Лекция №1

1. Основные понятия дисциплины.
2. Основные этапы разработки ПО. Жизненный цикл ПО.
3. Модели ЖЦ ПО.

Основные понятия дисциплины

Программное обеспечение (software) – это набор команд, управляющих работой компьютера. Без программного обеспечения компьютер не сможет выполнять задачи, которые обычно связаны с компьютерами.

Функции программного обеспечения следующие:

- управлять компьютерными ресурсами;
- обеспечивать пользователя всеми инструментами, необходимыми для извлечения пользы из этих ресурсов;
- выполнять роль посредника между пользователем и хранимой информацией.
- Выбор соответствующего потребностям организации программного обеспечения – одна из ключевых задач управляющего персонала.

Программа

Программа (program) – это набор команд для компьютера. Процесс создания или написания программ называется *программированием*, а люди, которые специализируются на этом виде. . Синонимом слову "программа" является термин *"приложение" (application).*

Для того чтобы программа была выполнена, она должна быть загружена в оперативную память компьютера вместе с данными, которые необходимо обработать (обычно говорят *запустить программу* или *запустить на выполнение*). Когда выполнение программы завершено, она выгружается из оперативной памяти компьютера. Все современные компьютеры позволяют загрузить на выполнение несколько программ одновременно.

ОСНОВНЫЕ ТИПЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Все программы, работающие на компьютере, можно условно разделить на три категории:

- **прикладные программы** - непосредственно обеспечивающие выполнение необходимых пользователям работ
- **системные программы**, выполняющие различные вспомогательные функции
- **инструментальные программные системы**, облегчающие процесс создания новых программ для компьютера.

Прикладные программы

К прикладному программному обеспечению (application software) относятся программы, написанные для пользователей или самими пользователями, для задания компьютеру конкретной работы.

Программы обработки заказов или создания списков рассылки – примеры прикладного программного обеспечения.

Программистов, которые пишут прикладное программное обеспечение, называют *прикладными программистами*.

Системные программы

- управление ресурсами компьютера;
- создание копий используемой информации;
- проверка работоспособности устройств компьютера;
- выдача справочной информации о компьютере и др.;

- Системное программное обеспечение (system software) – это набор программ, которые управляют компонентами компьютера, такими как процессор, коммуникационные и периферийные устройства. Программистов, которые создают системное программное обеспечение, называют *системными программистами*.

Системное ПО

Операционная система

- Распределяет ресурсы компьютера
- Планирует использование ресурсов
- Контролирует работу компьютера

Языковые трансляторы

- Интерпретаторы
- Компиляторы

Утилиты

Обслуживание компьютера и периферийных устройств

Прикладное ПО

Языки программирования

- Ассемблер
- FORTRAN
- COBOL
- BASIC
- Pascal
- C

Языки "четвертого поколения"

К программному обеспечению (ПО) относится также вся область деятельности по проектированию и разработке ПО:

- технология проектирования программ (например, нисходящее проектирование, структурное и объектно-ориентированное проектирование и др.);
- методы тестирования программ;
- методы доказательства правильности программ;
- анализ качества работы программ;
- документирование программ;
- разработка и использование программных средств, облегчающих процесс проектирования программного обеспечения, и многое другое.

Существуют следующие группы программного обеспечения

- операционные системы и оболочки;
- системы программирования;
- инструментальные системы;
- интегрированные пакеты программ;
- динамические электронные таблицы;
- системы машинной графики;
- системы управления базами данных (СУБД);
- прикладное программное обеспечение.

операционная система

- *Операционная система* — это комплекс взаимосвязанных системных программ, назначение которого — организовать взаимодействие пользователя с компьютером и выполнение всех других программ.
- Операционная система выполняет роль связующего звена между аппаратурой компьютера, с одной стороны, и выполняемыми программами, а также пользователем, с другой стороны.
- Операционная система обычно хранится во внешней памяти компьютера — *на диске*. При включении компьютера она считывается с дисковой памяти. Этот процесс называется *загрузкой операционной системы*.

В функции операционной системы входит

- осуществление диалога с пользователем;
- ввод-вывод и управление данными;
- планирование и организация процесса обработки программ;
- распределение ресурсов;
- запуск программ на выполнение;
- всевозможные вспомогательные операции обслуживания;
- передача информации между различными внутренними устройствами;
- программная поддержка работы периферийных устройств (дисплея, клавиатуры, дисковых накопителей, принтера и др.).

Оболочка

Оболочки — это программы, созданные для упрощения работы со сложными программными системами, такими, например, как DOS. Они преобразуют неудобный командный пользовательский интерфейс в дружественный графический интерфейс или интерфейс типа "меню". Оболочки предоставляют пользователю удобный доступ к файлам и обширные сервисные услуги.

системы программирования

Система программирования — это система для разработки новых программ на конкретном языке программирования.

Современные системы программирования обычно предоставляют пользователям мощные и удобные средства разработки программ. В них входят:

- компилятор или интерпретатор;
- интегрированная среда разработки;
- средства создания и редактирования текстов программ;
- обширные библиотеки стандартных программ и функций;
- отладочные программы, т.е. программы, помогающие находить и устранять ошибки в программе;
- "дружественная" к пользователю диалоговая среда;
- многооконный режим работы;
- мощные графические библиотеки; утилиты для работы с библиотеками
- встроенная справочная служба;
- другие специфические особенности.

Популярные системы программирования — *Turbo Basic*, *Quick Basic*, *Turbo Pascal*, *Turbo C*.

Интегрированные пакеты

Интегрированные пакеты представляют собой набор нескольких программных продуктов, объединенных в единый удобный инструмент. Наиболее развитые из них включают в себя текстовый редактор, органайзер, электронную таблицу, СУБД, средства поддержки электронной почты, программу создания презентационной графики.

Результаты, полученные отдельными подпрограммами, могут быть объединены в окончательный документ, содержащий табличный, графический и текстовый материал.

Интегрированные пакеты, как правило, содержат некоторое ядро, обеспечивающее возможность тесного взаимодействия между составляющими.

динамические электронные таблицы

Табличный процессор — это комплекс взаимосвязанных программ, предназначенный для обработки электронных таблиц.

Электронная таблица — это компьютерный эквивалент обычной таблицы, состоящей из строк и граф, на пересечении которых располагаются клетки, в которых содержится числовая информация, формулы или текст.

Самые популярные табличные процессоры — *Microsoft Excel*

Пакеты прикладных программ (ППП)

Пакеты прикладных программ (ППП) — это специальным образом организованные программные комплексы, рассчитанные на общее применение в определенной проблемной области и дополненные соответствующей технической документацией.

В зависимости от характера решаемых задач различают следующие разновидности ППП:

- *пакеты для решения типовых инженерных, планово-экономических, общенаучных задач;*
- *пакеты системных программ;*
- *пакеты для обеспечения систем автоматизированного проектирования и систем автоматизации научных исследований;*
- *пакеты педагогических программных средств и другие.*

Пакеты прикладных программ (ППП)

Чтобы пользователь мог применить ППП для решения конкретной задачи, пакет должен обладать средствами настройки (иногда путём введения некоторых дополнений).

Каждый ППП обладает обычно рядом возможностей по методам обработки данных и формам их представления, полноте диагностики, что дает возможность пользователю выбрать подходящий для конкретных условий вариант.

ППП обеспечивают значительное снижение требований к уровню профессиональной подготовки пользователей в области программирования, вплоть до возможности эксплуатации пакета без программиста.

Часто пакеты прикладных программ располагают базами данных для хранения данных и передачи их прикладным программам.

Этапы разработки ПО

Типовой проект включает в себя следующие этапы разработки программного обеспечения:

- *анализ требований к проекту;*
- *проектирование;*
- *разработка;*
- *тестирование продукта;*
- *документация;*
- *внедрение и поддержка.*

Анализ требований к проекту

На этом этапе формулируются цели и задачи проекта, выделяются базовые сущности и взаимосвязи между ними. То есть, создается основа для дальнейшего проектирования системы.

В рамках данного этапа не только фиксируются требования заказчика, но и проводится их формирование – клиентам подбирается оптимальное решение их проблем, определяется необходимая степень автоматизации, выявляются наиболее актуальные для автоматизации бизнес-процессы.

При анализе требований определяются сроки и стоимость разработки ПО, формируется и подписывается ТЗ на разработку программного обеспечения.

Рассмотрим более подробно процедуру анализа требований:

- В процессе сбора требований важно принимать во внимание возможные противоречия требований различных заинтересованных лиц, таких как заказчики, разработчики или пользователи.

- Полнота и качество анализа требований играют ключевую роль в успехе всего проекта. Требования к ПО должны быть документируемые, выполнимые, тестируемые, с уровнем детализации, достаточным для проектирования системы. Требования могут быть функциональными и нефункциональными.

Анализ требований включает три типа деятельности:

- Сбор требований — общение с клиентами и пользователями, чтобы определить, каковы их требования; анализ предметной области.
- Анализ требований — определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; решение этих проблем; выявление взаимосвязи требований.
- Документирование требований — требования могут быть задокументированы в различных формах, таких как простое описание, сценарии использования, пользовательские истории, или спецификации процессов.

Процесс анализа требований к информационной системе включает следующие фазы:

- Разработка требований
  - Выявление требований
  - Анализ требований
  - Спецификация требований
  - Проверка требований
- Управление требованиями

Спецификация требований программного обеспечения

Спецификация требований программного обеспеченияСпецификация требований программного обеспечения (англ. *Software Requirements Specification, SRS*) является полным описанием поведения системы, которая будет создана. Она включает ряд сценариев использования, которые описывают все виды взаимодействия пользователей с программным обеспечением. Сценарии использования также известны как функциональные требования. В дополнении к сценариям использования, спецификация программного обеспечения также содержит нефункциональные (или дополнительные) требования.

Нефункциональные требования — требования, которые налагают дополнительные ограничения на систему (такие как требования эффективности работы, стандарты качества, или проектные ограничения).

Рекомендуемые подходы для спецификации требований программного обеспечения описаны стандартом IEEE 830—1998. Этот стандарт описывает возможные структуры, желательное содержание, и качества спецификации требований программного обеспечения.

Типы требований

Требования систематизируются несколькими способами. Ниже представлены общие классификации требований, которые касаются технического управления:

1. Требования клиентов

Клиенты, это те, кто выполняет основные функции системного проектирования, со специальным акцентом на пользователе системы как ключевом клиенте. Пользовательские требования определят главную цель системы и, как минимум, ответят на следующие вопросы:

- Где система будет использоваться?
- Как система достигнет целей?
- параметры системы являются критическими для достижения цели?
- Как различные компоненты системы должны использоваться?
- Насколько эффективной должна быть система для выполнения назначения?
- Как долго система будет использоваться?

Типы требований

2. Функциональные требования

Функциональные требования объясняют, что должно быть сделано. Они идентифицируют задачи или действия, которые должны быть выполнены. Функциональные требования определяют действия, которые система должна быть способной выполнить, связь входа/выхода в поведении системы.

Типы требований

3. Нефункциональные требования

Нефункциональные требования — требования, которые определяют критерии работы системы в целом, а не отдельные сценарии поведения. Нефункциональные требования определяют системные свойства такие как производительность, удобство сопровождения, расширяемость, надежность, средовые факторы эксплуатации.

Типы требований

4. Производные требования

Требования, которые подразумеваются или преобразованы из высокоуровневого требования. Например, требование для большего радиуса действия или высокой скорости.

Проектирование

Одним из наиболее важных этапов создания сложной системы является этап проектирования.

На основе предыдущего этапа проводится проектирование системы. Во время проектирования разрабатываются проектные решения по выбору платформы, где будет функционировать система языка или языков реализации, назначаются требования к пользовательскому интерфейсу, определяется наиболее подходящая СУБД. Разрабатывается функциональная спецификация ПО: выбирается архитектура системы, оговариваются требования к аппаратному обеспечению, определяется набор орг. мероприятий, которые необходимы для внедрения ПО, а также перечень документов, регламентирующих его использование.

Проектирование ПО – это процесс построения приложений реальных размеров и практической значимости, удовлетворяющих заданным требованиям функциональности и производительности, таких, например, как текстовый редактор, операционная система.

Проектирование

Разнообразные технические вопросы, возникающие в процессе проектирования, обсуждаются со всеми заинтересованными сторонами, включая заказчика. Определяются технологии, которые будут использоваться в проекте, загрузка команды, ограничения, временные рамки и бюджет. В соответствии с уточненными требованиями выбираются наиболее подходящие проектные решения.

Утвержденный дизайн системы определяет перечень разрабатываемых программных компонентов, взаимодействие с третьими сторонами, функциональные характеристики программы, используемые базы данных и многое другое. Дизайн, как правило, закрепляется отдельным документом – дизайн-спецификацией.

На этом этапе для упрощения визуализации процесса проектирования используются так называемые нотации – схематическое выражение характеристик разрабатывемой системы. Основные используемые нотации:

- – Блок-схемы;
- – ER-диаграммы;
- – UML-диаграммы;
- – Макеты – например, нарисованный в фотошопе прототип сайта.

Разработка

После того как требования и дизайн продукта утверждены, происходит переход к следующей стадии жизненного цикла – непосредственно разработке. Здесь начинается написание программистами кода программы в соответствии с ранее определенными требованиями.

Системные администраторы настраивают программное окружение, программисты разрабатывают пользовательский интерфейс программы и логику ее взаимодействия с сервером.

Программирование предполагает четыре основных стадии:

- 1) Разработка алгоритмов – фактически, создание логики работы программы;
- 2) Написание исходного кода;
- 3) Компиляция – преобразование в машинный код;

В результата этапа реализации появляется рабочая версия продукта.

Тестирование продукта

Тестирование тесно связано с такими этапами разработки программного обеспечения как проектирование и реализация. В систему встраиваются специальные механизмы, которые дают возможность производить тестирование системы на соответствие требований к ней, проверку оформления и наличие необходимого пакета документации.

Тести́рование програ́ммного обеспе́че́ния — процесс исследования, испытания программного продукта, имеющий две различные цели:

- продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
- выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации

Результатом тестирования является устранение всех недостатков системы и заключение о ее качестве.

Документация

Этот этап выделяют достаточно условно, поскольку, те или иные документы создаются на всех стадиях жизненного цикла программы. Тем не менее, помимо проектной документации и сопровождающих разработку записей, существуют также и другие текстовые документы, описывающие, например, функции программы и способы ее использования.

Всего существует четыре уровня документации

- – Архитектурная (проектная) – например, дизайн-спецификация. Это документы, описывающие модели, методологии, инструменты и средства разработки, выбранные для данного проекта.
- – Техническая – вся сопровождающая разработку документация. Сюда входят различные документы, поясняющие работу системы на уровне отдельных модулей. Как правило, пишется в виде комментариев к исходному коду, которые впоследствии структурируются в виде HTML-документов.
- – Пользовательская – включает справочные и поясняющие материалы, необходимые конечному пользователю для работы с системой. Это, к примеру, Readmeи Userguide, раздел справки по программе.
- – Маркетинговая – включает рекламные материалы, сопровождающие выпуск продукта. Ее цель – в красочной форме представить функциональность и конкурентные преимущества продукта.

Внедрение и поддержка

Внедрения системы обычно предусматривает следующие шаги:

- установка системы,
- обучение пользователей,
- эксплуатация.

К любой разработке прилагается полный пакет документации, который включает в себя описание системы, руководства пользователей и алгоритмы работы.

Поддержка функционирования ПО должна осуществляться группой технической поддержки разработчика.

Жизненный цикл программного обеспечения

Жизненный цикл программного обеспечения (ПО) — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Этот цикл — процесс построения и развития ПО.

Основные этапы ЖЦ ПО

планирование разработки;
определение требований к системе;
- выработка требований;
- анализ требований;

проектирование системы;
- проектирование архитектуры системы;
- детальное проектирование компонент системы, в т.ч. для программного обеспечения;
- общее проектирование программного обеспечения;
- проектирование отдельных программных компонент;

реализация и тестирование системы;
- создание отдельных компонент системы, в т.ч. для программного обеспечения;
- создание отдельных программных модулей;
- тестирование отдельных программных модулей;
- тестирование компонент системы, в т.ч. программного обеспечения как единого компонента системы;
- интегрирование отдельных компонент в систему;

выпуск системы;
эксплуатация системы;
завершение разработки.

Модель жизненного цикла программного обеспечения

Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.

Выделяют модели ЖЦ ПО: спиральная, каскадная, с промежуточным контролем и V-образная модель.

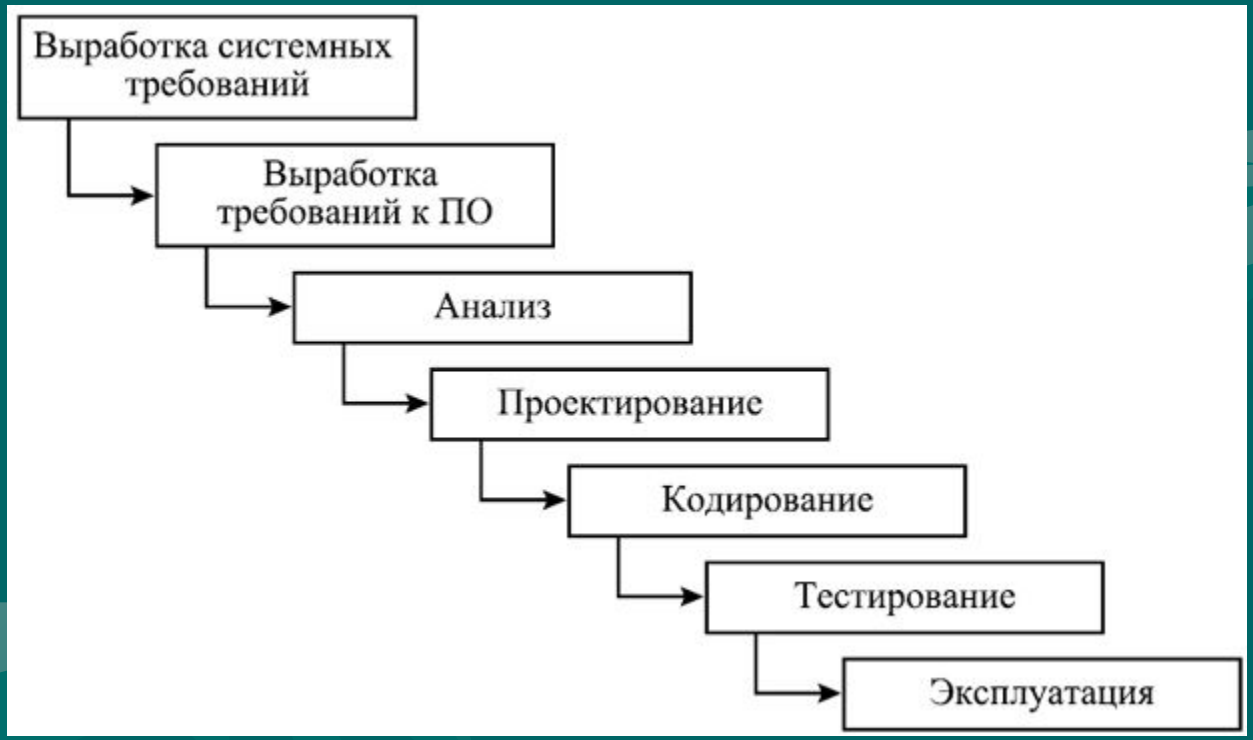
Каскадная модель

Основной характеристикой является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Положительные стороны применения каскадного подхода заключаются в следующем:

- на каждом этапе формируется законченный набор проектной
- документации, отвечающий критериям полноты и согласованности;
- выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадная модель



Каскадная модель

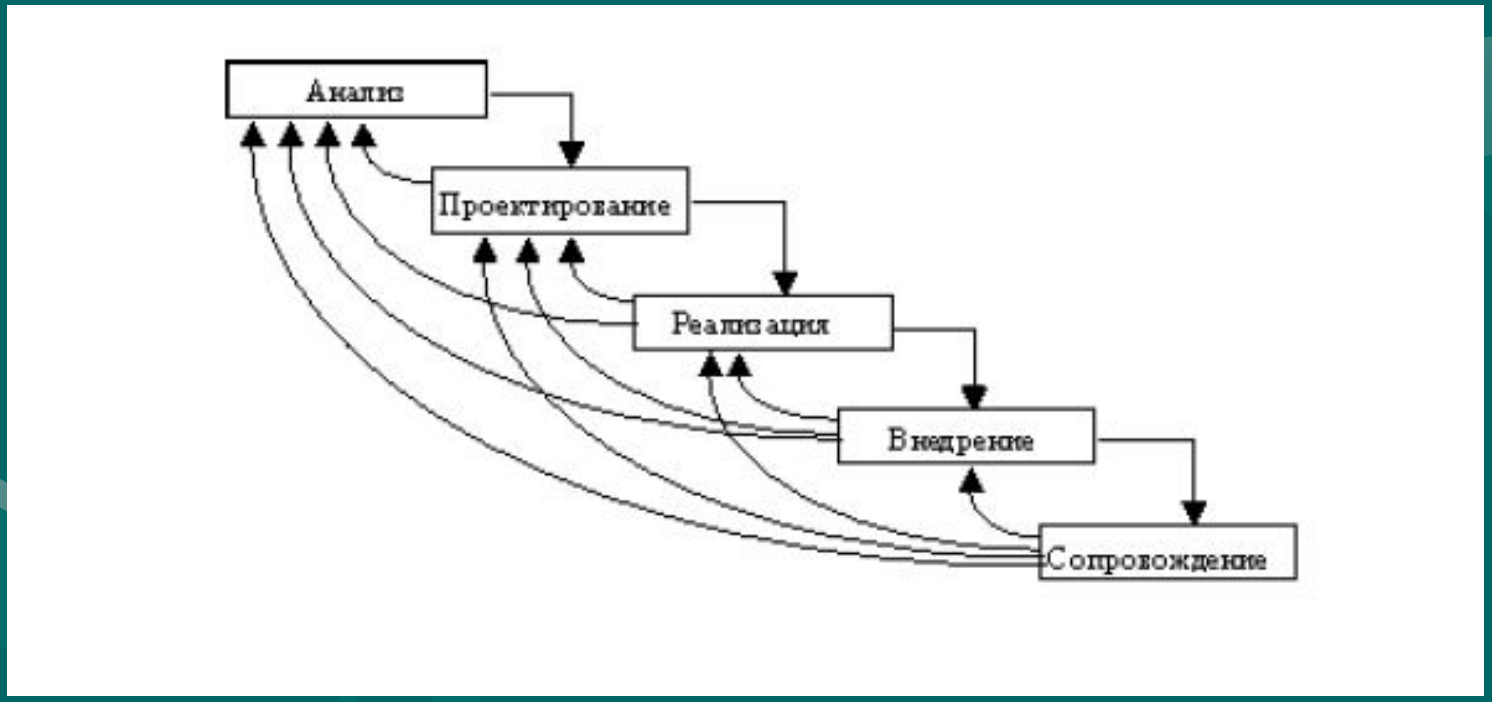
Преимущества:

- Последовательное выполнение этапов проекта в строгом фиксированном порядке
- Позволяет оценивать качество продукта на каждом этапе

Недостатки:

- Отсутствие обратных связей между этапами
- Не соответствует реальным условиям разработки программного продукта

В результате реальный процесс создания ПО принимает следующий вид

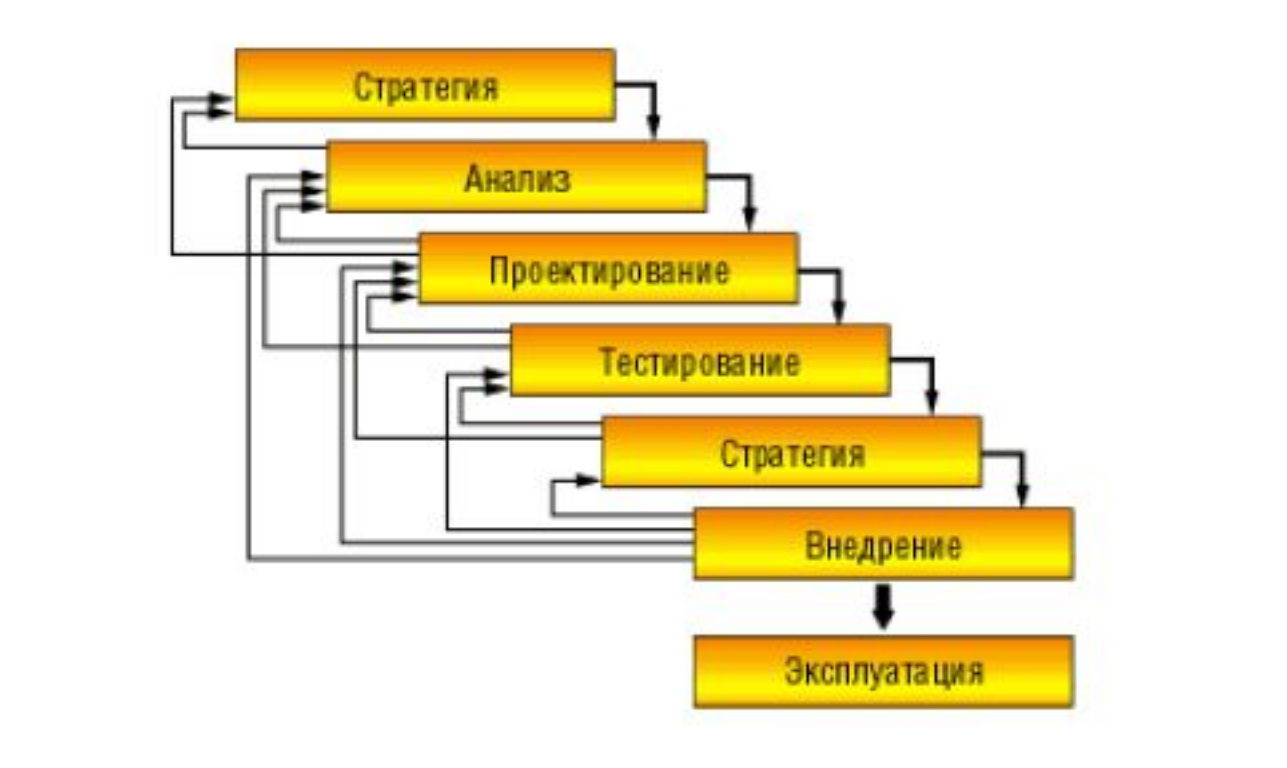


Поэтапная каскадная модель

Для преодоления этих проблем предложена поэтапная модель с промежуточным контролем.

В поэтапной модели с промежуточным контролем разработка ПО ведётся блоками с циклами обратной связи между этапами. Межэтапные корректировки позволяют уменьшить трудоёмкость процесса разработки по сравнению с каскадной моделью. Время жизни каждого из этапов растягивается на весь период разработки.

Поэтапная каскадная модель



V-model

V-модель – это улучшенная версия классической каскадной модели. Здесь на каждом этапе происходит контроль текущего процесса, для того чтобы убедится в возможности перехода на следующий уровень. В этой модели тестирование начинается еще со стадии написания требований, причем для каждого последующего этапа предусмотрен свой уровень тестового покрытия.

Для каждого уровня тестирования разрабатывается отдельный тест-план, то есть во время тестирования текущего уровня, мы также занимаемся разработкой стратегии тестирования следующего. Создавая тест-планы, также определяются ожидаемые результаты тестирования и указываются критерии входа и выхода для каждого этапа.

В V-модели каждому этапу проектирования и разработки системы соответствует отдельный уровень тестирования. Здесь процесс разработки представлен нисходящей последовательностью в левой части условной буквы V, а стадии тестирования – на ее правом ребре. Соответствие этапов разработки и тестирования показано горизонтальными линиями.

Плюсы и минусы V-модели

- + строгая этапизация;
- + планирование тестирования и верификация системы производятся на ранних этапах;
- + улучшенный, по сравнению с каскадной моделью, тайм-менеджмент;
- + промежуточное тестирование.
- - недостаточная гибкость модели;
- - собственно создание программы происходит на этапе написания кода, то есть уже в середине процесса разработки;

Когда использовать V-модель:

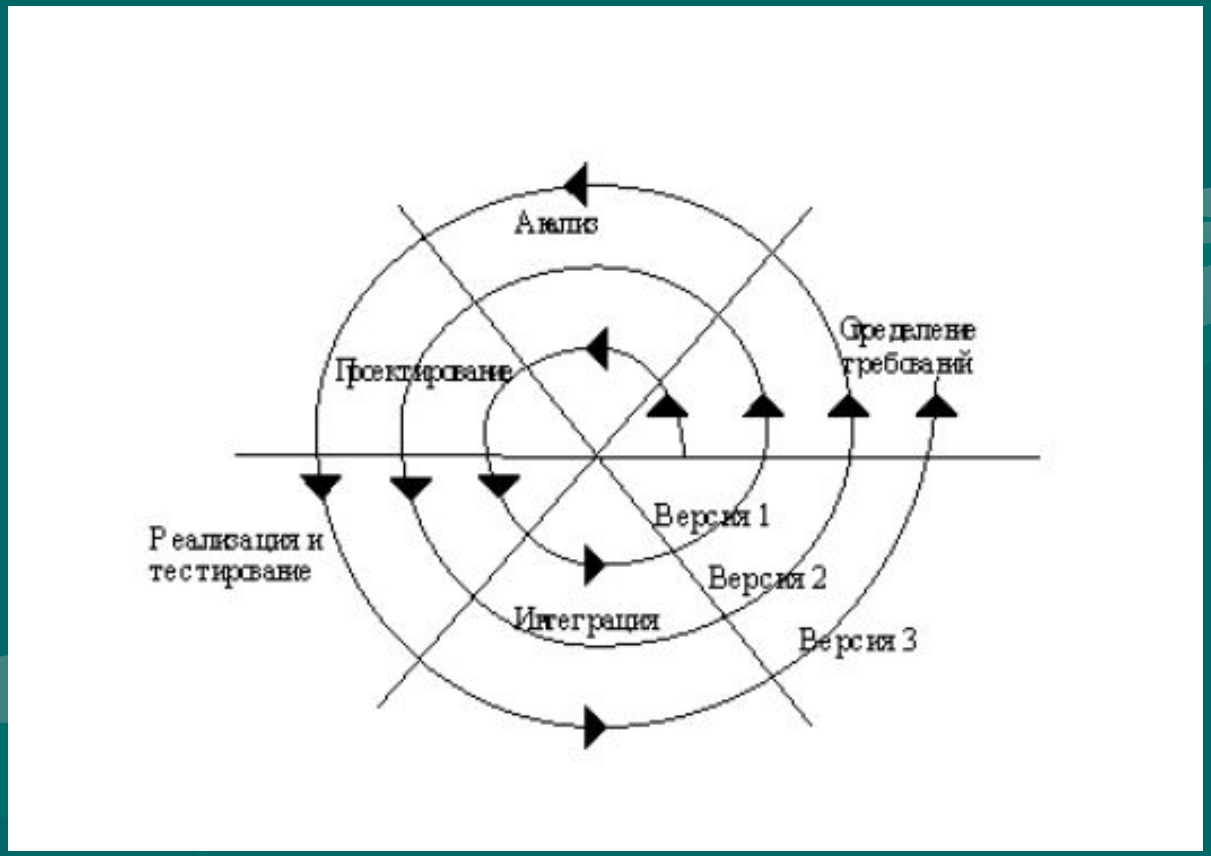
– В проектах, в которых существуют временные и финансовые ограничения;

– Для задач, которые предполагают более широкое, по сравнению с каскадной моделью, тестовое покрытие.

Спиральная модель ЖЦ

Для преодоления проблем, которые возникали при каскадном подходе была предложена спиральная модель жизненного цикла (ЖЦ), делающая упор на начальные этапы ЖЦ: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации.

Спиральная модель ЖЦ



Спиральная модель ЖЦ

Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача - как можно быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Основным нормативные документы

Основным нормативным документом, регламентирующим ЖЦ ПО, является международный стандарт ISO/IEC 12207 (ISO - International Organization of Standardization - Международная организация по стандартизации, IEC - International Electrotechnical Commission - Международная комиссия по электротехнике). Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

Структура ЖЦ ПО по стандарту ISO/IEC 12207 базируется на трех 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).