**Основные понятия и определения ИС. Жизненный цикл информационных систем**

**Понятие информационной системы**

Под информационной системой обычно понимается прикладная программная подсистема, ориентированная на сбор, хранение, поиск и обработку текстовой и/или фактографической информации.

Подавляющее большинство информационных систем работает в режиме диалога с пользователем.

**Свойства информационных систем:**

• любая ИС может быть подвергнута анализу, построена и управляема на основе общих принципов построения сложных систем;

• при построении ИС необходимо использовать системный подход;

• ИС является динамичной и развивающейся системой;

• ИС следует воспринимать как систему обработки информации, состоящую из компьютерных и телекоммуникационных устройств, реализованную на базе современных технологий;

• выходной продукцией ИС является информация, на основе которой принимаются решения или производится автоматическое выполнение рутинных операций;

• участие человека зависит от сложности системы, типов и наборов данных, степени формализации решаемых задач.

**Процессы в информационной системе:**

* ввод информации из внешних и внутренних источников;
* обработка входящей информации;
* хранение информации для последующего ее использования; вывод информации в удобном для пользователя виде;
* обратная связь, т.е. представление информации, переработанной в данной организации, для корректировки входящей информации.

**С учетом сферы применения выделяют:**

* технические ИС,
* экономические ИС,
* ИС в гуманитарных областях и т.д.

**Классификация ИС**

1. **По областям применения.**

Информационных системы в экономике (АСЭ – автоматизированные системы в экономике). В образовании (АСО). В научных исследованиях (АСНИ) и т.д.

2. **По характеру информации, которой оперирует ИС**. Фактографические или документальные

3. **По роли, которую ИС играют в профессиональной деятельности.**

• Системы управления. АСУ (автоматизированная система управления), САУ(система автоматического управления - без участия человека).

• Вычислительные информационные системы.

• Поисково-справочные информационные системы.

• Системы принятия решения.

• Информационные обучающие системы.

4. **По техническим средствам:**

Один компьютер / Локальная сеть / Глобальная сеть

**Соотношение между ИС и ИТ**

**Информационная технология** - процесс различных операций и действий над данными.

Все процессы преобразования информации в информационной системе осуществляются с помощью информационных технологий.

**Информационная система** - среда, составляющими элементами которой являются компьютеры, компьютерные сети, программные продукты, базы данных, люди, различного рода технологические и программные средства и т.д.

Таким образом, информационная технология является более емким понятием, чем информационная система.

Реализация функций информационной системы невозможна без знаний ориентированной на нее информационной технологии. Информационная технология может существовать и вне сферы информационной системы.

**Понятие жизненного цикла (ЖЦ) ИС**

***ЖЦИС***- это период создания и использования ИС, начиная с момента возникновения потребности в ИС и заканчивая моментом полного её выхода из эксплуатации.

Традиционные основные этапы ЖЦ *ПО*:

•анализ требований;

•проектирование;

•кодирование (программирование);

•тестирование и отладка;

•эксплуатация и сопровождение.

ЖЦ является моделью создания и использования *ПО*, отражающей его различные состояния, начиная с момента возникновения необходимости в данном программном изделии и заканчивая моментом его полного выхода из употребления у всех пользователей.

**Стадии жизненного цикла информационной системы**

**1. Предпроектное обследование**

1.1. Сбор материалов для проектирования; формулирование требований, изучение объекта автоматизации, даются предварительные выводы предпроектного варианта ИС.

1.2. Анализ материалов и разработка документации; разрабатывается технико- экономическое обоснование с техническим заданием на стадии проектирования ИС.

**2. Проектирование**

**2.1. Предварительное проектирование:**

* выбор проектных решений по аспектам разработки ИС; описание реальных компонент ИС;
* оформление и утверждение технического проекта (ТП).

**2.2. Детальное проектирование:**

* выбор или разработка математических методов или алгоритмов программ;
* корректировка структур БД;
* создание документации на доставку и установку программных продуктов;
* выбор комплекса технических средств с документацией на её установку.

**2.3. Разработка техно-рабочего проекта ИС (ТРП).**

**2.4. Разработка методологии реализации функций управления с помощью ИС и описанием регламента действий аппарата управления.**

**3. Разработка ИС**

• получение и установка технических и программных средств;

• тестирование и доводка

программного комплекса;

• разработка инструкций по эксплуатации программно- технических средств.

**4. Ввод ИС в эксплуатацию**

• ввод технических средств;

• ввод программных средств;

• обучение и сертификация персонала;

• опытная эксплуатация;

• сдача и подписание актов приёмки-сдачи работ.

**5. Эксплуатация ИС**

• повседневная эксплуатация;

• общее сопровождение всего проекта

**Структура ЖЦ ПО по стандарту ISO/IEC 12207 базируется на трёх группах процессов:**

•**основные процессы** ЖЦ ПО (приобретение, поставка, разработка, эксплуатация, сопровождение);

•**вспомогательные процессы**, обеспечивающие 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).

**Разработка** включает в себя все работы по созданию ПО и его компонент в соответствии с заданными требованиями.

• оформление проектной и эксплуатационной документации,

• подготовка материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала и т.д.

• анализ, проектирование и реализацию (программирование).

**Эксплуатация** включает в себя работы по внедрению компонентов ПО в эксплуатацию.

• конфигурирование базы данных и рабочих мест пользователей,

• обеспечение эксплуатационной документацией,

проведение обучения персонала и т.д.,

• эксплуатация, в том числе локализация проблем и устранение причин их возникновения,

• модификацию ПО в рамках установленного регламента,

• подготовку предложений по совершенствованию, развитию и модернизации системы.

**Управление проектом** связано с вопросами планирования и организации работ, создания коллективов разработчиков и контроля за сроками и качеством выполняемых работ.

**Техническое и организационное обеспечение проекта включает:**

• выбор методов и инструментальных средств для

реализации проекта,

• определение методов описания промежуточных состояний разработки,

• разработку методов и средств испытаний ПО, обучение

персонала и т.п.

**Обеспечение качества проекта** связано с проблемами верификации, проверки и тестирования ПО.

**Верификация** - это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа.

Проверка позволяет оценить **соответствие параметров разработки с исходными требованиями.**

Проверка частично совпадает с тестированием, которое связано с идентификацией различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям.

В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля конфигурации отдельных компонентов и всей системы в целом.

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы жизненного цикла ПО, прежде всего, процессы разработки и сопровождения ПО.

При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учёта их связей и функций, создания унифицированной структуры и обеспечения развития всей системы.

Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ.

Общие принципы и рекомендации конфигурационного учёта, планирования и управления конфигурациями ПО отражены в проекте стандарта **ISO 12207-2.**

Каждый процесс характеризуется определёнными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами.

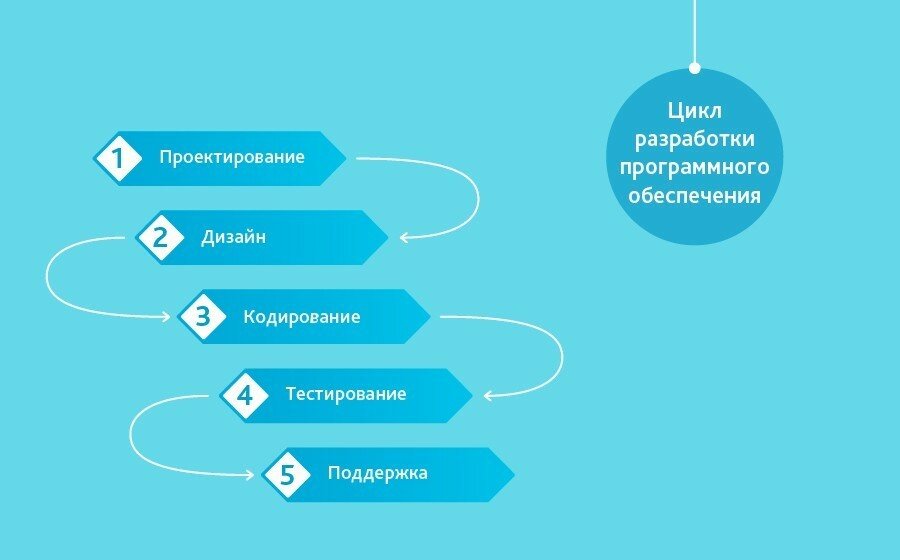
Результатами анализа, в частности, являются функциональные модели, информационные модели и соответствующие им диаграммы.

ЖЦ ПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

**Модели ЖЦ ИС**

Существующие модели ЖЦ определяют порядок исполнения этапов в ходе разработки, а также критерии перехода от этапа к этапу.

В соответствии с этим наибольшее распространение получили три следующие модели ЖЦ:



«Waterfall Model» (каскадная модель или «водопад») (70-80г.г.) — предполагает переход на следующий этап после полного окончания работ по предыдущему этапу

В модели Waterfall . Благодаря её жесткости, разработка проходит быстро, стоимость и срок заранее определены.

Каскадная модель будет давать отличный результат только в проектах с четко и заранее определенными требованиями и способами их реализации.

Нет возможности сделать шаг назад, тестирование начинается только после того, как разработка завершена или почти завершена.

Продукты, разработанные по данной модели без обоснованного ее выбора, могут иметь недочеты (список требований нельзя скорректировать в любой момент), о которых становится известно лишь в конце из-за строгой последовательности действий.

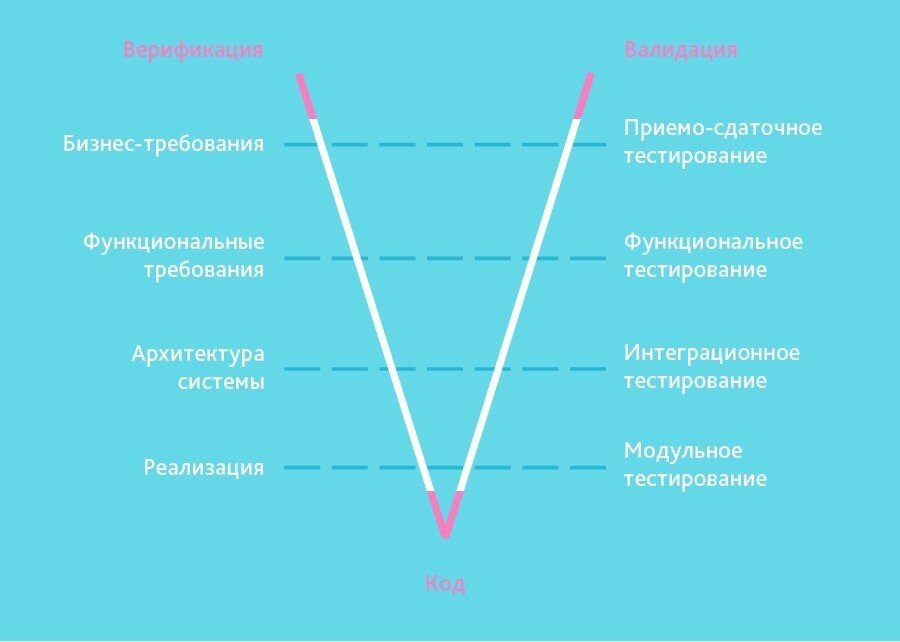
Стоимость внесения изменений высока, так как для ее инициализации приходится ждать завершения всего проекта. Тем не менее, фиксированная стоимость часто перевешивает минусы подхода. Исправление осознанных в процессе создания недостатков возможно, и, требует от одного до трех дополнительных соглашений к контракту с небольшим ТЗ.

**Когда использовать каскадную методологию?**

• Только тогда, когда требования известны, понятны и зафиксированы. Противоречивых требований не имеется.

• Нет проблем с доступностью программистов нужной квалификации.

• В относительно небольших проектах.



V-Model унаследовала структуру «шаг за шагом» от каскадной модели

V- образная модель применима к системам, которым особенно важно бесперебойное функционирование.

Например, прикладные программы в клиниках для наблюдения за пациентами, интегрированное ПО для механизмов управления аварийными подушками безопасности в транспортных средствах и так далее.

Особенностью модели можно считать то, что она направлена на тщательную проверку и тестирование продукта, находящегося уже на первоначальных стадиях проектирования.

Стадия тестирования проводится одновременно с соответствующей стадией разработки, например, во время кодирования пишутся модульные тесты.

Пример работы на основе V-методологии — мобильное приложение для европейского сотового оператора, который экономит расходы на роуминг во время путешествий.

Проект выполняется по четкому ТЗ, но в него включен значительный этап тестирования: удобства интерфейса, функционального, нагрузочного и в том числе интеграционного, которое должно подтверждать, что несколько компонентов от различных производителей вместе работают стабильно, невозможна кража денег и кредитов.

**Когда использовать V-модель?**

• Когда требуется тщательное тестирование

• Для малых и средних проектов, где требования четко определены и фиксированы.

• В условиях доступности инженеров необходимой квалификации, особенно тестировщиков.

**«Incremental Model» (инкрементная модель)**

В инкрементной модели полные требования к системе делятся на различные сборки.

Терминология часто используется для описания поэтапной сборки ПО.

Имеют место несколько циклов разработки, и вместе они составляют жизненный цикл «мульти-водопад».

Цикл разделен на более мелкие легко создаваемые модули.

Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования.

Процедура разработки по инкрементной модели предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых «инкрементов».

Процесс продолжается до тех пор, пока не будет создана полная система.

Инкрементные модели используются там, где отдельные запросы на изменение ясны, могут быть легко формализованы и реализованы.

Пример проектов: читалка DefView, сеть электронных библиотек Vivaldi.

**Пример одного инкремента.**

Сеть электронных библиотек Vivaldi пришла на смену DefView.

DefView подключалась к одному серверу документов, а теперь может подключаться ко многим.

На площадку учреждения, желающего транслировать свой контент определенной аудитории, устанавливается сервер хранения, который напрямую обращается к документам и преобразует их в нужный формат.

Появился корневой элемент архитектуры — центральный сервер Vivaldi, выступающий в роли единой поисковой системы по всем серверам хранения, установленным в различных учреждениях.

**Когда использовать инкрементную модель?**

• Когда основные требования к системе четко определены и понятны. В то же время некоторые детали могут дорабатываться с течением времени.

• Требуется ранний вывод продукта на рынок.

• Есть несколько рисковых фич или целей.

**«RAD Model» (rapid application development model или быстрая разработка приложений)**



RAD-модель — разновидность инкрементной модели.

В RAD-модели компоненты или функции разрабатываются несколькими высококвалифицированными командами параллельно, будто несколько мини-проектов.

Временные рамки одного цикла жестко ограничены.

Созданные модули затем интегрируются в один рабочий прототип. Синергия позволяет очень быстро предоставить клиенту для обозрения что-то рабочее с целью получения обратной связи и внесения изменений.

**Модель быстрой разработки приложений включает следующие фазы:**

• Бизнес-моделирование: определение списка информационных потоков между различными подразделениями.

• Моделирование данных: информация, собранная на предыдущем этапе, используется для определения объектов и иных сущностей, необходимых для циркуляции информации.

• Моделирование процесса: информационные потоки связывают объекты для достижения целей разработки.

• Сборка приложения: используются средства автоматической сборки для преобразования моделей системы автоматического проектирования в код.

• Тестирование: тестируются новые компоненты и интерфейсы.

**Когда используется RAD-модель?**

Может использоваться только при наличии высококвалифицированных и узкоспециализированных архитекторов.

Бюджет проекта большой, чтобы оплатить этих специалистов вместе со стоимостью готовых инструментов автоматизированной сборки.

RAD-модель может быть выбрана при уверенном знании целевого бизнеса и необходимости срочного производства системы в течение 2-3 месяцев.

**«Agile Model» (гибкая методология разработки)**

В «гибкой» методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет.

Это одно из преимуществ гибкой модели.

К ее недостаткам относят то, что из-за отсутствия конкретных формулировок результатов сложно оценить трудозатраты и стоимость, требуемые на разработку.

Экстремальное программирование (XP) является одним из наиболее известных применений гибкой модели на практике.

В основе такого типа — непродолжительные ежедневные встречи — «Scrum» и регулярно повторяющиеся собрания (раз в неделю, раз в две недели или раз в месяц), которые называются «Sprint». На ежедневных совещаниях участники команды обсуждают:

• отчёт о проделанной работе с момента последнего Scrum’a;

• список задач, которые сотрудник должен выполнить до следующего собрания;

• затруднения, возникшие в ходе работы.

Методология подходит для больших или нацеленных на длительный жизненный цикл проектов, постоянно адаптируемых к условиям рынка.

Соответственно, в процессе реализации требования изменяются.

Гибкая разработка лучше всего подходит для психотипа руководителей - творческие люди, которым свойственно генерировать, выдавать и опробовать новые идеи еженедельно или даже ежедневно.

Внутренние стартапы компании разрабатываются по Agile.

Примером клиентских проектов является Электронная Система Медицинских Осмотров, созданная для проведения массовых медосмотров в считанные минуты.

**Когда использовать Agile?**

• в динамическом бизнесе.

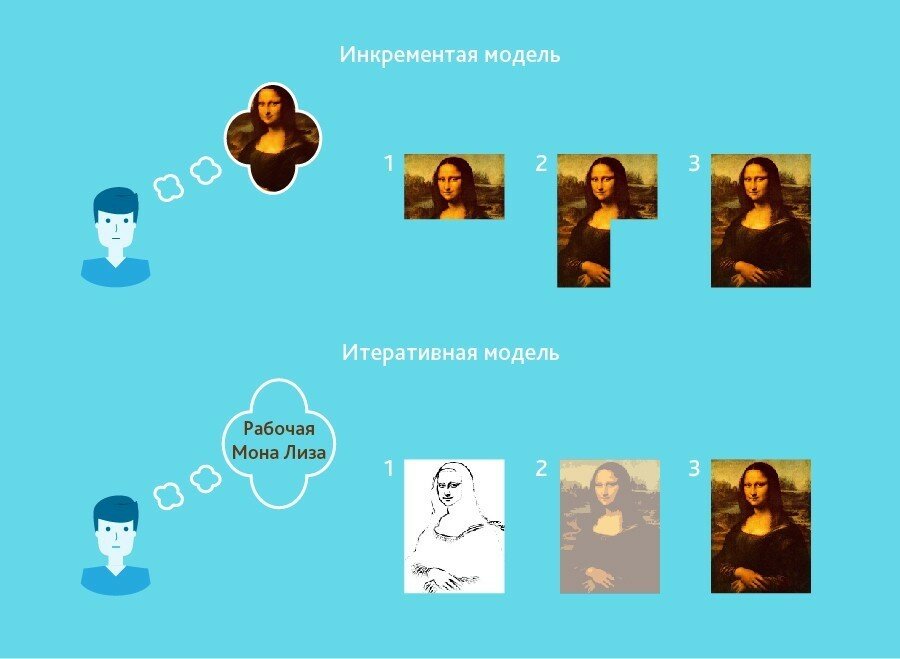
• Изменения на Agile реализуются за меньшую цену из-за частых инкрементов.

• В отличие от модели водопада, в гибкой модели для старта проекта достаточно лишь небольшого планирования.

**«Iterative Model» (итеративная или итерационная модель)**

Итерационная модель жизненного цикла не требует для начала полной спецификации требований.

Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований.



Этот процесс повторяется. Версия может быть неидеальна, главное, чтобы она работала. Понимая конечную цель, идет стремление к ней так, чтобы каждый шаг был результативен, а каждая версия — работоспособна.

Примером итерационной разработки может служить распознавание голоса.

Первые исследования и подготовка научного аппарата начались давно, в начале — в мыслях, затем — на бумаге.

С каждой новой итерацией качество распознавания улучшалось.

Тем не менее, идеальное распознавание еще не достигнуто, следовательно, задача еще не решена полностью.

**Когда оптимально использовать итеративную модель?**

• Требования к конечной системе заранее четко определены и понятны.

• Проект большой или очень большой.

• Основная задача должна быть определена, но детали реализации могут эволюционировать с течением времени.

**«Spiral Model» (спиральная модель)**

«Спиральная модель» похожа на инкрементную, но с акцентом на анализ рисков.



Она хорошо работает для решения критически важных бизнес-задач, когда неудача несовместима с деятельностью компании, в условиях выпуска новых продуктовых линеек, при необходимости научных исследований и практической апробации.

**Спиральная модель предполагает 4 этапа для каждого витка:**

1. планирование;

2. анализ рисков;

3. конструирование;

4. оценка результата и при удовлетворительном качестве переход к новому витку.

Эта модель не подойдет для малых проектов, она резонна для сложных и дорогих, например, таких, как разработка системы документооборота для банка, когда каждый следующий шаг требует большего анализа для оценки последствий, чем программирование.

В современной практике модели разработки программного обеспечения многовариантны. Нет единственно верной для всех проектов, стартовых условий и моделей оплаты.

Даже столь любимая всеми Agile не может применяться повсеместно из-за неготовности некоторых заказчиков или невозможности гибкого финансирования.

Методологии частично пересекаются в средствах и отчасти похожи друг на друга.

**Сопровождение информационных систем (ИС) состоит из двух больших и разноплановых задач.**

Первая задача — эксплуатация информационной системы. Решение этой задачи начинается с установки прикладного программного обеспечения (ПО) в определенном программно-аппаратном окружении и настройкой ПО в соответствии с документацией разработчика таким образом, чтобы обеспечить максимальную надежность и производительность работы приложения. В дальнейшем инженерами службы поддержки обеспечивается функционирование информационной системы с заданными параметрами доступности — и программно-аппаратного окружения, и прикладного ПО.

Вторая задача — внесение изменений в информационную систему. Изменения могут включать донастройки тиражируемого ПО или доработки заказного ПО. И донастройки и доработки, как правило, требуют привлечения консультантов по бизнес-процессам, а также программистов, обладающих необходимыми компетенциями.  
Однако, достаточно часто в договор по сопровождению информационной системы включается и первая, и вторая задачи. При этом договор заключается с одним исполнителем — разработчиком ПО или компанией, внедряющей информационную систему. В таком объединении задач есть серьезный недостаток: разработчик или внедряющая организация, решая в ограниченные сроки задачи внесения изменений в ПО, могут поставить в «боевую» эксплуатацию недотестированную версию ПО, содержащую ошибки.

В этом случае заказчик может оказаться в ситуации, когда версия приложения с доработками или измененными настройками, не функционирует должным образом и сотрудники заказчика не могут выполнять свою работу. Далее исполнитель, в большей степени заинтересованный сдать доработанное ПО, пытается поправить ситуацию и проводит доработки «на ходу», вместо того, чтобы «откатить» систему на предыдущую версию. И достаточно часто это приводит к еще большим простоям бизнеса и потерям.