**Информационные технологии**

1. **Определение и задачи информационной технологии. Структура проекта в области информационных технологий.**

**Информационная технология** — совокупность методов и способов получения, обработки, представления информации, направленных на изменение ее состояния, свойств, формы, содержания и осуществляемых в интересах пользователей.

Можно выделить **три уровня рассмотрения информационных технологий:**

**первый уровень** — теоретический. Основная задача — создание комплекса взаимосвязанных моделей информационных процессов, совместимых параметрически и критериально;

**второй уровень** — исследовательский. Основная задача — разработка методов, позволяющих автоматизировано конструировать оптимальные конкретные информационные технологии;

**третий уровень** — прикладной, который целесообразно разделить на две страты: инструментальную и предметную.

Инструментальная страта (аналог — оборудование, станки, инструмент) определяет пути и средства реализации информационных технологий, которые можно разделить на методические, информационные, математические, алгоритмические, технические и программные.

Предметная страта связана со спецификой конкретной предметной области и находит отражение в специализированных информационных технологиях, например, организационное управление, управление технологическими процессами, автоматизированное проектирование, обучение и др.

Задачи:

1. **Определение и основные характеристики информационного общества.**

**Информационное общество** — это такое общество, в котором производство и потребление информации является важнейшим видом деятельности, а информация признается наиболее значимым ресурсом.

**Основными отличительными признаками информационного общества являются:**

• информационная экономика;

• высокий уровень информационных потребностей всех членов общества и фактическое их удовлетворение для основной массы населения;

• высокая информационная культура;

• свободный доступ каждого члена общества к информации, ограниченный только информационной безопасностью личности, общественных групп и всего общества.

**Информационному обществу присущи:**

• единое информационное пространство;

• доминирование в экономике новых технологических укладов;

• ведущая роль информационных ресурсов в развитии общества;

• возрастание роли инфраструктуры (телекоммуникационной, транспортной, организационной) в системе общественного производства;

• фактическое удовлетворение потребностей общества в информационных продуктах и услугах;

• высокий уровень образования;

• высокая значимость проблем обеспечения информационной безопасности и наличие эффективной системы обеспечения прав граждан и социальных институтов на свободное получение, распространение и использование информации.

**К стратегическим направлениям перехода к информационному обществу относятся:**

• формирование единого информационного пространства региона и интеграция его в информационное пространство России и единое мировое информационное пространство;

• подготовка населения региона к формированию информационного общества и к жизни в этом обществе;

• создание правовых основ перехода к информационному обществу и обеспечение условий их практического применения;

• создание информационной экономики региона;

• осуществление свободного доступа к информации каждому жителю региона при обеспечении информационной безопасности личности, общественных групп и всего населения региона в целом;

• научное сопровождение формирования информационного общества;

• участие в межрегиональном и международном сотрудничестве и разделении труда при формировании информационного общества города, регионов, стран и мира.

1. **Процесс передачи информации. Каналы связи.**

**Передача информации** — физический процесс, посредством которого осуществляется перемещение информации в пространстве.

Данный процесс характеризуется наличием следующих компонентов:

* Источник информации.
* Приёмник информации.
* Носитель информации.
* Среда передачи.

Передача информации — заблаговременно организованное техническое мероприятие, результатом которого становится воспроизведение информации, имеющейся в одном месте, условно называемом "источником информации", в другом месте, условно называемом "приёмником информации". Данное мероприятие предполагает предсказуемый срок получения указанного результата.

"Информация" здесь понимается в техническом аспекте, как осмысленное множество символов, чисел, параметров абстрактных или физических объектов, без достаточного "объёма" которого не могут быть решены задачи управления, выживания, развлечения, совершения преступлений или денежных операций.

Для осуществления п.и. необходимо наличие, с одной стороны, так называемого "запоминающего устройства", или "носителя", обладающего возможностью перемещения в пространстве и времени между "источником" и "приёмником". С другой стороны, необходимы заранее известные "источнику" и "приемнику" правила и способы нанесения и снятия информации с "носителя". С третьей стороны, "носитель" должен продолжать существовать как таковой к моменту прибытия в пункт назначения. (к моменту окончания снятия с него информации "приёмником")

В качестве "носителей" на современном этапе развития техники используются как вещественно-предметные, так и волново-полевые объекты физической природы. Носителями могут быть при определённых условиях и сами передаваемые "информационные" "объекты" (виртуальные носители).

В любом процессе передачи или обмене информацией существует ее *источник* и *получатель*, а сама информация передается по *каналу связи* с помощью *сигналов*: механических, тепловых, электрических и др. В обычной жизни для человека любой звук, свет являются сигналами, несущими смысловую нагрузку. Например, сирена — это звуковой сигнал тревоги; звонок телефона — сигнал, чтобы взять трубку; красный свет светофора — сигнал, запрещающий переход дороги.



В качестве источника информации может выступать живое существо или техническое устройство. От него информация попадает на кодирующее устройство, которое предназначено для преобразования исходного сообщения в форму, удобную для передачи. С такими устройствами вы встречаетесь постоянно: микрофон телефона, лист бумаги и т. д. По каналу связи информация попадает в декодирующее устройство получателя, которое преобразует кодированное сообщение в форму, понятную получателю. Одни из самых сложных декодирующих устройств — человеческие ухо и глаз.

  
В процессе передачи информация может утрачиваться, искажаться. Это происходит из-за различных помех, как на канале связи, так и при кодировании и декодировании информации. С такими ситуациями вы встречаетесь достаточно часто: искажение звука в телефоне, помехи при телевизионной передаче, ошибки телеграфа, неполнота переданной информации, неверно выраженная мысль, ошибка в расчетах. Прием-передача информации могут происходить с разной скоростью. Количество информации, передаваемое за единицу времени, есть *скорость передачи информации* или скорость информационного потока.

Очевидно, эта скорость выражается в таких единицах, как бит в секунду (бит/с), байт в секунду (байт/с), килобайт в секунду (Кбайт/с) и т.д.

Максимальная скорость передачи информации по каналу связи называется п*ропускной способностью канала*.

***Канал связи***– это совокупность средств, предназначенных для передачи сигналов (сообщений).

Существуют различные типы каналов, которые можно классифицировать по различным признакам:

1. ***По типу линий связи:***проводные; кабельные; оптико-волоконные;

линии электропередачи; радиоканалы и т.д.

2***. По характеру сигналов:***непрерывные; дискретные; дискретно-непрерывные (сигналы на входе системы дискретные, а на выходе непрерывные, и наоборот).

3***. По помехозащищенности:***каналы без помех; с помехами.

**Каналы связи характеризуются:**

1. ***Емкость канала***определяется как произведениевремени использования канала *Tк,* ширины спектра частот, пропускаемых каналом*Fк*и динамического диапазона*Dк*., который характеризует способность канала передавать различные уровни сигналов

*Vк = Tк Fк Dк.*

2.***Скорость передачи информации*** – среднее количество информации, передаваемое в единицу времени.

3. ***Пропускная способность канала связи*** – наибольшая теоретически достижимая скорость передачи информации при условии, что погрешность не превосходит заданной величины.

4. ***Избыточность –*** обеспечивает достоверность передаваемой информации (*R* = 0¸1).

***Проводные:***

1. **Проводные**– витая пара (что частично подавляет электромагнитное излучение других источников). Скорость передачи до 1 Мбит/с. Используется в телефонных сетях и для передачи данных.

2. **Коаксиальный кабель.**Скорость передачи 10–100 Мбит/с – используется в локальных сетях, кабельном телевидении и т.д.

3**. Оптико-волоконная.**Скорость передачи 1 Гбит/с.

В средах 1–3 затухание в дБ линейно зависит от расстояния, т.е. мощность падает по экспоненте. Поэтому через определенное расстояние необходимо ставить регенераторы (усилители).

***Радиолинии:***

1. **Радиоканал.**Скорость передачи 100–400 Кбит/с. Использует радиочастоты до 1000 МГц. До 30 МГц за счет отражения от ионосферы возможно распространение электромагнитных волн за пределы прямой видимости. Но этот диапазон сильно зашумлен (например, любительской радиосвязью). От 30 до 1000 МГц – ионосфера прозрачна и необходима прямая видимость. Антенны устанавливаются на высоте (иногда устанавливаются регенераторы). Используются в радио и телевидении.

2. **Микроволновые линии.**Скорости передачи до 1 Гбит/с. Используют радиочастоты выше 1000 МГц. При этом необходима прямая видимость и остронаправленные параболические антенны. Расстояние между регенераторами 10–200 км. Используются для телефонной связи, телевидения и передачи данных.

3. **Спутниковая связь**. Используются микроволновые частоты, а спутник служит регенератором (причем для многих станций). Характеристики те же, что у микроволновых линий.

1. **Модели процессов обработки информации. Централизованная, децентрализованная и смешанная формы обработки.**

При **числовой** обработке используются такие объекты, как переменные, векторы, матрицы, многомерные массивы, константы и т.д. При **нечисловой** обработке объектами могут быть файлы, записи, поля, иерархии, сети, отношения и т.д. Другое отличие заключается в том, что при числовой обработке содержание данных не имеет большого значения, в то время как при нечисловой обработке нас интересуют непосредственные сведения об объектах, а не их совокупность в целом.

**С точки зрения реализации на основе современных достижений вычислительной техники выделяют следующие виды обработки информации:**

• последовательная обработка, применяемая в традиционной фоннеймановской архитектуре ЭВМ, располагающей одним процессором;

• параллельная обработка, применяемая при наличии нескольких процессоров в ЭВМ;

• конвейерная обработка, связанная с использованием в архитектуре ЭВМ одних и тех же ресурсов для решения разных задач, причем если эти задачи тождественны, то это последовательный конвейер, если задачи одинаковые — векторный конвейер.

Принято относить существующие архитектуры ЭВМ с точки зрения обработки информации к одному из следующих классов.

**Архитектуры с одиночным потоком команд и данных (SISD).** К этому классу относятся традиционные фоннеймановские однопроцессорные системы, где имеется центральный процессор, работающий с парами «атрибут — значение».

**Архитектуры с одиночными потоками команд и данных (SIMD).** Особенностью данного класса является наличие одного (центрального) контроллера, управляющего рядом одинаковых процессоров.

**Архитектуры с множественным потоком команд и одиночным потоком данных (MISD). К** этому классу могут быть отнесены конвейерные процессоры.

**Архитектуры с множественным потоком команд и множественным**

**потоком данных (MIMD).** К этому классу могут быть отнесены следующие конфигурации: мультипроцессорные системы, системы с мультобработкой, вычислительные системы из многих машин, вычислительные сети.

**Основные процедуры обработки данных представлены на рис.** 

       Централизованные формы применения средств вычислительной техники, которые существовали до массового использования персональных электронно-вычислительных машин (ПЭВМ), предполагали их сосредоточение в одном месте и организацию информационно-вычислительных центров (ИВЦ) индивидуального и коллективного пользования (ИВЦКП). Деятельность ИВЦ и ИВЦКП характеризовалась обработкой больших объемов информации, использованием нескольких средних и больших ЭВМ, квалификационным персоналом для обслуживания техники и разработки программного обеспечения. Централизованное  
применение вычислительных и других технических средств позволяло организовать их надежную работу, планомерную загрузку и квалификационное обслуживание.  
     При централизованных формах, когда у пользователей нет непосредственного контакта с ЭВМ, его роль сводится к передаче исходных данных на обработку, получению результатов, выявлению и устранению ошибок.

    Децентрализованные формы использования вычислительных ресурсов начали формироваться со второй половины 80-х годов, когда сфера экономики получила возможность перейти к массовому использованию персональных ЭВМ. Децентрализация предусматривает размещение ПЭВМ в местах возникновения и потребления информации, где создаются автономные пункты ее обработки. К ним относятся абонентские пункты и автоматизированные рабочие места (АРМ). **Централизованная обработка**:

***плюсы***:  
- нет накладных расходов, связанных с согласованием информации в разных местах

- полный контроль над системой в одном месте

- проще разработка

***минусы***:  
- ограничение производительности/пропускной способности - мощность одной машины может расти не бесконечно

- меньшая степень надёжности - одно звено определяет работоспособность всей системы

**Децентрализованная обработка**:

**плюсы**:  
- лучше масштабируемость - то есть можно наращивать пропускную способность, производительность увеличением кол-ва компонентов системы  
- больше надёжность - при отказе части компонентов вся система может продолжать работать, пусть и не в полной мере

**минусы**:  
- необходимость синхронизации возможно противоречивых данных из разных источников, накладные расходы на это

- сложнее разрабатывать такое ПО

1. **Технологии и методологии проектирования информационных систем. Характеристика CASE-средств.**

На данный момент в технологии разработки программного обеспечения существуют два основных подхода к разработке информационных систем, отличающиеся критериями декомпозиции: функционально-модульный (структурный) и объектно-ориентированный.

**Функционально-модульный** подход основан на принципе алгоритмической декомпозиции с выделением функциональных элементов и установлением строгого порядка выполняемых действий.

**Объектно-ориентированный** подход основан на объектной декомпозиции с описанием поведения системы в терминах взаимодействия объектов.

Главным недостатком функционально-модульного подхода является однонаправленность информационных потоков и недостаточная обратная связь. В случае изменения требований к системе это приводит к полному перепроектированию, поэтому ошибки, заложенные на ранних этапах, сильно сказываются на продолжительности и стоимости разработки. Другой важной проблемой является неоднородность информационных ресурсов, используемых в большинстве информационных систем. В силу этих причин в настоящее время наибольшее распространение получил объектно-ориентированный подход.

Под **CASE-технологией** будем понимать комплекс программных средств, поддерживающих процессы создания и сопровождения программного обеспечения, включая анализ и формулировку требований, проектирование, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом (CASE-средство может обеспечивать поддержку только в заданных функциональных областях или в широком диапазоне функциональных областей).

В связи с наличием двух подходов к проектированию программного обеспечения существуют CASE-технологии ориентированные на структурный подход, объектно-ориентированный подход, а также комбинированные. Однако сейчас наблюдается тенденция переориентации инструментальных средств, созданных для структурных методов разработки, на объектно-ориентированные методы, что объясняется следующими причинами:

• возможностью сборки программной системы из готовых компонентов, которые можно использовать повторно;

• возможностью накопления проектных решений в виде библиотек классов на основе механизмов наследования;

• простотой внесения изменений в проекты за счет инкапсуляции данных в объектах;

• быстрой адаптацией приложений к изменяющимся условиям за счет использования свойств наследования и полиформизма;

• возможностью организации параллельной работы аналитиков, проектировщиков и программистов.

Современные CASE-средства поддерживают процессы инжиниринга и автоматизированного реинжиниринга.

Идеальное объектно-ориентированное CASE-средство должно содержать четыре основных блока: анализ, проектирование, разработка и инфраструктура.

**Основные требования к блоку анализа:**

• возможность выбора выводимой на экран информации из всей совокупности данных, описывающих модели;

• согласованность диаграмм при хранении их в репозитарии;

• внесение комментариев в диаграммы и соответствующую документацию для фиксации проектных решений;

• возможность динамического моделирования в терминах событий;

• поддержка нескольких нотаций

**Основные требования к блоку проектирования:**

• поддержка всего процесса проектирования приложения;

• возможность работы с библиотеками, средствами поиска и выбора;

• возможность разработки пользовательского интерфейса;

• поддержка стандартов OLE, ActiveX и доступ к библиотекам HTML или Java;

• поддержка разработки распределенных или двух- и трехзвенных клиент-серверных систем (работа с CORBA, DCOM, Internet).

**Основные требования к блоку реализации:**

• генерация кода полностью из диаграмм;

• возможность доработки приложений в клиент- серверных CASE-средствах типа Power Builder;

• реинжиниринг кодов и внесение соответствующих изменений в модель системы;

• наличие средств контроля, которые позволяют выявлять несоответствие между диаграммами и генерируемыми кодами и обнаруживать ошибки как на стадии проектирования, так и на стадии реализации.

**Основные требования к блоку инфраструктуры:**

• наличие репозитория на основе базы данных, отвечающего за генерацию кода, реинжиниринг, отображение кода на диаграммах, а также обеспечивающего соответствие между моделями и программными кодами;

• обеспечение командной работы (многопользовательской работы и управление версиями) и реинжиниринга.

**Выделим основные критерии оценки и выбора CASE-средств.**

**1. Функциональные характеристики:**

• среда функционирования: проектная среда, программное обеспечение/технические средства, технологическая среда;

• функции, ориентированные на фазы жизненного цикла: моделирование, реализация, тестирование;

• общие функции: документирование, управление конфигурацией, управление проектом;

**2. Надежность; 3. Простота использования; 4. Эффективность; 5. Сопровождаемое; 6. Переносимость; 7. Общие критерии (стоимость, затраты, эффект внедрения, характеристики поставщика)**.

1. **+-Модели процессов накопления информации. Основные принципы поиска информации. Информационно-поисковые системы.**

Хранение и накопление являются одними из основных действий, осуществляемых над информацией и главным средством обеспечения ее доступности в течение некоторого промежутка времени. В настоящее время определяющим направлением реализации этой операции является концепция базы данных, склада (хранилища) данных.

База данных может быть определена как совокупность взаимосвязанных данных, используемых несколькими пользователями и хранящихся с регулируемой избыточностью. Хранимые данные не зависят от программ пользователей, для модификации и внесения изменений применяется общий управляющий метод.

Банк данных — система, представляющая определенные услуги по хранению и поиску данных определенной группе пользователей по определенной тематике.

Система баз данных — совокупность управляющей системы, прикладного программного обеспечения, базы данных, операционной системы и технических средств, обеспечивающих информационное обслуживание пользователей.

Хранилище данных (ХД — используют также термины Data Warehouse, «склад данных», «информационное хранилище») — это база, хранящая данные, агрегированные по многим измерениям. Основные отличия ХД от БД: агрегирование данных; данные из ХД никогда не удаляются; пополнение ХД происходит на периодической основе; формирование новых агрегатов данных, зависящих от старых — автоматическое; доступ к ХД осуществляется на основе многомерного куба или гиперкуба.

ИПС (информационно-поисковая система) - это система, обеспечивающая поиск и отбор необходимых данных в специальной базе с описаниями источников информации (индексе) на основе информационно-поискового языка и соответствующих правил поиска.

Главной задачей любой ИПС является поиск информации релевантной информационным потребностям пользователя. Очень важно в результате проведенного поиска ничего не потерять, то есть найти все документы, относящиеся к запросу, и не найти ничего лишнего. Поэтому вводится качественная характеристика процедуры поиска - релевантность.

Релевантность - это соответствие результатов поиска сформулированному запросу.

Основными показателями ИПС для WWW являются пространственный масштаб и специализация. По пространственному масштабу ИПС можно разделить на локальные, глобальные, региональные и специализированные. Локальные поисковые системы могут быть разработаны для быстрого поиска страниц в масштабе отдельного сервера. Региональные ИПС описывают информационные ресурсы определенного региона, например, русскоязычные страницы в Интернете. Глобальные поисковые системы в отличие от локальных стремятся объять необъятное - по возможности наиболее полно описать ресурсы всего информационного пространства сети Интернет.

В общем случае, можно выделить следующие поисковые инструменты для WWW: каталоги, поисковые системы, метапоисковые системы.

**1. Каталог**

**Каталог** - поисковая система с классифицированным по темам списком аннотаций со ссылками на web-ресурсы. Классификация, как правило, проводится людьми.

Поиск в каталоге очень удобен и проводится посредством последовательного уточнения тем. Тем не менее, каталоги поддерживают возможность быстрого поиска определенной категории или страницы по ключевым словам с помощью локальной поисковой машины. База данных ссылок (индекс) каталога обычно имеет ограниченный объем, заполняется вручную персоналом каталога. Некоторые каталоги используют автоматическое обновление индекса.

Результат поиска в каталоге представляется в виде списка, состоящего из краткого описания (аннотации) документов с гипертекстовой ссылкой на первоисточник.

Адреса популярных каталогов:

Зарубежные каталоги:

Yahoo - www.yahoo.com

Magellan - www.mckinley.com

Российские каталоги:

@Rus - www.aport.ru

Weblist - www.weblist.ru

Улитка - www.ulitka.ru

**2. Поисковая машина**

**Поисковая машина** - поисковая система с формируемой роботом базой данных, содержащей информацию об информационных ресурсах.

Отличительной чертой поисковых машин является тот факт, что база данных, содержащая информацию об Web-страницах, статьях Usenet и т.д., формируется программой-роботом.

Поиск в такой системе проводится по запросу, составляемому пользователем, состоящему из набора ключевых слов или фразы, заключенной в кавычки. Индекс формируется и поддерживается в актуальном состоянии роботами-индексировщиками.

В описании документа чаще всего содержится несколько первых предложений или выдержки из текста документа с выделением ключевых слов. Как правило, указана дата обновления (проверки) документа, его размер в килобайтах, некоторые системы определяют язык документа и его кодировку (для русскоязычных документов).

Адреса наиболее популярных поисковых машин за рубежом и в России.

Зарубежные поисковые машины:

Google - www.google.com

Altavista - www.altavista.com

Excite - www.excite.com

Российские поисковые машины:

Яndex - www.yandex.ru (или www.ya.ru)

Рэмблер - www.rambler.ru

Апорт - www.aport.ru

**3. Метапоисковая машина**

**Метапоисковые системы** (поисковые службы) - системы, способные послать запросы пользователя одновременно нескольким поисковым серверам, затем объединить полученные результаты и представить их пользователю в виде документа со ссылками.

Адреса известных метапоисковых систем:

MetaCrawler - www.metacrawler.com

SavvySearch - www.savvysearch.com

1. **Модели, методы и средства реализации мультимедиа технологий.**

В настоящее время мультимедиа-технологии являются бурно развивающейся областью информационных технологий. В этом направлении активно работает значительное число крупных и мелких фирм, технических университетов и студий (в частности IBM, Apple, Motorola, Philips, Sony, Intel и др.). Области использования

чрезвычайно многообразны: интерактивные обучающие и информационные системы, САПР, развлечения и др.

**Основными характерными особенностями этих технологий являются:**

• объединение многокомпонентной информационной среды (текста, звука, графики, фото, видео) в однородном цифровом представлении;

• обеспечение надежного и долговечного хранения больших объемов информации;

• простота переработки информации.

Многокомпонентную мультимедиа-среду целесообразно разделить на три группы: аудиоряд, видеоряд, текстовая информация.

**Аудиоряд** может включать речь, музыку, эффекты. Главной проблемой при использовании этой группы мультисреды является информационная емкость. Для записи одной минуты WAVE-звука высшего качества необходима память порядка 10 Мбайт.

Другим направлением является использование в мультисреде звуков MIDI (Musical Instrument Digitale Interface).

**Видеоряд** характеризуется большим числом элементов.

Выделяют:

Статический видеоряд включает графику и фото.

Динамический видеоряд представляет собой последовательность статических элементов (кадров). Можно выделить три типовых группы:

• обычное видео (life video) — последовательность фотографий (около 24 кадров в секунду);

• квазивидео — разреженная последовательность фотографий (6—12 кадров в секунду);

• анимация — последовательность рисованных изображений.

Первая проблема при реализации видеорядов — разрешающая способность экрана и число цветов.

Вторая проблема — объем памяти.

При размещении **текстовой** информации на CD-ROM нет никаких сложностей и ограничений ввиду большого информационного объема оптического диска.

**Основные направления использования мультимедиа-технологий:**

• электронные издания для целей образования, развлечения и др.;

• в телекоммуникациях со спектром возможных применений от просмотра заказной телепередачи и выбора нужной книги до участия в мультимедиа-конференциях.

• мультимедийные информационные системы («мультимедиа-киоски»), выдающие по запросу пользователя наглядную информацию.

1. **Модели, методы и средства реализации геоинформационных технологий.**

Графическое представление какой-либо ситуации на экране компьютера подразумевает отображение различных графических образов. Сформированный на экране ЭВМ графический образ состоит из двух различных с точки зрения среды хранения частей — графической «подложки» или графического фона и других графических объектов.

Таким образом, геоинформационные технологии предназначены для широкого внедрения в практику методов и средств работы с пространственно-временными данными, представляемыми в виде системы электронных карт, и предметно-ориентированных сред обработки разнородной информации для различных категорий пользователей.

**Основным классом данных геоинформационных систем (ГИС)** являются координатные данные, содержащие геометрическую информацию и отражающие пространственный аспект. Основные типы координатных данных: точка (узлы, вершины), линия (незамкнутая), контур (замкнутая линия), полигон (ареал, район).

**Рассмотренные типы данных имеют большее число разнообразных связей, которые можно условно разделить на три группы:**

• взаимосвязи для построения сложных объектов из простых элементов;

• взаимосвязи, вычисляемые по координатам объектов;

• взаимосвязи, определяемые с помощью специального описания и семантики при вводе данных.

Основой визуального представления данных при использовании ГИС-технологий является графическая среда, основу которой составляют векторные и растровые (ячеистые) модели.

Векторные модели основаны на представлении геометрической информации с помощью векторов, занимающих часть пространства, что требует при реализации меньшего объема памяти.

В растровых моделях объект (территория) отображается в пространственные ячейки, образующие регулярную сеть. Каждой ячейке растровой модели соответствует одинаковый по размерам, но разный по характеристикам (цвет, плотность) участок поверхности. Ячейка модели характеризуется одним значением, являющимся средней характеристикой участка поверхности. Эта процедура называется пикселизацией.

Таким образом, векторная модель содержит информацию о местоположении объекта, а растровая о том, что расположено в той или иной точке объекта. Векторные модели относятся к бинарным или квазибинарным. Растровые позволяют отображать полутона.

**Важным моментом при проектировании ГИС является размерность модели.**

Применяют двухмерные модели координат (2D) и трехмерные (3D). Двухмерные модели используются при построении карт, а трехмерные — при моделировании геологических процессов, проектировании инженерных сооружений (плотин, водохранилищ, карьеров и др.), моделировании потоков газов и жидкостей.

**Большинство современных ГИС осуществляет комплексную обработку информации:**

• сбор первичных данных;

• накопление и хранение информации;

• различные виды моделирования (семантическое, имитационное, геометрическое, эвристическое);

• автоматизированное проектирование;

• документационное обеспечение.

Основные области использования ГИС:

• электронные карты;

• городское хозяйство;

• государственный земельный кадастр;

• экология;

• дистанционное зондирование;

• экономика;

• специальные системы военного назначения.

1. **+-Технологии защиты информации. Основные понятия о защите информации. Безопасность передачи данных.**
2. **Методы защиты информации. Криптография. Аутентификация. Цифровая подпись.**

Методы защиты:

1. Криптографические метода – шифрование информации и эл.подпись
2. Программные и технические методы – программные: средства упр-я доступом и аудит, имеющиеся в ОС; технич.: замки в дверях, сис-ма видеонаблюдения, камеры, сигнализация
3. Методич.ср-ва – это различные законодательные нормативные док-т, кот определяют правила доступа и работы с инф-ей, а т же санкции за их нарушения.

Криптография

чтобы никто посторонний не мог ознакомиться с передаваемым сооб-ем М, А некот-м обр-м преобразует сооб-е в форму недоступную для прочтения посторонних. С=Ек(М). И преобрз-е так обр сооб-е направляется дальше. В выпо-ет обратное преобраз-е и восстанавливает из него исх.сооб-е М. М=Dк(С).

Исх соо-е наз-ся открытым текстом, нез-мо от типа его содержания. Преобразованное сооб-е С наз-ся шифр-текстом.

Процесс преобраз-ся открытого текста в шифртекст наз-ся зашифрование. Обратный процесс – расшифрование.

пар-р К – ключ для заш-я, расш-я.

Наука, изучающая методы защиты сооб-ий наз-мя криптографией, м-ды вскрытия текста – криптоанализом.

преобраз-е открытого текста в шифртекст заключается в выч-ии некот ф-ции от открытого текста.

проц-сс расш-я закл-ся в применении к шифр-тексту обратной ф-ции-ф-ции расш-я.

Сов-ть ф-ций расш-я и заш-я наз-ся криптографич алгоритмом или шифром.

Надеж-ть любого совр шифра основ-ся на исп-нии ключа. Знание ф-ции заш-я и рас-я явл.недостаточным для раскрытия сообщ-я - Это принцип Кирхгофа.

Если в шифре ключ рас-я совпадает с ключом заш-я или м\бфть из него получен, тогда шифр наз-ся симметричным.

Если ключ рас-я(личный) практически невозможно получить из ключа заш-я(открытый), то шифр наз ассимметричным или шифром с открытым ключом.

шифры:

1. Блочные(инф обраб-ся блоками фиксир длины, длина блока от 64 бит)
2. Потоковые(инф обраб-ся потоком, побитно или побайтно)

Установление подлинности объекта **(аутентификация)** - методика, которая позволяет одной стороне доказывать подлинность другой стороны. Объект может быть человеком, процессом, клиентом или сервером. **Объект, подлинность которого должна быть доказана, называется претендентом;** признаки подлинности ( идентификационный код) претендент а названы верификатором.

**Категории проверки**

В установлении подлинности объекта претендент должен идентифицировать себя для верификатора. Этот может быть сделано одним из трех видов свидетелей: нечто, известное только претенденту , нечто, чем обладает только претендент, или нечто, свойственное только претенденту.

- Нечто известное. Это - секретная информация известная только претенденту, что может быть проверено верификатором. Примеры: пароль, PIN-код, ключ засекречивания и секретный ключ.

**- Нечто, чем обладает**. Это то, что может доказать опознавательный код претендента. Примеры: паспорт, водительские права, удостоверение личности, кредитная карточка и карточка с интегральной схемой, включающей микропроцессор.

- Нечто свойственное. Это свойственные претенденту характеристики. Примеры: обычные подписи, отпечатки пальца, голос, характеристики лица, образец сетчатки глаза и почерк.

Аутентификация бывает **односторонней** (обычно клиент доказывает свою подлинность серверу) и **двусторонней** (**взаимной**). Пример односторонней аутентификации - процедура входа пользователя в систему.

В соответствии с рекомендациями стандарта Х.509 различают процедуры строгой аутентификации следующих типов:

• односторонняя аутентификация;

• двусторонняя аутентификация;

• трехсторонняя аутентификация.

Односторонняя аутентификация предусматривает обмен информацией только в одном направлении.

Двусторонняя аутентификация по сравнению с односторонней содержит дополнительный ответ проверяющей стороны доказывающей стороне, который должен убедить ее, что связь устанавливается именно с той стороной, которой были предназначены аутентификационные данные;

Трехсторонняя аутентификация содержит дополнительную передачу данных от доказывающей стороны проверяющей. Этот подход позволяет отказаться от использования меток времени при проведении аутентификации.

В зависимости от используемых криптографических алгоритмов протоколы строгой аутентификации делятся на протоколы, основанные:

• на симметричных алгоритмах шифрования;

• однонаправленных ключевых хэш-функциях;

• асимметричных алгоритмах шифрования;

• алгоритмах электронной цифровой подписи.

**Электронная подпись** (ЭП) — информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию.

По своему существу электронная подпись представляет собой реквизит электронного документа, позволяющий установить отсутствие искажения информации в электронном документе с момента формирования ЭП и проверить принадлежность подписи владельцу сертификата ключа ЭП. Значение реквизита получается в результате криптографического преобразования информации с использованием закрытого ключа ЭП.

Электронная подпись предназначена для идентификации лица, подписавшего электронный документ и является полноценной заменой (аналогом) собственноручной подписи в случаях, предусмотренных законом.

Использование электронной подписи позволяет осуществить:

- Контроль целостности передаваемого документа: при любом случайном или преднамеренном изменении документа подпись станет недействительной, потому что вычислена она на основании исходного состояния документа и соответствует лишь ему.

- Защиту от изменений (подделки) документа: гарантия выявления подделки при контроле целостности делает подделывание нецелесообразным в большинстве случаев.

- Невозможность отказа от авторства. Так как создать корректную подпись можно, лишь зная закрытый ключ, а он должен быть известен только владельцу, то владелец не может отказаться от своей подписи под документом.

Доказательное подтверждение авторства документа: Так как создать корректную подпись можно, лишь зная закрытый ключ, а он должен быть известен только владельцу, то владелец пары ключей может доказать своё авторство подписи под документом. В зависимости от деталей определения документа могут быть подписаны такие поля, как «автор», «внесённые изменения», «метка времени» и т. д.

1. **Модели, методы и средства реализации технологий искусственного интеллекта.**

**Система называется интеллектуальной, если в ней реализованы следующие основные функции:**

• накапливать знания об окружающем систему мире, классифицировать и оценивать их с точки зрения прагматической полезности и непротиворечивости, инициировать процессы получения новых знаний, осуществлять соотнесение новых знаний с ранее хранимыми;

• пополнять поступившие знания с помощью логического вывода, получать обобщенные знания на основе более частных знаний и логически планировать свою деятельность;

• общаться с человеком на языке, максимально приближенном к естественному человеческому языку, и получать информацию от каналов, аналогичных тем, которые использует человек при восприятии окружающего мира.

**База знаний** представляет собой совокупность сред, хранящих знания различных типов. Рассмотрим кратко их назначение.

База фактов (данных) хранит конкретные данные, а база правил — элементарные выражения, называемые в теории искусственного интеллекта продукциями.

База процедур содержит прикладные программы, с помощью которых выполняются все необходимые преобразования и вычисления.

База закономерностей включает различные сведения, относящиеся к особенностям той среды, в которой действует система.

База метазнаний (база знаний о себе) содержит описание самой системы и способов ее функционирования.

База целей содержит целевые структуры, называемые сценариями, позволяющие организовать процессы движения от исходных фактов, правил, процедур к достижению той цели, которая поступила в систему от пользователя.

**Решатель** осуществляет ряд функций:

- дедуктивного вывода

- индуктивного и правдоподобного выводов.

- блок планирования

- система управления решениями.

**Рецепторы и эффекторы** осуществляют так называемое невербальное общение и используются в интеллектуальных роботах.

**Разновидности интеллектуальных систем:**

• интеллектуальные информационно-поисковые системы;

• экспертные системы (ЭС);

• расчетно-логические системы;

• гибридные экспертные системы.

**Интеллектуальные информационно-поисковые**. Для них характерно использование, помимо базы знаний, реализующей семантическую модель представления знаний о проблемной области, лингвистического процессора.

**Экспертные системы**. Для них характерна аккумуляция в системе знаний и правил рассуждений опытных специалистов в данной предметной области, а также наличие специальной системы объяснений.

**Расчетно-логические системы.** Реализуются благодаря наличию базы знаний в виде функциональной семантической сети и компонентов дедуктивного вывода и планирования.

В последнее время в специальный класс выделяются гибридные экспертные системы. Указанные системы должны вобрать в себя лучшие черты как экспертных, так и расчетно-логических и информационно-поисковых систем.

**В настоящее время выделяют следующие основные типы моделей представления знаний:**

Семантические сетиопределяют, как граф общего вида, в котором можно выделить множество вершин и ребер.

Фреймом называют структуру данных для представления и описания стереотипных объектов, событий или ситуаций.

Существует два типа фреймов:

• образец (прототип) — интенсиональное описание некоторого множества экземпляров;

• экземпляр (пример) — экстенсиональное представление фрейм-образца.

Слоты — это некоторые незаполненные подструктуры фрейма, заполнение которых приводит к тому, что данный фрейм ставится в соответствие некоторой ситуации, явлению или объекту.

В качестве данных фрейм может содержать обращения к процедурам. Выделяют два вида процедур: процедуры-демоны и процедуры-слуги. Процедуры-демоны активизируются при каждой попытке добавления или удаления данных из слота. Процедуры-слуги активизируются только при выполнении условий, определенных пользователем при создании фрейма.

**Продукционные модели** — это набор правил вида «условия — действие», где условиями являются утверждения о содержимом базы данных, а действия представляют собой процедуры, которые могут изменять содержимое базы данных.

**Обычно выделяют следующие типы задач**:

• интерпретация символов или сигналов — составление смыслового описания по входным данным;

• диагностика — определение неисправностей (заболеваний) по симптомам;

• предсказание — определение последствий наблюдаемых ситуаций;

• конструирование — разработка объекта с заданными свойствами при соблюдении установленных ограничений;

• планирование — определение последовательности действий, приводящих к желаемому состоянию объекта;

• слежение — наблюдение за изменяющимся состоянием объекта и сравнение его показателей с установленными или желаемыми;

• управление — воздействие на объект для достижения желаемого поведения.

1. **Информационные технологии организационного управления Корпоративные информационные технологии. ERP, CRM, HRM, CPM, EAI**.

Можно выделить следующие три группы методов управления: ресурсами, процессами, корпоративными знаниями (коммуникациями).

Первоначально была разработана методология планирования материальных ресурсов предприятия **MRP** (Material Requirements Planning), которая использовалась с методологией объемно-календарного планирования MPS (Master Planning Shedule). Следующим шагом было создание методологии планирования производственных ресурсов (мощностей) — **CRP** (Capacitiy Requirements Planning). Эта методология была принципиально похожа на MRP, но ориентирована на расчет производственных мощностей, а не материалов и компонентов.

Объединение указанных выше методологий привело к появлению задачи MRP «второго уровня» — **MRP II** (Manufacturing Resource Planning) — интегрированной методологии планирования, включающей MRP/CRP и использующей MPS и FRP (Finance Resource/requirements Planning) — планирование финансовых ресурсов. Далее была предложена концепция **ERP** (Economic Requirements Planning) — интегрированное планирование всех «бизнес-ресурсов» предприятия.

Следующим шагом было создание концепции управления производственными ресурсами — **CSPP** (Customer Synchronized Resource Planning) — планирование ресурсов, синхронизированное с потреблением. Отличием данной концепции является учет вспомогательных ресурсов, связанных с маркетингом, продажей и послепродажным обслуживанием.

В связи с тем, что в современном производстве задействовано множество поставщиков и покупателей, появилась новая концепция логистических цепочек (Supply Chain). Суть этой концепции состоит в учете при анализе хозяйственной деятельности всей цепочки (сети) превращения товара из сырья в готовое изделие.

**При этом акцент сделан на следующие факторы:**

• стоимость товара формируется на протяжении всей логистической цепочки, но определяющей является стадия продажи конечному потребителю;

• на стоимости товара критическим образом сказывается общая эффективность всех операций;

• наиболее управляемыми являются начальные стадии производства товара, а наиболее чувствительными — конечные (продажные).

**ERP**

В основе ERP-систем лежит принцип создания единого хранилища (репозитория) данных, содержащего всю корпоративную бизнес-информацию: плановую и финансовую информацию, производственные данные, данные по персоналу и др. Наличие единого корпоративного репозитория устраняет необходимость в передаче данных от одной системы к другой (например, от производственной системы к финансовой или к кадровой), а также обеспечивает одновременную доступность информации для любого числа сотрудников предприятия, обладающих соответствующими полномочиями. Целью ERP-систем является не только улучшение управления производственной деятельностью предприятия, но и уменьшение затрат и усилий на поддержку его внутренних информационных потоков.

Существует немало определений ERP-систем. Одно из них, наиболее часто встречающихся, следующее:

ERP-система - это набор интегрированных приложений, позволяющих создать интегрированную информационную среду (ИИС) для автоматизации планирования, учета, контроля и анализа всех основных бизнес-операций предприятия. Основой ИИС предприятия являются именно ERP-системы.

Основные функции ERP-систем:

- ведение конструкторских и технологических спецификаций, которые определяют состав производимых изделий, а также материальные ресурсы и операции, необходимые для его изготовления;

- формирование планов продаж и производства;

- планирование потребностей в материалах и комплектующих, сроков и объемов поставок для выполнения плана производства продукции;

- управление запасами и закупками: ведение договоров, реализация централизованных закупок, обеспечение учета и оптимизации складских и цеховых запасов;

- планирование производственных мощностей: от стратегии всего предприятия до планов использования отдельных станков и оборудования;

- оперативное управление финансами, включая составление финансового плана и осуществление контроля его исполнения, финансовый и управленческий учет;

- управление проектами, включая планирование этапов и ресурсов, необходимых для их реализации.

**СRM-система** (Customer Relationship Management - Управление отношениями с клиентами) - корпоративная информационная система, незаменимый современный инструмент для ведения бизнеса. Дает возможность не просто автоматизировать взаимодействие с клиентами и процесс продаж, а выстроить их работу таким образом, чтобы получать максимальный результат.

**Возможности CRM-систем:**

- Быстрый доступ к актуальной информации о клиентах;

- Оперативность обслуживания клиентов и проведения сделок;

- Формализация схем взаимодействия с клиентами, автоматизация документооборота;

- Быстрое получение всех необходимых отчетных данных и аналитической информации;

- Снижение операционных затрат менеджеров;

- Контроль работы менеджеров;

- Согласованное взаимодействие между сотрудниками и подразделениями.

**HRM-системы** предназначены для управления персоналом, но их функциональность шире, чем у систем автоматизации кадровых операций. Продукты этого класса позволяют работать не только с количественными, но и с качественными показателями персонала. Основная их задача — привлечь и удержать ценных для компании специалистов.

Необходимость во внедрении HRM-систем определяется такими критичными для успешного развития бизнеса потребностями, как:

**- Управление расходами:** Расходы на оплату труда являются одной из крупнейших затратных статей,. По данным Forrester Research (HRMS Q4 2008) на их долю в США приходится в среднем 36,4% от общего объема расходов. HRM-системы используются для планирования и оптимизации расходов .

**- Эффективное управление бизнес-процессами.** HRM-системы поддерживает множество бизнес HR процессов: принятие кадровых решений, поддержание записей о сотрудниках в актуальном состоянии,расчет заработной платы, разработка схем мотивации и пр.Повышение эффективности при выполнении этих задач происходит за счет предоставления прямого доступа сотрудникам ([Employee self-service](http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:Employee_self-service" \o "Employee self-service)) и менеджерам ([Manager self-service](http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:Manager_self-service" \o "Manager self-service)) к нужной им информации.

**- Соблюдение всех правовых норм**, регламентирующих взаимоотношения работника и работодателя. Использование HRM-систем позволяет грамотно разрешать сложные вопросы, реализовывать гибкие схемы расчета заработной платы и кадрового документооборота.

**- Повышение ценности человеческого капитала.** Несмотря на экономический кризис, мало кто будет спорить о ценности человеческого капитала в качестве корпоративного актива. Компании с сильной функцией управления персоналом будут акцентировать внимание на «качественном» росте сотрудников, разрабатывая поощрительные программы и схемы мотивации.

Corporate Performance Management (CPM) — система управления эффективностью предприятия. Комплекс, объединяющий все процессы, методологии и метрики, необходимые для измерения показателей деятельности организации и управления этими показателями. Также можно встретить такие синонимы, как Enterprise Performance Management (EPM), Strategic Enterprise Management (SEM) и Business Performance Management (BPM).

Системы CPM предназначены для целостного мониторинга эффективности бизнес-процессов компании, для контроля и анализа соответствия оперативной деятельности подразделений стратегическим целям и задачам бизнеса. CPM-системы ориентированы на глубокий анализ всех аспектов хозяйственной деятельности предприятия.

В отличие от [ERP](http://www.tadviser.ru/index.php/ERP)-систем предоставляющих возможности оперативного управления бизнесом, в круг задач CPM входят стратегические аспекты деятельности. Именно поэтому системы CPM представляют собой набор мощных возможностей многомерного анализа [BI](http://www.tadviser.ru/index.php/BI)/[OLAP](http://www.tadviser.ru/index.php/OLAP), бизнес-планирования и бюджетирования в масштабах всей компании.

Внедрение CPM-системы в бизнес позволяет:

- создать качественную и адекватную бизнесу модель бюджетирования;

- оценить состояние дел в компании;

- выработать план для достижения поставленных целей.

1. **Системный подход к организации информационных процессов. Модель открытых систем OSI.**
2. **Признаки сложных систем. Сложность, присущая ПС.**

1. Сложные системы часто являются иерархическими и состоят из взаимосвязанных подсистем, которые в свою очередь могут быть разделены на подсистемы, и т.д., вплоть до самого низкого уровня.

2. Выбор, какие компоненты в данной системе считаются элементарными, относительно произволен и в большей степени оставляется на усмотрение исследователя.

3. Внутрикомпонентная связь обычно сильнее, чем связь между компонентами. Это обстоятельство позволяет отделять «высокочастотные» взаимодействия внутри компонентов от «низкочастотной» динамики взаимодействия между компонентами.

4. Иерархические системы обычно состоят из немногих типов подсистем, по-разному скомбинированных и организованных.

5. Любая работающая сложная система является результатом развития работавшей более простой системы… Сложная система, спроектированная с «нуля», никогда не заработает. Следует начинать с работающей простой системы.

Сложные системы в программировании ничем не отличаются от других сложных систем, таких как биологические, социальные и т.п. Сложность системы можно выразить в уровнях управления, чем сложнее система, тем больше уровней управления она имеет. Однако большое количество уровней не означает, что те задачи, которые решает система, не могут быть решены более простой системой, с меньшим количеством уровней управления.

Любая сложная система основана на процессах, которые определяют существо этой системы и которые остаются неизменными во времени. Однако реализация этих процессов может постоянно видоизменяться, принимая различные формы. Сложная система не может иметь детальной формализации в силу своей сложности. Попытка составить детальное формальное описание такой системы приводит к слишком высоким расходам человеческих, временных и финансовых ресурсов. За то время пока создаётся детальное формальное описание, предметная область или наши знания о ней могут существенно измениться, что сделает часть описания непригодным для использования.

*Сделаем выводы о том, что сложная система:*

1. основана на ограниченном числе базовых процессов, определяющих существо системы;

2. не имеет детального формального описания;

3. обладает выраженной иерархичностью;

4. находится в постоянном развитии и, как следствие, не имеет завершённого состояния;

5. разложима на более простые составляющие, вплоть до элементарных;

6. определяется сложностью связей между её составляющими;

7. выполняет только функции управления по отношению к составляющим;

1. **Классический жизненный цикл ПС.**

Очень часто классический жизненный цикл называют каскадной или водопадной моделью, подчеркивая, что разработка рассматривается как последовательность этапов, причем переход на следующий, иерархически нижний этап происходит только после полного завершения работ на текущем этапе (рис. 1.1).

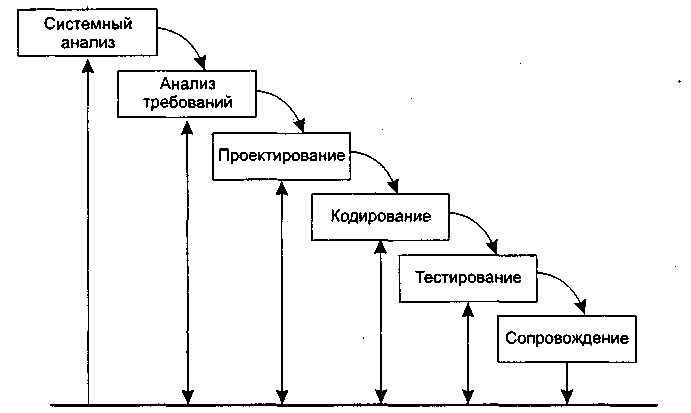
Охарактеризуем содержание основных этапов.

Подразумевается, что разработка начинается на системном уровне и проходит через анализ, проектирование, кодирование, тестирование и сопровождение. При этом моделируются действия стандартного инженерного цикла.

*Системный анализ* задает роль каждого элемента в компьютерной системе, взаимодействие элементов друг с другом. Поскольку ПО является лишь частью большой системы, то анализ начинается с определения требований ко всем системным элементам и назначения подмножества этих требований программному «элементу». Необходимость системного подхода явно проявляется, когда формируется интерфейс ПО с другими элементами (аппаратурой, людьми, базами данных). На этом же этапе начинается решение задачи планирования проекта ПО. В ходе планирования проекта определяются объем проектных работ и их риск, необходимые трудозатраты, формируются рабочие задачи и план-график работ.

*Анализ требований* относится к программному элементу — программному обеспечению. Уточняются и детализируются его функции, характеристики и интерфейс.

Все определения документируются в *спецификации анализа.* Здесь же завершается решение задачи планирования проекта.



**Рис. 1.1.** Классический жизненный цикл разработки ПО

Проектирование состоит в создании представлений:

* архитектуры ПО;
* модульной структуры ПО;
* алгоритмической структуры ПО;
* структуры данных;
* входного и выходного интерфейса (входных и выходных форм данных).

Исходные данные для проектирования содержатся в *спецификации анализа,* то есть в ходе проектирования выполняется трансляция требований к ПО во множество проектных представлений. При решении задач проектирования основное внимание уделяется качеству будущего программного продукта.

*Кодирование* состоит в переводе результатов проектирования в текст на языке программирования.

*Тестирование* — выполнение программы для выявления дефектов в функциях, логике и форме реализации программного продукта.

*Сопровождение —* это внесение изменений в эксплуатируемое ПО. Цели изменений:

* исправление ошибок;
* адаптация к изменениям внешней для ПО среды;
* усовершенствование ПО по требованиям заказчика.

Сопровождение ПО состоит в повторном применении каждого из предшествующих шагов (этапов) жизненного цикла к существующей программе но не в разработке новой программы.

Как и любая инженерная схема, классический жизненный цикл имеет достоинства и недостатки.

*Достоинства классического жизненного цикла:* дает план и временной график по всем этапам проекта, упорядочивает ход конструирования.

*Недостатки классического жизненного цикла:*

1) реальные проекты часто требуют отклонения от стандартной последовательности шагов;

2) цикл основан на точной формулировке исходных требований к ПО (реально в начале проекта требования заказчика определены лишь частично);

3) результаты проекта доступны заказчику только в конце работы.

1. **Итерационный и эволюционный жизненный цикл ПС.**

Общепринятая модель жизненного цикла является идеальной уже потому, что только очень простые задачи проходят все этапы без каких-либо итераций — возвратов на предыдущие шаги производственного процесса. При программировании, например, может обнаружиться, что реализация некоторой функции очень громоздка, неэффективна и вступает в противоречие с требуемой от системы производительностью. В этом случае необходимо перепроектирование, а может быть, и переделка спецификаций. При разработке больших нетрадиционных систем итеративность возникает регулярно на любом этапе жизненного цикла как из-за допущенных на предыдущих шагах ошибок и неточностей, так и из-за изменений внешних требований к условиям эксплуатации системы.

Существуют 3 стратегии конструирования ПО:

* *однократный проход* (водопадная стратегия) — линейная последовательность этапов конструирования;
* *инкрементная стратегия.* В начале процесса определяются все пользовательские и системные требования, оставшаяся часть конструирования выполняется в виде последовательности версий. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система;
* *эволюционная стратегия.* Система также строится в виде последовательности версий, но в начале процесса определены не все требования. Требования уточняются в результате разработки версий.

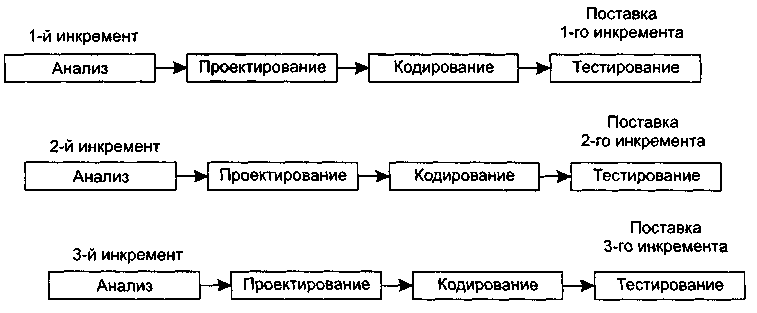
***Инкрементная модель***

Инкрементная модель является классическим примером инкрементной стратегии конструирования (рис. 1.4). Она объединяет элементы последовательной водопадной модели с итерационной философией макетирования.

Первый инкремент приводит к получению базового продукта, реализующего базовые требования (правда, многие вспомогательные требования остаются нереализованными).

План следующего инкремента предусматривает модификацию базового продукта, обеспечивающую дополнительные характеристики и функциональность.

По своей природе инкрементный процесс итеративен, но, в отличие от макетирования, инкрементная модель обеспечивает на каждом инкременте работающий продукт.



1. **Современные методы разработки ПП: ХР-процесс**

Экстр. программирование— облегченный (подвижный) процесс (или методология), (Кент Бек 1999) [11]. ХР-процесс ориентирован на группы малого и среднего размера, строящие программное обеспечение в условиях неопределенных или быстро изменяющихся требований. ХР-группу образуют до 10 сотрудников, которые размещаются в одном помещении. Динамизм обеспечивается с помощью четырех характеристик: непрерывной связи с заказчиком (и в пределах группы), простоты (всегда выбирается минимальное решение), быстрой обратной связи (с помощью модульного и функционального тестирования), смелости в проведении профилактики возможных проблем.

Основная идея ХР — устранить высокую стоимость изменения. ХР-процесс должен быть высокодинамичным процессом. ХР-группа имеет дело с изменениями требований на всем протяжении итерационного цикла разработки, причем цикл состоит из очень коротких итераций. 4 базовые действия в ХР-цикле: кодирование, тестирование, выслушивание заказчика и проектирование.

Базис ХР образуют перечисленные ниже двенадцать методов.

1. Игра планирования (Planning game) — быстрое определение области действия следующей реализации путем объединения деловых приоритетов и технических оценок. Заказчик формирует область действия, приоритетность и сроки с точки зрения бизнеса, а разработчики оценивают и прослеживают продвижение (прогресс).

2. Частая смена версий (Small releases) — быстрый запуск в производство простой системы. Новые версии реализуются в очень коротком (двухнедельном) цикле.

3. Метафора (Metaphor) — вся разработка проводится на основе простой, общедоступной истории о том, как работает вся система.

4. Простое проектирование (Simple design) — проектирование выполняется настолько просто, насколько это возможно в данный момент.

5. Тестирование (Testing) — непрерывное написание тестов для модулей, которые должны выполняться безупречно; заказчики пишут тесты для демонстрации законченности функций. «Тестируй, а затем кодируй» означает, что входным критерием для написания кода является «отказавший» тестовый вариант.

6. Реорганизация (Refactoring) — система реструктурируется, но ее поведение не изменяется; цель — устранить дублирование, улучшить взаимодействие, упростить систему или добавить в нее гибкость.

7. Парное программирование (Pair programming) — весь код пишется двумя программистами, работающими на одном компьютере.

8. Коллективное владение кодом (Collective ownership) — любой разработчик может улучшать любой код системы в любое время.

9. Непрерывная интеграция (Continuous integration) — система интегрируется и строится много раз в день, по мере завершения каждой задачи. Непрерывное регрессионное тестирование, то есть повторение предыдущих тестов, гарантирует, что изменения требований не приведут к регрессу функциональности.

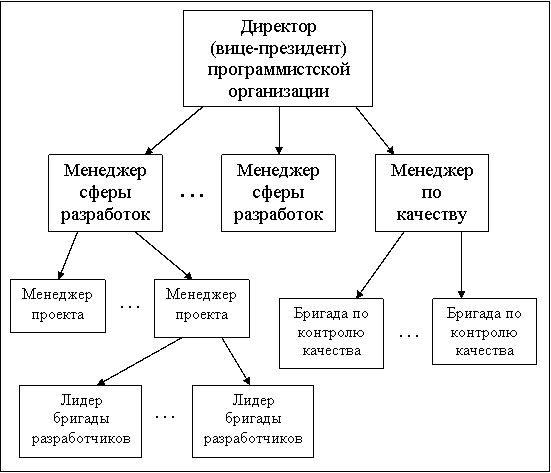
10. 40-часовая неделя (40-hour week) — как правило, работают не более 40 часов в неделю. Нельзя удваивать рабочую неделю за счет сверхурочных работ.

11. Локальный заказчик (On-site customer) — в группе все время должен находиться представитель заказчика, действительно готовый отвечать на вопросы разработчиков.

12. Стандарты кодирования (Coding standards) — должны выдерживаться правила, обеспечивающие одинаковое представление программного кода во всех частях программной системы

1. **Структура управления разработкой программного средства.**

Разработка ПС обычно производится в организации, в которой одновременно могут вестись разработки ряда других программных средств. Для управления всеми этими программными проектами используется иерархическая структура управления.



***директор***программистской организации, (упр. Всеми разработками ПС, решение о начале/прекращении проекта, обсуждение общих организ-х требований к проекту, решение возникающих проблем, требующих использование общих ресурсов программистской организации или изменения заказчиком общих требований)

***менеджер сферы разработок*** (упр-е разр-ками ПС опр. типа, формирование кол-ва исполнителей по проекту, осущ-ет обобщение опыта разр-к в своей сфере, накопление программных средств и документов для повторного использования.)

***менеджер проекта*** (назначается на каждый прогр. проект, осущ-ет планирование и составление расписаний работы бригад)

***бригада*** (большие проекты разбиваются а несколько относительно независимых подпроектов таким образом, чтобы каждый подпроект мог быть выполнен отдельной небольшой бригадой разработчиков)

     обычные бригады (лидер бригады руководит работой младших программистов. Ошибки одного программиста могут препятствовать работе других программистов.)

     неформальные демократические бригады (проект обсуждается всеми членами бригады, задания распределяются в зависимости от способностей и опыта )

     бригады ведущего программиста (Ядро бригады - ведущий программист , его дублер, администратор базы данных разработки).

***менеджер по качеству –***управление обеспечением качества. Каждый этап разр-ки заканчивается смотром по контролю качества - функцией управления разработкой, связан с оценкой того, насколько результаты этой работы согласуются с декларированными требованиями относительно качества ПС.

Существенную роль в управлении качеством ПС играют программные (софтверные) стандарты – они фиксируют удачный опыт высоко квалифицированных специалистов по разработке. Различают два вида таких стандартов:

     стандарты ПС (программного продукта),

     стандарты процесса создания и использования ПС.

*Стандарты ПС* определяют некоторые свойства, которыми должны обладать программы или документы ПС. ( стандарты на языки программирования, на состав документации, на структуру различных документов, на различные форматы и другие).

*Стандарты процесса создания и использования ПС* определяют, как должен проводится этот процесс, т.е. подход к разработке ПС, структуру жизненного цикла ПС и его технологические процессы.

***Бригада по контролю качества*** состоит из ассистентов по качеству ПС. Она проводит смотр ПС с целью поиска возникающих проблем в процессе его разработки, с учетом требований, сформулированных в спецификации качества ПС. Результат – замечания разработчикам (устные/письменные)

1. **Метрики и методы оценки объема работ.**
2. **Внешнее описание программного средства.**

Внешнее описание ПС играет роль точной постановки задачи, решение которой должно обеспечить разрабатываемое ПС. Более того, оно должно содержать всю информацию, которую необходимо знать пользователю для применения ПС. Оно является исходным документом для трех параллельно протекающих процессов: разработки текстов (конструированию и кодированию) программ, входящих в ПС, разработки документации по применению ПС и разработки существенной части комплекта тестов для тестирования ПС. Ошибки и неточности во внешнем описании, в конечном счете, трансформируются в ошибки самой ПС и обходятся особенно дорого, во-первых, потому, что они делаются на самом раннем этапе разработки ПС, и, во-вторых, потому, что они распространяются на три параллельных процесса. Это требует принятия особенно серьезных мер по их предупреждению.

Исходным документом для разработки внешнего описания ПС являются *определение требований* к ПС. Но так как через этот документ передается от заказчика (пользователя) к разработчику основная информация относительно требуемого ПС, то формирование этого документа представляет собой довольно длительный и трудный итерационный процесс взаимодействия между заказчиком и разработчиком, с которого и начинается этап разработки требований к ПС.

В определении внешнего описания легко бросаются в глаза две самостоятельные его части. Описание поведения ПС определяет функции, которые должна выполнять ПС, и потому его называют *функциональной спецификацией* ПС. Функциональная спецификация определяет допустимые фрагменты программ, реализующих декларированные функции. Требования к качеству ПС должны быть сформулированы так, чтобы разработчику были ясны цели, которые он должен стремиться достигнуть при разработке этого ПС. Эту часть внешнего описания будем называть *спецификацией качества* ПС (в литературе ее часто называют *нефункциональной спецификацией*, но она, как правило, включает и требования к технологическим процессам). Она, в отличие от функциональной спецификации, представляется в неформализованном виде и играет роль тех ориентиров, которые в значительной степени определяют выбор подходящих альтернатив при реализации функций ПС, а также определяет стиль всех документов и программ требуемого ПС. Тем самым, спецификация качества играет решающую роль в обеспечении требуемого качества ПС.

Обычно разработка спецификации качества предшествует разработке функциональной спецификации ПС, так как некоторые требования к качеству ПС могут предопределять включение в функциональную спецификацию специальных функций, например, функции защиты от несанкционированного доступа к некоторым объектам информационной среды. Таким образом, структуру внешнего описания ПС можно выразить формулой:

Внешнее описание ПС = определение требований + спецификация качества ПС  + функциональная спецификация ПС

Таким образом, внешнее описание определяет, что должно делать ПС и какими внешними свойствами оно должно обладать. Оно должно достаточно точно и полно определять задачи, которые должны решить разработчики требуемого ПС. В то же время оно должно быть понято представителем пользователем - на его основании заказчиком достаточно часто принимается окончательное решение на заключение договора на разработку ПС. Внешнее описание играет большую роль в обеспечении требуемого качества ПС, так как спецификация качества ставит для разработчиков ПС конкретные ориентиры, управляющие выбором приемлемых решений при реализации специфицированных функций.

1. **Основные классы архитектур программных средств.**

**Архитектура ПС** - это представление ПС как системы, состоящей из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают обычно отдельные программы.

Разработка архитектуры является первым этапом борьбы со сложностью ПС, на котором реализуется принцип выделения относительно независимых компонент.

**Основные задачи разработки архитектуры ПС:**

1)выделение программных подсистем и отображение на них внешних функций (заданных во внешнем описании) ПС;

2)определение способов взаимодействия между выделенными программными подсистемами.

**Основные классы архитектур программных средств:**

1)цельная программа;

2)комплекс автономно выполняемых программ;

3)слоистая программная система;

4)коллектив параллельно выполняемых программ.

**Цельная программа** представляет вырожденный случай архитектуры ПС: в состав ПС входит только одна программа. Такую архитектуру выбирают обычно в том случае, когда ПС должно выполнять одну какую-либо ярко выраженную функцию и ее реализация не представляется слишком сложной. Естественно, что такая архитектура не требует какого-либо описания (кроме фиксации класса архитектуры), так как отображение внешних функций на эту программу тривиально, а определять способ взаимодействия не требуется (в силу отсутствия какого-либо внешнего взаимодействия программы, кроме как взаимодействия ее с пользователем, а последнее описывается в документации по применению ПС).

**Комплекс автономно выполняемых программ** состоит из набора программ, такого, что:

1)любая из этих программ может быть активизирована (запущена) пользователем;

2)при выполнении активизированной программы другие программы этого набора не могут быть активизированы до тех пор, пока не закончит выполнение активизированная программа;

3)все программы этого набора применятся к одной и той же информационной среде.

Таким образом, программы этого набора по управлению никак не взаимодействуют - взаимодействие между ними осуществляется только через общую информационную среду.

**Слоистая программная система** состоит из некоторой упорядоченной совокупности программных подсистем, называемых слоями, такой, что:

1)на каждом слое ничего не известно о свойствах (и даже существовании) последующих (более высоких) слоев;

2)каждый слой может взаимодействовать по управлению (обращаться к компонентам) с непосредственно предшествующим (более низким) слоем через заранее определенный интерфейс, ничего не зная о внутреннем строении всех предшествующих слоев;

3)каждый слой располагает определенными ресурсами, которые он либо скрывает от других слоев, либо предоставляет непосредственно последующему слою (через указанный интерфейс) некоторые их абстракции.

**Коллектив параллельно действующих программ** представляет собой набор программ, способных взаимодействовать между собой, находясь одновременно в стадии выполнения. Это означает, что такие программы, во-первых, вызваны в оперативную память, активизированы и могут попеременно разделять по времени один или несколько центральных процессоров, а во-вторых, осуществлять между собой динамические (в процессе выполнения) взаимодействия, на базе которых производиться их синхронизация. Обычно взаимодействие между такими процессами производится путем передачи друг другу некоторых сообщений.

1. **Понятие программного модуля и его основные характеристики.**

Приступая к разработке каждой программы ПС, следует иметь ввиду, что она, как правило, является большой системой, поэтому мы должны принять меры для ее упрощения. Для этого такую программу разрабатывают по частям, которые называются программными модулями. Программный модуль - это любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в описаниях процесса. Это означает, что каждый программный модуль программируется, компилируется и отлаживается отдельно от других модулей программы, и тем самым, физически разделен с другими модулями программы. Более того, каждый разработанный программный модуль может включаться в состав разных программ, если выполнены условия его использования, декларированные в документации по этому модулю.

**Основные характеристики программного модуля:**

хороший модуль снаружи проще, чем внутри;

хороший модуль проще использовать, чем построить.

Размер модуля измеряется числом содержащихся в нем операторов (строк). Модуль не должен быть слишком маленьким или слишком большим.

**Прочность модуля** - это мера его внутренних связей. Чем выше прочность модуля, тем больше связей он может спрятать от внешнейпо отношению к нему части программы и, следовательно, тем больший вклад в упрощение программы он может внести

**Функционально прочный модуль** - это модуль, выполняющий (реализующий) одну какую-либо определенную функцию. При реализации этой функции такой модуль может использовать и другие модули.

**Информационно прочный модуль** - это модуль, выполняющий (реализующий) несколько операций (функций) над одной и той же структурой данных (информационным объектом), которая считается неизвестной вне этого модуля. Для каждой из этих операций в таком модуле имеется свой вход со своей формой обращения к нему.

**Сцепление модуля** - это мера его зависимости по данным от других модулей. Характеризуется способом передачи данных. Чем слабее сцепление модуля с другими модулями, тем сильнее его независимость от других модулей.

**Рутинность модуля** - это его независимость от предыстории обращений к нему.

1. **Методы разработки структуры программ.**

В процессе разработки программы ее модульная структура может по-разному формироваться и использоваться для определения порядка программирования и отладки модулей, указанных в этой структуре. Поэтому можно говорить о разных методах разработки структуры программы. Обычно в литературе обсуждаются два метода: **метод восходящей разработки и метод нисходящей разработки.**

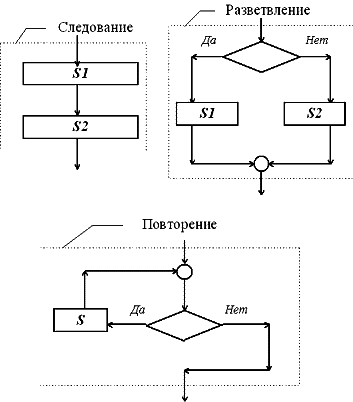
**Метод восходящей разработки** заключается в следующем. Сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модулей самого нижнего уровня (листья дерева модульной структуры программы), в таком порядке, чтобы для каждого программируемого модуля были уже запрограммированы все модули, к которым он может обращаться. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в принципе в таком же (восходящем) порядке, в каком велось их программирование.

**Метод *нисходящей разработки*** заключается в следующем. Как и в предыдущем методе сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модуля самого верхнего уровня (головного), переходя к программированию какого-либо другого модуля только в том случае, если уже запрограммирован модуль, который к нему обращается. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в таком же (нисходящем) порядке. При этом первым тестируется головной модуль программы, который представляет всю тестируемую программу и поэтому тестируется при естественном состоянии информационной среды, при котором начинает выполняться эта программа. При этом те модули, к которым может обращаться головной, заменяются их имитаторами (так называемыми заглушками). Каждый *имитатор* модуля представляется весьма простым программным фрагментом, который, в основном, сигнализирует о самом факте обращения к имитируемому модулю, производит необходимую для правильной работы программы обработку значений его входных параметров (иногда с их распечаткой) и выдает, если это необходимо, заранее запасенный подходящий результат.

1. **Структурное программирование и пошаговая детализация. Псевдокод.**

***Структурное программирование.***

Построение программы как композиции из нескольких типов управляющих конструкций (структур), которые позволяют сильно повысить понимаемость логики работы программы. Программирование с использованием только таких конструкций назвали *структурным*.



Основными конструкциями структурного программирования являются: следование, разветвление и повторение (см. Рис.5). Компонентами этих конструкций являются обобщенные операторы (узлы обработки) S, S1, S2 и условие (предикат) P. В качестве обобщенного оператора может быть либо простой оператор используемого языка программирования (операторы присваивания, ввода, вывода, обращения к процедуре), либо фрагмент программы, являющийся композицией основных управляющих конструкций структурного программирования. Существенно, что каждая из этих конструкций имеет по управлению только один вход и один выход. Тем самым, и обобщенный оператор имеет только один вход и один выход.

Весьма важно также, что эти конструкции являются уже математическими объектами (что, по существу, и объясняет причину успеха структурного программирования). Доказано, что для каждой неструктурированной программы можно построить функционально эквивалентную (т.е. решающую ту же задачу) структурированную программу. Для структурированных программ можно математически доказывать некоторые свойства, что позволяет обнаруживать в программе некоторые ошибки.

***Пошаговая детализация и понятие о псевдокоде.***

В качестве основного метода построения текста модуля современная технология программирования рекомендует *пошаговую детализацию.* Сущность этого метода заключается в разбиении процесса разработки текста модуля на ряд шагов.

На первом шаге описывается общая схема работы модуля в обозримой линейной текстовой форме (т.е. с использованием очень крупных понятий), причем это описание не является полностью формализованным и ориентировано на восприятие его человеком. На каждом следующем шаге производится уточнение и детализация одного из понятий (будем называть его *уточняемым*), в каком либо описании, разработанном на одном из предыдущих шагов. В результате такого шага создается описание выбранного уточняемого понятия либо в терминах базового языка программирования (т.е. выбранного для представления модуля), либо в такой же форме, что и на первом шаге с использованием новых уточняемых понятий. Этот процесс завершается, когда все уточняемые понятия будут *уточнения* (т.е. в конечном счете будут выражены на базовом языке программирования). Последним шагом является получение текста модуля на базовом языке программирования путем замены всех вхождений уточняемых понятий заданными их описаниями и выражение всех вхождений конструкций структурного программирования средствами этого языка программирования.

Пошаговая детализация связана с использованием частично формализованного языка для представления указанных описаний, который получил название *псевдокода*. Этот язык позволяет использовать все конструкции структурного программирования, которые оформляются формализовано, вместе с неформальными фрагментами на естественном языке для представления обобщенных операторов и условий.

1. **Тестирование и отладка программных средств.**

*Отладка* ПС − это деятельность, направленная на обнаружение и исправление ошибок в ПС с использованием процессов выполнения его программ. *Тестирование* ПС − это процесс выполнения его программ на некотором наборе данных, для которого заранее известен результат применения или известны правила поведения этих программ. Указанный набор данных называется *тестовым* или просто *тестом*. Таким образом, отладку можно представить в виде многократного повторения трех процессов: тестирования, в результате которого может быть констатировано наличие в ПС ошибки, поиска места ошибки в программах и документации ПС и редактирования программ и документации с целью устранения обнаруженной ошибки. Другими словами:

Отладка = Тестирование + Поиск ошибок + Редактирование.

*1)Автономная отладка программного средства:*

При автономной отладке ПС каждый модуль на самом деле тестируется в некотором программном окружении, кроме случая, когда отлаживаемая программа состоит только из одного модуля. Это окружение состоит из других модулей, часть которых является модулями отлаживаемой программы, которые уже отлажены, а часть − модулями, управляющими отладкой (отладочными модулями, см. ниже). Таким образом, при автономной отладке тестируется всегда некоторая программа (тестируемая программа), построенная специально для тестирования отлаживаемого модуля.

*2)Комплексная отладка программного средства:*

**Тестирование архитектуры ПС***.* Целью тестирования является поиск несоответствия между описанием архитектуры и совокупностью программ ПС.

**Тестирование внешних функций**. Целью тестирования является поиск расхождений между функциональной спецификацией и совокупностью программ ПС

**Тестирование качества ПС***.* Целью тестирования является поиск нарушений требований качества, сформулированных в спецификации качества ПС.

**Тестирование документации по применению ПС***.* Целью тестирования является поиск несогласованности документации по применению и совокупностью программ ПС, а также выявление неудобств, возникающих при применении ПС.

**Тестирование определения требований к ПС.** Целью тестирования является выяснение, в какой мере ПС не соответствует предъявленному определению требований к нему.

1. **Объектно-ориентированный подход к разработке ПС.**

Объектно-ориентированное представление ПС основывается на принципах абстрагирования, инкапсуляции, модульности и иерархической организации.

*Абстрагирование* – это отвлечение от несущественных характеристик конкретных объектов, выделение только существенных характеристик. (часы – время без учета формы , цвета.. )

*Инкапсуляция* является процессом разделения элементов абстракции на секции с различной видимостью. Инкапсуляция служит для отделения интерфейса абстракции от ее реализации.

Инкапсуляция и абстракция — взаимодополняющие понятия: абстракция выделяет внешнее поведение объекта, а инкапсуляция содержит и скрывает реализацию, которая обеспечивает это поведение.

Модульность определяет способность системы подвергаться декомпозиции на ряд сильно связанных и слабо сцепленных модулей. Общая цель декомпозиции на модули: уменьшение сроков разработки и стоимости ПС за счет выделения модулей, которые проектируются и изменяются независимо.

Иерархическая организация — формирование из абстракций иерархической структуры. Определением иерархии в проекте упрощаются понимание проблем заказчика и их реализация — сложная система становится обозримой человеком.

Двумя важными инструментами иерархической организации в объектно-ориентированных системах являются:

* структура из классов *(«is a»*-иерархия);
* структура из объектов *(«part of»*-иерархия).

**объектно-ориентированный подход помогает справиться с такими сложными проблемами, как**

1)уменьшение сложности программного обеспечения;

2)повышение надежности программного обеспечения;

3)обеспечение возможности модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов;

4)обеспечение возможности повторного использования отдельных компонентов программного обеспечения.

Систематическое применение объектно-ориентированного подхода позволяет разрабатывать хорошо структурированные, надежные в эксплуатации, достаточно просто модифицируемые программные системы. Этим объясняется интерес программистов к объектно-ориентированному подходу и объектно-ориентированным языкам программирования. Объектно-ориентированный подход является одним из наиболее интенсивно развивающихся направлений теоретического и прикладного программирования.

1. **Унифицированный язык моделирования UML.**

UML — стандартный язык для написания моделей анализа, проектирования и реализации объектно-ориентированных программных систем. UML может использоваться для визуализации, спецификации, конструирования и документирования результатов программных проектов. UML — это не визуальный язык программирования, но его модели прямо транслируются в текст на языках программирования (Java, C++, Visual Basic, Ada 95, Object Pascal) и даже в таблицы для реляционной БД.

Словарь UML образуют три разновидности строительных блоков: предметы, отношения, диаграммы.

Предметы — это абстракции, которые являются основными элементами в модели, отношения связывают эти предметы, диаграммы группируют коллекции предметов.

**Язык UML предназначен для решения следующих задач:**

1)Предоставить в распоряжение пользователей легко воспринимаемый и выразительный язык визуального моделирования, специально предназначенный для разработки и документирования моделей сложных систем самого различного целевого назначения.

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

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

4)Описание языка UML должно включать в себя семантический базис для понимания общих особенностей объектно-ориентированного анализа и проектирования.

5)Поощрять развитие рынка объектных инструментальных средств.

6)Способствовать распространению объектных технологий и соответствующих понятий объектно-ориентированного анализа и проектирования.

7)Интегрировать в себя новейшие и наилучшие достижения практики объектно-ориентированного анализа и проектирования.

**Общая структура языка UML:**

С самой общей точки зрения описание языка UML состоит из двух взаимодействующих частей, таких как:

1)Семантика языка UML. Представляет собой некоторую метамодель, которая определяет абстрактный синтаксис и семантику понятий объектного моделирования на языке UML.

2)Нотация языка UML. Представляет собой графическую нотацию для визуального представления семантики языка UML.

В UML имеются четыре разновидности предметов:

***Структурные предметы***являются существительными в UML-моделях. Представляют статические части модели — понятийные или физические элементы. Перечислим восемь разновидностей структурных предметов.

1. *Класс* — описание множества объектов, которые разделяют одинаковые свойства, операции, отношения и семантику (смысл). Класс реализует один или несколько интерфейсов.
2. *Интерфейс —* набор операций, которые определяют услуги класса или компонента. Интерфейс описывает поведение элемента, видимое извне.
3. *Кооперация* (сотрудничество) определяет взаимодействие и является совокупностью ролей и других элементов, которые работают вместе для обеспечения коллективного поведения более сложного, чем простая сумма всех элементов.
4. *Актер —* набор согласованных ролей, которые могут играть пользователи при взаимодействии с системой. Каждая роль требует от системы определенного поведения.
5. *Элемент Use Case* (Прецедент) — описание последовательности действий (или нескольких последовательностей), выполняемых системой в интересах отдельного актера и производящих видимый для актера результат.
6. *Активный класс —* класс, чьи объекты имеют один или несколько процессов (или потоков) и поэтому могут инициировать управляющую деятельность
7. *Компонент —* физическая и заменяемая часть системы, которая соответствует набору интерфейсов и обеспечивает реализацию этого набора интерфейсов
8. *Узел —* физический элемент, который существует в период работы системы и представляет ресурс, обычно имеющий память и возможности обработки

***Предметы поведения*** *—* динамические части UML-моделей. Они являются глаголами моделей, представлением поведения во времени и пространстве. Существует две основные разновидности предметов поведения.

1. *Взаимодействие* — поведение, заключающее в себе набор сообщений, которыми обменивается набор объектов в конкретном контексте для достижения определенной цели.
2. *Конечный автомат —* поведение, которое определяет последовательность состояний объекта или взаимодействия, выполняемые в ходе его существования в ответ на события .С помощью конечного автомата может определяться поведение индивидуального класса или кооперации классов. Элементами конечного автомата являются состояния, переходы (от состояния к состоянию), события (предметы, вызывающие переходы) и действия (реакции на переход).

***Группирующие предметы***— организационные части UML-моделей. Это ящики, по которым может быть разложена модель. Предусмотрена одна разновидность группирующего предмета — *пакет —* общий механизм для распределения элементов по группам. В пакет могут помещаться структурные предметы, предметы поведения и даже другие группировки предметов. В отличие от компонента (который существует в период выполнения), пакет существует только в период разработки.

***Поясняющие предметы*** *—* разъясняющие части UML-моделей. Они являются замечаниями, которые можно применить для описания, объяснения и комментирования любого элемента модели. Предусмотрена одна разновидность поясняющего предмета — *примечание —* символ для отображения ограничений и замечаний, присоединяемых к элементу или совокупности элементов

В UML имеются четыре разновидности отношений:

*Зависимость* — семантическое отношение между двумя предметами, в котором изменение в одном предмете (независимом предмете) может влиять на семантику другого предмета (зависимого предмета).

1. *Ассоциация —* структурное отношение, которое описывает набор связей, являющихся соединением между объектами.
2. *Обобщение* — отношение специализации/обобщения, в котором объекты специализированного элемента (потомка, ребенка) могут заменять объекты обобщенного элемента (предка, родителя). Иначе говоря, потомок разделяет структуру и поведение родителя.
3. *Реализация —* семантическое отношение между классификаторами, где один классификатор определяет контракт, который другой классификатор обязуется выполнять (к классификаторам относят классы, интерфейсы, компоненты, элементы Use Case, кооперации).

*Диаграмма* — графическое представление множества элементов, наиболее часто изображается как связный граф из вершин (предметов) и дуг (отношений). UML включает девять видов диаграмм:

1) диаграммы классов;

2) диаграммы объектов;