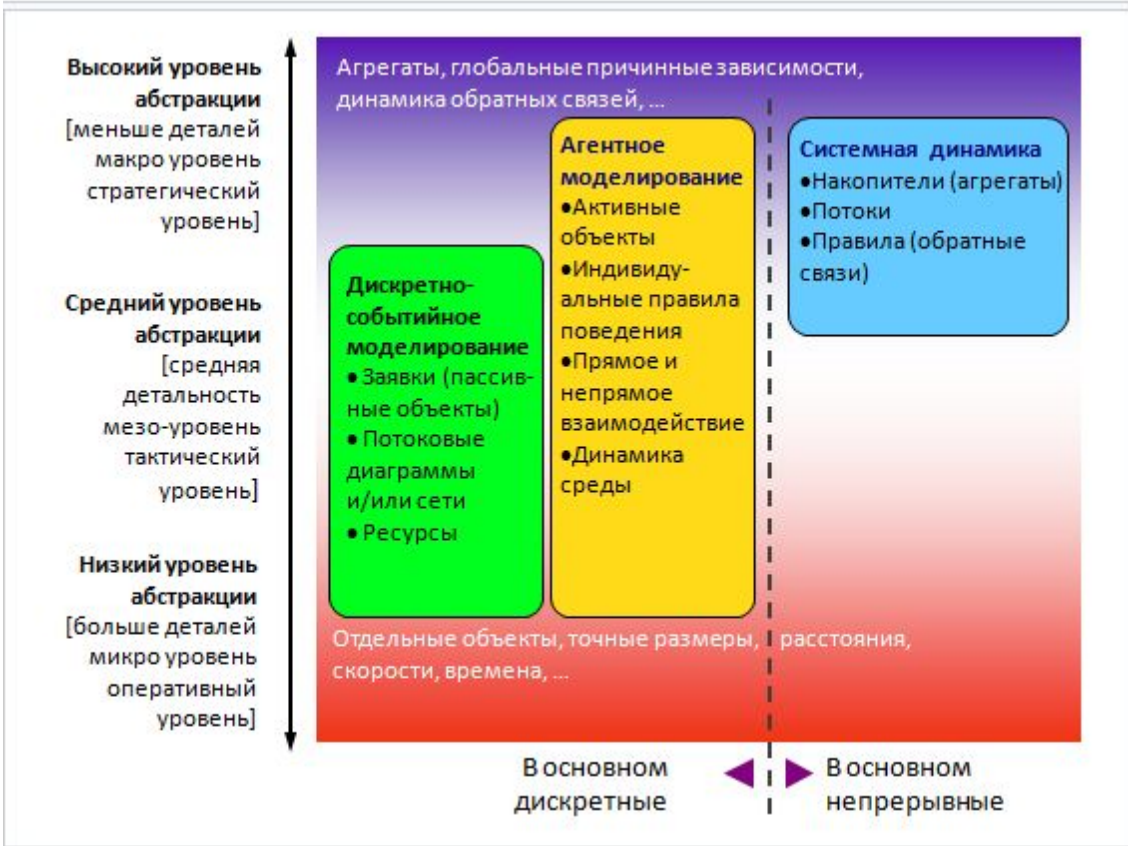
- Интерактивная графическая среда для построения блок-диаграмм
- Расширяемая библиотека готовых блоков
- Удобные средства построение многоуровневых иерархических многокомпонентных моделей
- Средство навигации и настройки параметров сложных моделей - Model Explorer
- Средства интеграции готовых C/C++, FORTRAN,ADA и MATLAB-алгоритмов в модель, взаимодействие с внешними программами для моделирования
- Современные средства решения дифференциальных уравнений для непрерывных, дискретных, линейных и нелинейных объектов (в т.ч. с гистерезисом и разрывами)
- Имитационное моделирование нестационарных систем с помощью решателей с переменным и постоянным шагом или методом управляемого из MATLAB пакетного моделирования
- Удобная интерактивная визуализация выходных сигналов, средства настройки и задания входных воздействий
- Средства отладки и анализа моделей
- Полная интеграция с MATLAB, включая численные методы, визуализацию, анализ данных и графические интерфейсы

Пакет **Vensim** представляет собой инструмент для визуального моделирования, поддерживающий разработку концептуальной модели, документирование, собственно моделирование, анализ результатов и оптимизацию моделей динамических систем (всмысле системная динамика). Он позиционируется на рынке программных продуктов как простое и гибкое средство для построения имитационных моделей систем с причинно-следственными связями, фондами и потоками. Следует отметить, что Vensim существует и в версии для академического использования в образовательных целях. Пакет имеет графический редактор для построения с помощью мыши классических форрестеровских моделей, Equation Editor для завершения формирования модели, а также развитые средства визуализации поведения модели.

35. Инструмент для разработки и исследования имитационных моделей AnyLogic: панели, окна, редакторы.

AnyLogic — программное обеспечение для имитационного моделирования, разработанное российской компанией The AnyLogic Company (бывшая «Экс Джей Текнолоджис», англ. XJ Technologies). Инструмент обладает современным графическим интерфейсом и позволяет использовать язык Java для разработки моделей[1]. Версия AnyLogic PLE доступна бесплатно для образовательных целей и самообучения.[2]

Методы имитационного моделирования



Подходы имитационного моделирования на шкале уровня абстракции

Модели AnyLogic могут быть основаны на любой из основных парадигм имитационного моделирования: дискретно-событийном моделировании, системной динамике и агентном моделировании.

Системная динамика и дискретно-событийное (процессное) моделирование, под которым мы понимаем любое развитие идей GPSS — это традиционные устоявшиеся подходы, агентное моделирование — относительно новый. Системная динамика оперирует в основном с непрерывными во времени процессами, тогда как дискретно-событийное и агентное моделирование — с дискретными.

Системная динамика и дискретно-событийное моделирование исторически преподаются совершенно разным группам студентов. СД чаще преподается студентам из области менеджмента, ДС — инженерам по организации производства и инженерам-разработчикам систем управления. В результате возникли два практически не пересекающихся сообщества, которые почти никак не общаются друг с другом.

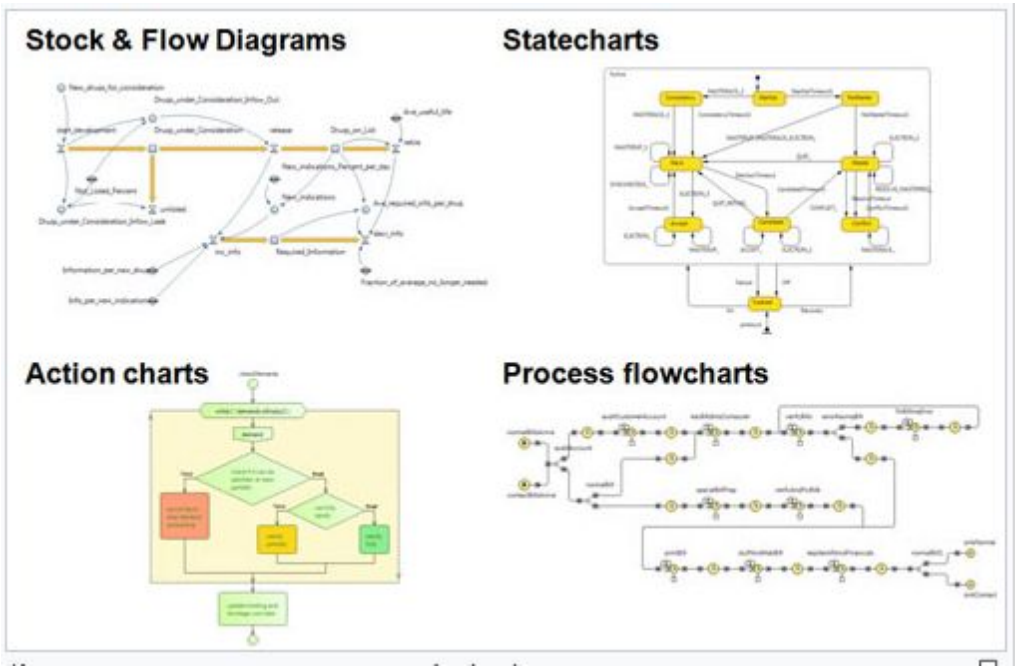
Агентное моделирование до недавнего времени было строго академическим направлением. Однако, растущий спрос на глобальную оптимизацию со стороны бизнеса заставил ведущих аналитиков обратить внимание именно на агентное моделирование и его объединение с традиционными подходами с целью получения более полной картины взаимодействия сложных процессов различной природы. Так родился спрос на программные платформы, позволяющие интегрировать различные подходы.[5]

Теперь рассмотрим подходы имитационного моделирования на шкале уровня абстракции. Системная динамика, заменяя индивидуальные объекты их агрегатами, предполагает наивысший уровень абстракции. Дискретно-событийное моделирование работает в низком и среднем диапазоне. Что же касается агентного моделирования, то оно может применяться практически на любом уровне и в любых масштабах. Агенты могут представлять пешеходов, автомобили или роботов в физическом пространстве, клиента или продавца на среднем уровне, или же конкурирующие компании на высоком.

При разработке моделей в AnyLogic можно использовать концепции и средства из нескольких методов моделирования. Например, в агентной модели можно использовать методы системной динамики для представления изменений состояния среды; в непрерывной модели динамической системы можно учесть дискретные события. Например, управление цепочками поставок при помощи имитационного моделирования требует описания участников цепи поставок агентами: производители, продавцы, потребители, сеть складов. При этом производство описывается в рамках дискретно-событийного (процессного) моделирования, где продукт или его части — это заявки, а автомобили, поезда, штабелёры — ресурсы. Сами поставки представляются дискретными событиями, но при этом спрос на товары может описываться непрерывной системно-динамической диаграммой. Возможность смешивать подходы позволяет описывать процессы реальной жизни, а не подгонять процесс под доступный математический аппарат.

Возможности программы

Среда моделирования



Конструкции среды моделирования AnyLogic

Графическая среда моделирования AnyLogic включает в себя следующие элементы[1]:

- **Stock & Flow Diagrams** (диаграмма потоков и накопителей) применяется при разработке моделей, используя метод системной динамики.
- **Statecharts** (карты состояний) в основном используется в агентных моделях для определения поведения агентов. Но также часто используется в дискретно-событийном моделировании, например для симуляции машинных сбоев.
- **Action charts** (блок-схемы) используется для построения алгоритмов. Применяется в дискретно-событийном моделировании (маршрутизация звонков) и агентном моделировании (для логики решений агента).
- **Process flowcharts** (процессные диаграммы) основная конструкция, используемая для определения процессов в дискретно-событийном моделировании.

Среда моделирования также включает в себя: низкоуровневые конструкции моделирования (переменные, уравнения, параметры, события и т.п), формы представления (линии, квадраты, овалы и т.п), элементы анализа (базы данных, гистограммы, графики), стандартные картинки и формы экспериментов.

Среда моделирования AnyLogic поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов с моделью, включая различные виды анализа — от анализа чувствительности до оптимизации параметров модели относительно некоторого критерия.

36. Гибкие методологии проектирования (agile-методы).

Agile – это семейство методологий разработки программного обеспечения, для которых характерны:

− итеративный процесс разработки;

− динамичное формирование требований и их реализация;

− тесная взаимосвязь, активное взаимодействие, как между абсолютно всеми участниками команды разработчиков, так и между командой и заказчиком.

Agile Manifesto содержит 4 основные идеи и 12 принципов.

Основные идеи:

*Люди и взаимодействие **важнее** процессов и инструментов*

*Работающий продукт **важнее** исчерпывающей документации*

*Сотрудничество с заказчиком **важнее** согласования условий контракта*

*Готовность к изменениям **важнее** следования первоначальному плану*

Основные принципы:

− наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения;

− изменение требований приветствуется, даже на поздних стадиях разработки;

− работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев;

− на протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе;

− над проектом должны работать мотивированные профессионалы;

− непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды;

− работающий продукт – основной показатель прогресса;

− инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно;

− постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта;

− простота – искусство минимизации лишней работы – крайне необходима;

− самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд;

− команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Примечательно, что Agile Manifesto не содержит практических советов.

Agile Modeling (AM) – это набор понятий, принципов и приемов (практик), позволяющих быстро и просто выполнять моделирование и документирование в проектах разработки программного обеспечения.

AM описывает стиль моделирования, который позволит повысить качество и сократить сроки. AM не является технологическим процессом. Это не детальная инструкция по проектированию, он не содержит описаний, как строить диаграммы на UML. AM сосредоточен на эффективном моделировании и документировании. Он не охватывает программирование и тестирование, хотя в нем и говорится о проверке модели кодом и рассматривается тестируемость моделей. AM также не включает вопросы управления проектом, развертывания и сопровождения системы.

AM должен рассматриваться как дополнение к существующим методам, а не самостоятельная технология. Этот метод должен использоваться для повышения эффективности труда разработчиков, использующих процессы eXtreme Programming (XP), Dynamic Systems Development Method (DSDM), или RUP.

37. Понятие CASE-технологии проектирования ИС. Основные принципы CASE - технологии.

Классификация по уровню проектирования в жизненном цикле создания ИС:

- средства верхнего уровня (Upper CASE) – анализ предметной области, определение места ИС в контуре бизнес-системы;
- средства среднего уровня (Middle CASE) – разработка архитектуры ИС, создание проектных спецификаций;
- средства нижнего уровня (Lower CASE) – поддержка разработки программного обеспечения.

Классификация по типам отражает функциональную ориентацию CASE- средств на те или иные процессы ЖЦ и включает следующие типы:

- *средства анализа* (соответствуют Upper CASE), предназначенные для построения и анализа моделей предметной области – (CA ERwin Process Modeler ARIS);
- *средства анализа и проектирования* (соответствуют Middle CASE), поддерживающие наиболее распространенные методологии проектирования, которые используются для создания проектных спецификаций. (CASE-аналитик, ARIS);
- *средства проектирования баз данных* (соответствуют Middle CASE), обеспечивающие моделирование данных и генерацию схем баз данных для наиболее распространенных СУБД: ER-win, Data Base Designer, ARIS Tooleset;
- *средства разработки приложений* (соответствуют Lower CASE) - средства 4GL – SQL Windows (Gupta), Delphi (Borland) и др. и генераторы кода, входящие в состав Vantage Team Builder, PROIV и частично – в Silverrun;
- *средства реинжиниринга*, обеспечивающие анализ программных кодов и схем БД и формирование на их основе различных моделей и проектных спецификаций: ER-win, ARIS Tooleset, Designer/2000 и т.д. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства.

Классификация по степени интегрированности выделяет:

- локальные CASE-средства – применяются для анализа системы и разработки автоматизированных рабочих мест, поддерживают 1, 2 типа моделей и методов (CASE-аналитик, Design/IDEF);
- малые интегрированные CASE-средства, используются для создания небольших ИС, поддерживают несколько типов моделей и методов (ER-win, BPwin, Silverran);
- средние интегрированные CASE-средства – поддерживают от 4-15 моделей и методов. В этой категории Rational Rose, Designer/2000;

крупные интегрированные CASE-средства, поддерживаются свыше 15 типов моделей и методов (семейство программных продуктов ARIS).

38. [Это не то, должно быть из методы по МИСПИСИТ, стр.98] Классификация CASE-средств, стратегия их выбора.

Современные CASE-системы классифицируются по следующим признакам:

1) По поддерживаемым методологиям проектирования: функционально (структурно)-ориентированные, объектно-ориентированные и комплексно-ориентированные (набор методологий проектирования);

2) По поддерживаемым графическим нотациям построения диаграмм: с фиксированной нотацией, с отдельными нотациями и наиболее распространенными нотациями;

3) По степени интегрированности: tools (отдельные локальные средства), toolkit (набор неинтегрированных средств, охватывающих большинство этапов разработки ЭИС) и workbench (полностью интегрированные средства, связанные общей базой проектных данных - репозиторием);

4) По типу и архитектуре вычислительной техники: ориентированные на ПЭВМ, ориентированные на локальную вычислительную сеть (ЛВС), ориентированные на глобальную вычислительную сеть (ГВС) и смешанного типа;

5) По режиму коллективной разработки проекта: не поддерживающие коллективную разработку, ориентированные на режим реального времени разработки проекта, ориентированные на режим объединения подпроектов;

6) По типу операционной системы (ОС): работающие под управлением WINDOWS 3.11 и выше; работающие под управлением UNIX и работающие под управлением различных ОС (WINDOWS, UNIX, OS/2 и др.).

Рассмотрим классификацию Case-средств по типам и категориям.

Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ и включает следующие типы:

1. Средства анализа и проектирования, предназначенные для построения и анализа как моделей деятельности организации (предметной области), так и моделей проектируемой системы.

К таким средствам относятся BPwin (PLATINUM technology), Silverrun (Silverrun Technologies), Oracle Designer (Oracle), Rational Rose (Rational Software), Paradigm Plus (PLATINUM technology), Power Designer (Sybase), System Architect (Popkin Software).

Их целью является определение системных требований и свойств, которыми система должна обладать, а также создание проекта системы,

удовлетворяющей этим требованиям и обладающей соответствующими свойствами. Выходом таких средств являются спецификации компонентов системы и их интерфейсов, алгоритмов и структур данных.

2. Средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL – Structured Query Language – структурированном языке запросов) для наиболее распространенных СУБД. Средства проектирования баз данных имеются в составе таких CASE-средств, как Silverrun, Oracle Designer, Paradigm Plus, Power Designer. Наиболее известным средством, ориентированным только на проектирование БД, является ERwin (PLATINUM technology);

3. Средства управления требованиями, обеспечивающие комплексную поддержку разнородных требований к создаваемой системе.

Примерами таких средств являются RequisitePro (Rational Software) и DOORS – Dynamic Object-Oriented Requirements System – динамическая объектно-ориентированная система управления требованиями (Quality Systems and Software Inc.);

4. Средства управления конфигурацией ПО – PVCS (Merant), ClearCase (Rational Software) и др.;

5. Средства документирования.

Наиболее известным из них является SoDA – Software Document Automation – автоматизированное документирование ПО (Rational Software);

6. Средства тестирования.

Наиболее развитым на сегодняшний день средством является Rational Suite TestStudio (Rational Software) набор продуктов, предназначенных для автоматического тестирования приложений;

7. Средства управления проектом – Open Plan Professional (Welcom Software), Microsoft Project 98 и др.;

8. Средства реверсного инжиниринга, предназначенные для переноса существующей системы ПО в новую среду. Они обеспечивают 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 таких CASE-средств, как Silverrun, Oracle Designer, Power Designer, ERwin. Анализаторы программных кодов имеются в составе Rational Rose и Paradigm Plus.

Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи (tools), набор частично интегрированных средств, охватывающих большинство процессов ЖЦ ПО (toolkit), и полностью интегрированные средства, поддерживающие весь ЖЦ ПО и связанные общим репозиторием.

Помимо этого, CASE-средства можно также классифицировать по применяемым структурным или объектно-ориентированным методам анализа и проектирования ПО.

На сегодняшний день российский рынок программного обеспечения располагает практически всеми перечисленными выше средствами.

39. [Кривенько] Технологии планирования проектов по стандарту PMBOK Guide – Руководству к своду знаний по управлению проектами.

Есть кусок ответа альтернативной группы, если это не нравится, можно заменить. Но там тоже фигня, по-моему

Из курса «Управление IT-проектами» известно, что в самом общем виде методология проектного менеджмента определяет и формализует процедуры, методы и инструменты реализации пяти групп управленческих процессов (согласно стандарту PMBOK Guide – Руководству к своду знаний по управлению проектами [50]):

- инициации проекта;
- планирования;
- организации исполнения;
- контроля исполнения;
- завершения проекта.

Инициация проекта – процесс управления проектом, результатом которого является авторизация и санкционирование начала проекта или очередной фазы его жизненного цикла.

Инициация проекта может включать следующие процедуры:

- разработка концепции проекта:
- анализ проблемы и потребности в проекте;
- сбор исходных данных;
- определение целей и задач проекта;
- рассмотрение альтернативных вариантов проекта.
- рассмотрение и утверждение концепции.
- принятие решения о начале проекта:
- определение и назначение менеджера проекта;
- принятие решения об обеспечении ресурсами выполнения первой фазы проекта.

Планирование проекта – непрерывный процесс, направленный на определение и согласование наилучшего способа действий для достижения поставленных целей проекта с учетом всех факторов его реализации. Основным результатом этого этапа является план проекта. Однако, процесс планирования не завершается разработкой и утверждением первоначального плана проекта. В ходе осуществления проекта могут происходить изменения как внутри проекта, так и во внешнем окружении, которые требуют уточнения планов, а часто значительного перепланирования. Поэтому процессы планирования могут осуществляться на протяжении всего жизненного цикла проекта, начиная с предварительного укрупненного плана в составе концепции проекта, и заканчивая детальным планом работ завершающей фазы проекта.

Планирование предметной области проекта включает следующие задачи и процедуры:

− анализ текущего состояния и уточнение целей и результатов проекта;
− уточнение основных характеристик проекта;
− подтверждение и уточнение критериев успеха и неудач проекта;
− анализ и корректировку ограничений и допущений, принятых на стадии инициации проекта;
− выбор критериев оценки промежуточных и окончательных результатов создания проекта;
− построение структурной декомпозиции предметной области проекта;
− планирование времени проекта.

Планирование времени проекта. Согласованная работа всех участников проекта организуется на основе календарных планов или расписаний работ проекта, основными параметрами которых являются: сроки выполнения, ключевые даты, продолжительности работ и др. Календарными планами называют проектно-технологические документы, проекта.

Планирование проекта по временным параметрам заключается в составлении различных календарных планов (расписаний работ), удовлетворяющих всем требованиям и ограничениям проекта и его частей.

Календарные планы составляются на весь жизненный цикл проекта и его этапы, для различных уровней управления и участников проекта.

Календарное планирование проекта состоит из следующих этапов:

Этап 1. Составление структурной декомпозиции работ

Структурная декомпозиция работ (СДР) – графическое изображение иерархической структуры всех работ проекта.

Структурная декомпозиция работ проекта (Work Breakdown Structure - WBS) – разбиение проекта на составные части (элементы, модули, работы и др,), необходимые и достаточные для его эффективного планирования и контроля.

Структурная декомпозиция работ:

```
Цель
 |---Задача1
 |    |---Работа 1.1.
 |    |    |---Работа 1.1.1.
 |    |    |---Работа 1.1.2.
...
 |    |    `---Работа 1.1.N.
 |    |---Работа 1.2.
…
 |    `---Работа 1.N.
 |---Задача2
...
 `---Задача N
```

СДР является центральным инструментом определения работ, которые должны выполняться в рамках проекта. Описание работ (пакетов работ) должно включать:

− содержание работ;

− предполагаемые результаты;

− концептуальные границы интегрированного планирования и управления;

− последовательные измерения и оценки степени выполнения проекта.

При построении иерархии структуры работ (ИСР) необходимо соблюдать следующие правила:

− работы нижнего уровня являются способом достижения работ верхнего уровня;

− у каждой родительской работы может иметься несколько дочерних работ, достижение которых автоматически обеспечивает достижение родительской работы;

− у каждой дочерней работы может быть только одна родительская работа;

− декомпозиция родительской работы на дочерние производится по одному критерию, в качестве которого могут выступать: компоненты результатов и продуктов проекта, этапы жизненного цикла проекта, ресурсы и функциональные виды деятельности, а также элементы организационной структуры;

− на одном уровне дочерние работы, декомпозирующие родительскую должны быть равнозначны. В качестве критерия равнозначности могут выступать: объем и время выполнения работ, пр.;

− при построении иерархической структуры работ на различных уровнях можно и следует применять различные критерии декомпозиции.

− последовательность критериев декомпозиции работ следует выбирать таким образом, чтобы как можно большая часть зависимостей и взаимодействий между работами оказалась на самых нижних уровнях ИСР. На верхних уровнях работы должны быть автономны.

− декомпозиция работ прекращается тогда, когда работы нижнего уровня удовлетворяют следующим условиям:

− работы ясны и понятны менеджеру и участникам проекта (являются элементарными);

− понятен конечный результат работы и способы его достижения;

− временные характеристики и ответственность за выполнение работ могут быть однозначно определены.

Этап 2. Определение списка работ проекта на основе структурной декомпозиции проекта.

Этап 3. Определение последовательности выполнения работ и их взаимосвязей с помощью организационно-технологических моделей. Уточнение временных ограничений.

Этап 4. Определение продолжительности работ.

На данном шаге, необходимо указать продолжительность выполнения каждой работы по проекту. Эта продолжительность может быть рассчитана, исходя из нормативов, может быть указана, исходя из личного опыта.

Часто невозможно однозначно определить продолжительность той или иной работы. В таком случае можно использовать методы PERT [51].

Дальше идёт метод PERT вопрос 40

Этап 5 Составление сетевой диаграммы проекта.

Сетевая диаграмма – графическое отображение работ проекта и зависимостей между ними.

Цель методов сетевого планирования – сократить до минимума продолжительность проекта.

Как правило, сетевая диаграмма представляется в виде графа, в котором вершинами являются проектные работы, а взаимосвязь и последовательность работ отображается соединительными линиями, как показано на рисунке.

Работа в сетевой диаграмме отображается в виде прямоугольника, в котором содержится информация о работе: код в сетевой диаграмме работ (например:1.1.), наименование и продолжительность работы.

Стрелками, обозначается последовательность и взаимосвязь работ. Взаимосвязи также могут характеризоваться временными показателями. Например, работы 1.1. и 1.2. связаны соединительной стрелкой со значением «+1д», Это означает, что работа 1.2. должна начаться через день после того как начнется работа 1.1. А на стрелке, соединяющей работы 1.2. и 3. стоит значение «-2д». Это означает, что работа 3 должна начаться за два дня до окончания работы 1.2. Если на стрелке нет дополнительной информации, как например на стрелке, соединяющей работы 1.1. и 2., то это означает, что работа 2 начинается сразу как закончится работа 1.1.

Для оптимизации расписания работ в проекте могут быть использованы различные методы. Одним из них является метод критического пути (МКП).

Критический путь – максимальный по продолжительности полный путь в сети; работы, лежащие на этом пути, также называются критическими. Критическая работа – работа, увеличение продолжительности которой, влечет увеличение продолжительности всего проекта. На рисунке они отображены красным цветом.

Некритические работы имеют временной резерв. В случае, если этот временной резерв исчерпан в процессе реализации работы, она становится критической, т.е. продолжительность ее выполнения начинает влиять на продолжительность всего проекта.

Этап 6 Составление диаграммы Ганта

Диаграмма Ганта – горизонтальная линейная диаграмма, на которой работы проекта представляются протяженными во времени отрезками, характеризующимися временными и другими параметрами. Работы проекта отображается в виде прямоугольников, однако, в отличие от сетевой диаграммы, в диаграмме Ганта, длина прямоугольника соответствует продолжительности работы, Стрелки также характеризуют последовательность и взаимосвязь работ. При необходимости, можно дополнять диаграмму информацией о стоимости работ, об их исполнителях.

40. Метод PERT (Program Evaluation and Review Technique) при проектировании ИС.

Метод PERT – метод событийного сетевого анализа, используемый для определения длительности проекта при наличии неопределенности в оценке продолжительностей индивидуальных операций.

PERT основан на методе критического пути, длительность операций в котором рассчитывается как взвешенная средняя оптимистического, пессимистического и

ожидаемого прогнозов. PERT рассчитывает стандартное отклонение даты завершения от длительности критического пути.

Продолжительность работы рассчитывается в данном случае как средневзвешенная средняя оптимистического, пессимистического и ожидаемого прогнозов.

ПРИМЕР «Поиск практического материала для написания дипломного проекта»

Указываем оптимистический прогноз продолжительности выполнения данной работы, пессимистический прогноз, вероятности.

	Оптимистический	**Наиболее вероятный**	**Пессимистический**
Прогнозное значение	3д	10д	30д
Вероятность осуществления прогноза	0,2	0,6	0,2
Взвешенное значение	0,6д	6д	6д
Сумма взвешенных значений	12,6д		

Необходимо перемножить значение каждого прогноза на его вероятность и полученные величины сложить.

Таким образом, получится средневзвешенное значение продолжительности выполнения работы.

3д * 0,2 + 10д * 0,6 + 30д * 0,2 = 0,6д + 6д +6д = 12,6д

Примерная продолжительность работы около 13 дней.

41. [Добавлено, удалить повторения] Средства планирования проектов: OpenProj, Microsoft Project.

Microsoft Project (или MSP) — программа управления проектами, разработанная и продаваемая корпорацией Microsoft.

Microsoft Project создан, чтобы помочь менеджеру проекта в разработке планов, распределении ресурсов по задачам, отслеживании прогресса и анализе объёмов работ. Microsoft Project создаёт расписания критического пути. Расписания могут быть составлены с учётом используемых ресурсов. Цепочка визуализируется в диаграмме Ганта.

Особенности:

1. Встроенные шаблоны.

Встроенные настраиваемые шаблоны, созданные на основе передовых отраслевых наработок, позволяют вам сразу встать на правильный путь: вам не придется создавать планы проектов с нуля.

2. Планирование проектов

Знакомые функции планирования, такие как диаграммы Ганта и раскрывающиеся меню с готовым списком вариантов, помогают сократить время на обучение и упростить планирование проектов.

3. Готовые отчеты

Общие отчеты в масштабах всей организации помогут вам синхронизировать усилия всех участников. Отчеты могут содержать любую информацию: от диаграмм выработки до финансовых данных, и доступны на всех устройствах.

4. Наборы временных шкал

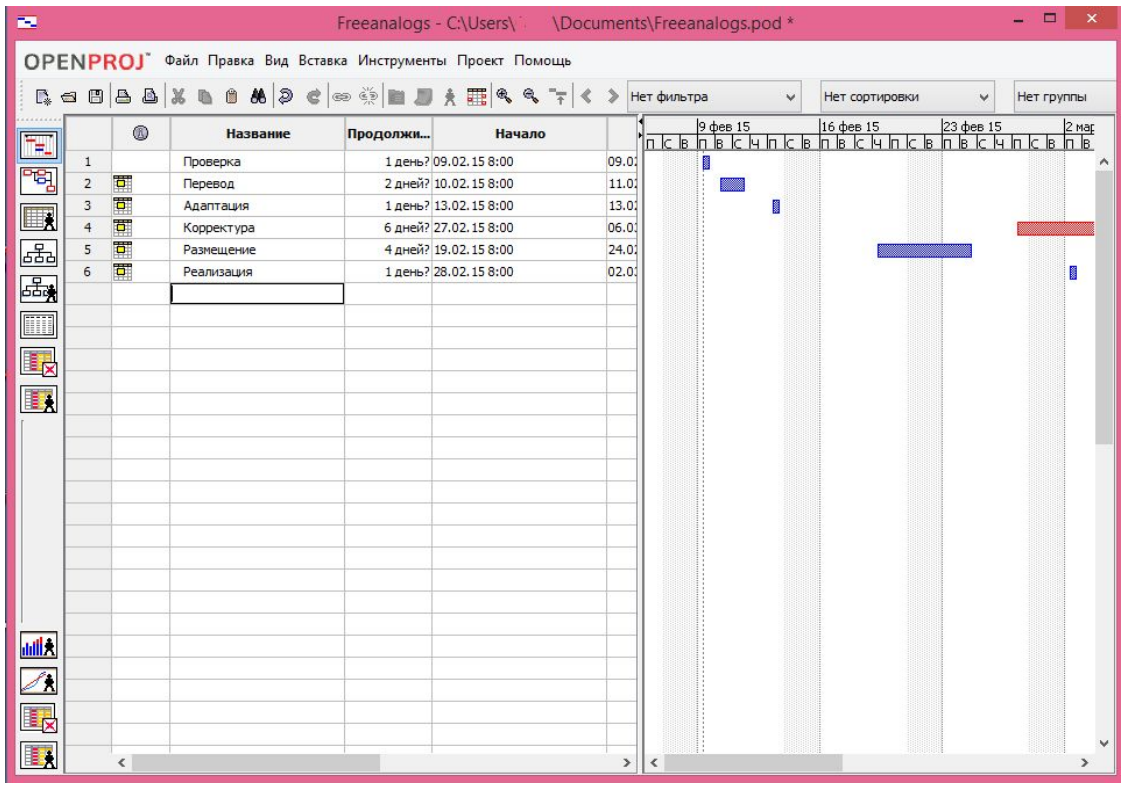
Вы можете мгновенно оценивать все процессы в рамках проектов: от задач до ключевых этапов. Настраивайте временные шкалы, выводя на них интересующие вас данные, и демонстрируйте результаты заинтересованным лицам.

OpenProj - бесплатный аналог Microsoft Project. Данное кроссплатформенное программное обеспечение предназначено для планирования проектов и является очень хорошей заменой платного ПО..

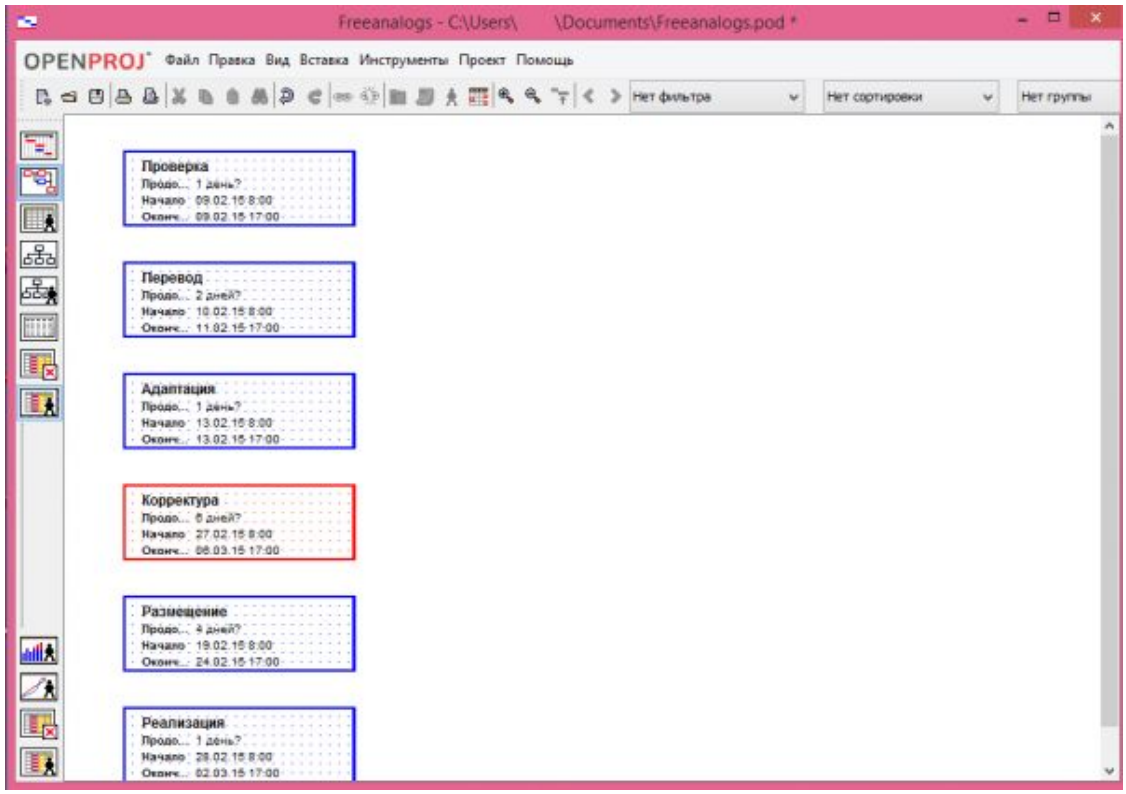
Однако стоит отметить, что программа не сможет посоревноваться со своим прототипом в виде Microsoft Project так как программе есть куда стремиться и развиваться. Если учесть отзывы пользователей, то можно сделать вывод, что программа больше подходит для ознакомительной работы с проектами.

Возможности:

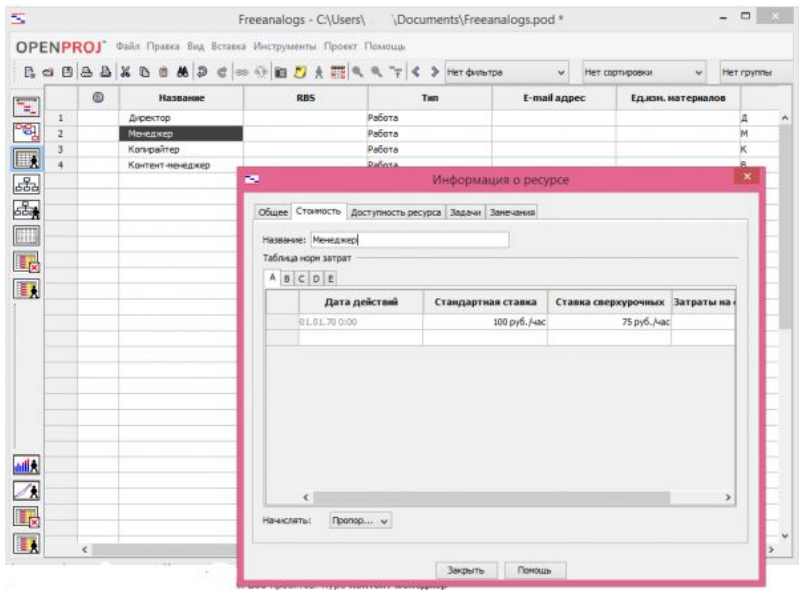
Диаграмма ганта



Сетевой график



Ресурсы



Отчёты

Поддерживается импорт/экспорт документов Microsoft Project

Добавила, завтра удалю то, что повторяется

OpenProj — кроссплатформенное программное обеспечение для управления проектами. Распространяется на условиях лицензии Common Public Attribution License Version 1.0 (CPAL). Позиционируется создателями как открытая замена коммерческому продукту Microsoft Project. Доступна для операционных систем Microsoft Windows, Linux, Unix, Mac OS X.

The OpenProj переведен на французский, испанский, немецкий, португальский, швейцарский, финский, русский, корейский и китайский язык.

Продукт куплен компанией Serena. Сразу после продажи поддержка продукта была прекращена ввиду угрозы судебных исков со стороны Microsoft по поводу копирования интерфейсных решений, а исходный текст продукта выложен бесплатно в Интернет.

Продолжение проекта взяло на себя сообщество Linux по разработке Libre Office. Продукт, выпускаемый группой волонтеров, называется ProjectLibre.

Возможности

§ Диаграмма Ганта
§ Сетевой график
§ Ресурсы
§ Отчёты
§ Поддерживается импорт/экспорт документов Microsoft Project.

Microsoft Project (или *MSP*) — программа управления проектами, разработанная и продаваемая корпорацией Microsoft.

Microsoft Project создан, чтобы помочь менеджеру проекта в разработке планов, распределении ресурсов по задачам, отслеживании прогресса и анализе объёмов работ. Microsoft Project создаёт расписания критического пути. Расписания могут быть составлены с учётом используемых ресурсов. Цепочка визуализируется в диаграмме Ганта.

Состав решения

Под маркой Microsoft Project доступны сразу несколько продуктов и решений:

§ Microsoft Project Standard — однопользовательская версия для небольших проектов

§ Microsoft Project Professional — корпоративная версия продукта, поддерживающая совместное управление проектами и ресурсами, а также управление портфелями проектов с помощью Microsoft Project Server.

§ Microsoft Project Web Access — Web-интерфейс для отчетности о выполнении задач, а также просмотра портфелей проектов
§ Microsoft Project Portfolio Server — продукт для отбора проектов для запуска на основе сбалансированных показателей, вошел в состав Microsoft Project Server с версии MS Project 2010.Начиная с 2013 года Microsoft начинает поставлять облачную версию Microsoft Project Online.

Рыночная позиция, экосистема и конкуренты

§ Primavera — основной конкурент в корпоративном сегменте
§ GanttProject -открытая программа на языке Java, предназначенная для планирования проектов на основе построения диаграмм Ганта и диаграмм типа PERT.
§ OpenProj — кросс-платформенный клон десктопа MS Project без поддержки формул и индикаторов. Куплен компанией Serena и поддержка продукта прекращена из-за возможных судебных претензий Microsoft по копированию интеллектуальной собственности. Тем не менее разработка продукта продолжается в рамках open source community под названием ProjectLibre

§ Basecamp — основной конкурент в сегменте ультра-лёгких решений по управлению поручениями в проектах

§ Другое ПО для управления проектами и сравнение их возможностей на английском

На рынке малых и однопользовательских решений Microsoft Project является де-факто монополистом, зарабатывая на продажах более 900 000 000 долларов в год и имея клиентскую базу в 20.000.000 пользователей[1], что составляет примерно 80 % рынка в сегменте малых и персональных решений. На рынке корпоративных систем традиционным и самым серьёзным соперником Microsoft является Oracle Primavera. По отчётам Gartner Oracle и Microsoft занимают 1-е и 2-е место по продажам в корпоративном сегменте. Данный отчёт Gartner не включает продажи десктопа Microsoft Project, а только решения с Microsoft Project Server.

Партнеры Microsoft и Oracle также серьёзно усиливают Microsoft Project и Oracle Primavera за счёт собственных разработок, так как Microsoft Project и Oracle Primavera являются не только готовыми продуктами, но и платформами для создания отраслевых решений. На этом базируется экосистемы двух основных игроков на рынке корпоративных систем управления проектами. Сильнейшие партнёры Microsoft и Oracle представляют не просто внедрение Microsoft Project или Oracle Primavera, а внедрения отраслевых решений с традиционным фокусом в крупнейших корпоративных заказчиков проектного управления: строительство, проектирование, конструкторские бюро, крупные IT-проекты, машиностроительные проекты, проекты разработки и производства вооружений.

Методология внедрения

Microsoft Project является только инструментом, для внедрения управления проектами необходимо выбрать методологию проектного управления. Как правило методология реализуется через «регламенты» проектного управления и отраслевые доработки MS Project.

Плюсы и минусы Microsoft Project

Имея 20 000 000 пользователей, Microsoft Project является монополистом.

Наиболее очевидным преимуществом продукта является то, что он входит в семейство Microsoft Office. Это обеспечивает следующие плюсы, характерные для всех продуктов MS Office:

§ Такое же малое время обучения пользователей, как и с остальными программами Microsoft Office

§ Богатые возможности по настройке в стиле формул Microsoft Excel (сам продукт выдержан в интерфейсе, максимально приближенном к Microsoft Excel)

§ Возможность адаптировать продукт под свою специфику путём программирования или покупки готовых решений, созданных на базе Visual Basic или Microsoft .Net.

Microsoft стимулирует покупку готовых решений у партнеров через программу Microsoft ISV Royalty, компенсируя клиентам и партнерам разработку отраслевых решений. Также данная программа направлена на уменьшение проблем продукта с технической поддержкой.

42. [ВРЕМЕННО ЗАБИТО] Проблемы и перспективы проектирования ИС: мультизадачность ИС, гетерогенность системы и среды, неопределенность задач и требований, Runtime, On-line проектирование.

Главной проблемой, стоящей в настоящее время перед проектировщиками ИС, является обеспечение быстро расширяющегося сообщества конечных пользователей удобным интерфейсом, т.е. создавать такие ИС, которые позволили бы пользователю выполнять с помощью ЭВМ необходимые действия без глубокого изучения в полном объеме специальной литературы по ВТ.

Современный уровень научно-технического развития выдвигает определенные принципы проектирования ИС, включая и экономические. Актуальность задачи разработки этих принципов вызвана следующим: объекты ИС становятся более крупномасштабными и дорогими, что приводит к удорожанию и увеличению сроков проектирования; ошибки, допущенные в процессе проектирования, приводят к существенным затратам материальных и трудовых ресурсов; растет сложность ИС: возрастает число решаемых задач, простейшие задачи стабилизации уступают место сложным задачам самонастройки системы на оптимум показателей; одновременно с ростом числа задач сокращается допустимое время принятия решений; проектирование начинается и проводится в условиях неопределенности, т.е. при отсутствии в полном объеме информации, необходимой для выбора решений.

Рост спроса на ИС предъявляет требования к самому проектированию: повысить производительность труда при разработке; повысить эффективность сопровождения, т.к. последнее требует больших затрат, чем непосредственно разработка.

Гетерогенность, т.е. объединение в единую инфраструктуру баз данных и других элементов информационной системы, поставляемых различными производителями, является актуальной проблемой при разработке и проектирования любой реальной информационной системы в силу исторических факторов, действующих в обстоятельствах реального бизнеса, а также в силу того, что прикладные системы разной функциональности, необходимые для деятельности компании, могут быть построены на СУБД различных производителей.

Распределенность и гетерогенность информационных ресурсов налагает дополнительные требования к информационным системам:

1. способность систем функционировать в условиях информационной и реализационной неоднородности, распределенности и автономности информационных ресурсов;
2. обеспечение интероперабельности(англ. **interoperability** — способность к взаимодействию), повторного использования неоднородных информационных ресурсов в разнообразных применениях;
3. возможность объединения систем в более сложные, интегрированные образования, основанные на интероперабельном взаимодействии компонентов;
4. осуществлении миграции унаследованных систем в новые системы, соответствующие новым требованиям и технологиям при сохранении их интероперабельности;

5. обеспечение более длительного жизненного цикла систем.

В процессе проектирования ИС могут существовать многие факторы **неопределенности**: неопределенность исходных данных, неопределенность внешней среды, неопределенность, связанная с характером, вариантами и моделью реализации проекта, неопределенность требований, предъявляемых к эффективности ИС.

Основные причины неопределенности параметров проекта:

1. неполнота или неточность проектной информации;
2. ошибки в прогнозировании параметров проекта;
3. ошибки в расчетах параметров проекта. Упрощения при формировании моделей сложных технических или организационно-экономических систем;
4. производственно-технологический риск (риск аварий, отказов оборудования и т. п.);

Неопределенность связана не только с неточным предвидением будущего, но и с тем, что параметры, относящиеся к настоящему или прошлому, неполны, неточны или на момент включения их в проектные материалы еще не измерены.

Хз, надо ли то, что дальше. Взято у Гис и не проверено на соответствие (херота это все, не читайте дальше ;))

привет

Многозада́чность (англ. *multitasking*) — свойство операционной системы или среды выполнения обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких задач. Истинная многозадачность операционной системы возможна только в распределённых вычислительных системах.

Существует 2 типа многозадачности[1]:

- *Процессная многозадачность* (основанная на процессах — одновременно выполняющихся программах). Здесь программа — наименьший элемент управляемого кода, которым может управлять планировщик операционной системы. Более известна большинству пользователей (работа в текстовом редакторе и прослушивание музыки).
- *Поточная многозадачность* (основанная на потоках). Наименьший элемент управляемого кода — поток (одна программа может выполнять 2 и более задачи одновременно).

Многопоточность — специализированная форма многозадачности

Гетероге́нная систе́ма (от греч. ἕτερος — разный; γένω — рождать) — *неоднородная* система, состоящая из однородных частей (фаз), разделённых поверхностью раздела. Однородные части (фазы) могут отличаться друг от друга по составу и свойствам. Число веществ (компонентов), термодинамических фаз и степеней свободы связаны правилом фаз. Фазы гетерогенной системы можно отделить друг от друга механическими методами[1] (отстаиванием, фильтрованием, магнитной

сепарацией и т. п.). Примерами гетерогенных систем могут служить: жидкость — насыщенный пар; насыщенный раствор с осадком; многие сплавы. Твёрдый катализатор в токе газа или жидкости тоже гетерогенная система (гетерогенный катализ). В технике гетерогенной системой является кирпичная и каменная кладка, состоящая из кладочных элементов (кирпича, природных или искусственных камней, бетонных блоков и др.) и строительного раствора.