**Жизненный цикл ПО** – это процесс, на протяжении которого продукт изменяется от состояния бизнес-идеи до готовой реализации. В том или ином виде, жизненный цикл ПО может включть в себя следующие этапы:

1. Определение требований.
2. Спецификация.
3. Проектирование.
4. Кодирование и отладка.

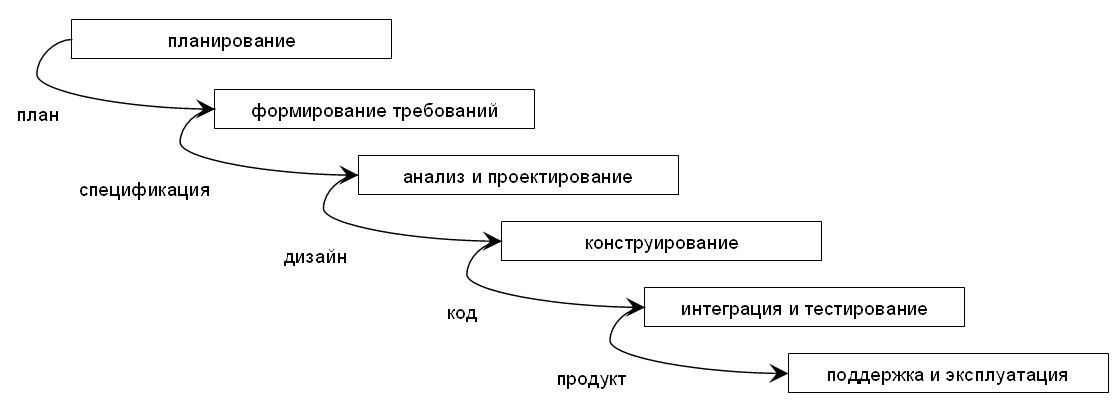
Верификация (доказательство правильности программы, т.е. Доказательство соответствия спецификацие).

Присутствие или отсутствие этапа, а также количество и порядок его возникновения определяется **моделью жизненного цикла ПО**. На сегоднешнем этапе можно выделить три модели:

* каскадная или последовательная
* итеративная и инкрементальная или эволюционная
* спиральная или модель Боэма

**Каскадная (водопадная) модель**

Каскадная, или как её ещё называют, водопадная модель, предполагает строго последовательное во времени и однократное выполнение всех этапов жизненного цикла проекта.



В каскадной модели переход от одной фазы проекта к другой предполагает полную корректность результата (выхода) предыдущей фазы. Однако, например, неточность какого-либо требования или некорректная его интерпретация, в результате, приводит к тому, что приходится

“откатываться” к ранней фазе проекта.

**Итеративная и инкрементальная модель – эволюционный подход**

Итеративная модель предполагает модель разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает “мини-проект”, включая все фазы жизненного цикла в применении к созданию меньших фрагментов функциональности, по сравнению с проектом, в целом. Результата финальной итерации содержит всю требуемую функциональность продукта.

Плюсы:

* можно очень рано начать тестирование пользователями;
* можно принять стратегию разработки в соответствии с бюджетом, полностью защищающую от перерасхода времени или средств

**Спиральная модель**

Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла.

Плюсы:

* Модель предполагает возможность эволюции жизненного цикла, развитие и изменение программного продукта.
* Модель уделяет специальное внимание предотвращению ошибок.
* Модель позволяет контролировать источники проектных работ и соответствующих затрат.
* Модель не проводит различий между разработкой нового продукта и расширением (или сопровождением) существующего. Этот аспект позволяет избежать часто встречающегося отношения к поддержке и сопровождению как ко “второсортной” деятельности.

**Стадии и этапы разработки программного обеспечения:**

* проектирования, кодирования и отладки
* разработки документа по применению ПО
* верификации ПО

**Спецификация:**

На этапе спецификации формируется набор документов, которые будут в последствии использоваться на этапах кодирования и тестирования. Полученный документ должен описывать до мелочей программный продукт.

**Проектирование:**

Результатом этапа проектирования становиться архитектура системы. В процессе разработки архитектуры, уделяется внимание таким аспектам, как цель архитектуры, ее ограничения, возможные альтернативы, а также технологии, используемые в реализации архитектуры.

**Кодирование и отладка:**

На этапе кодирования и отладки в дело вступают языки программирования. Результатом данного этапа – работоспособная программа, включающая в себя исполняемые модули и набор указаний или автоматизированных скриптов для запуска данной программы.

По окончанию этапа проектирования уже сделан выбор, какой язык или языки будут использоваться для реализации ПО. Можно немного рассказать о языках программирования.

**Тестирование:**

По окончанию этапа кодирования, наступает время проверить работоспособность программы. Этап тестирования должен начинаться уже на этапе формирования требований, для того, чтобы еще на концептуальном уровне обнаружить и дать возможность устранить ошибки, которые на последующих этапах станут крайне дорогостоящими, если полностью фатальными для разрабатываемого ПО.

**Документирование:**

Итогом этапа документирования становиться (как ни странно) набор артефактов (чаще всего документов), достаточный для того, чтобы пользователь мог использовать все функции разработанного ПО, доступные в соответствии со спецификацией.

**Сопровождение:**

Сопровождение является финальным этапом разработки ПО. На данном этапе ПО поставляется заказчику и устанавливается в его рабочем окружении. На этапе сопровождения исправляются ошибки в ПО, которые смогли просочиться через этап тестирования.

**Архитектура программных систем**

В строгом значении архитектура программных систем – описание подсистем и компонент программной системы, а также связей между ними. Можно рассказать про архитектурный стиль: про паттерны проектирования.

**Технология объектно-ориентированного программирования**

ООП – технология программирования, в которой основными концепциями являются понятия объектов и классов.

Выделяют три основных парадигмы ООП:

* Инкапсуляция — это принцип, согласно которому любой класс должен рассматриваться как чёрный ящик — пользователь класса должен видеть и использовать только интерфейсную часть класса (т. е. список декларируемых свойств и методов класса) и не вникать в его внутреннюю реализацию.
* Наследованием называется возможность порождать один класс от другого с сохранением всех свойств и методов класса-предка и добавляя, при необходимости, новые свойства и методы.
* Полиморфизмом называют явление, при котором функции (методу) с одним и тем же именем соответствует разный программный код в зависимости от того, объект какого класса используется при вызове данного метода.

**Методы тестирования программного обеспечения**

Тестирование ПО – процесс исследования ПО, с целью получения информации о его качестве.

Существует несколько вариантов классификации методов тестировния ПО, а именно:

**По объекту тестирования:**

* Функциональное тестирование
* Нагрузочное тестирование
  + Тестирование производительности
  + Тестирование стабильности
* Тестирование удобства использования
* Тестирование интерфейса пользователя
* Тестирование безопасности
* Тестирование локализации
* Тестирование совместимости

**По степени автоматизированности:**

* Ручное тестирование
* Автоматизированное тестирование

**По степени изолированности компонентов:**

* Компонентное (модульное) тестирование (тестирования отдельных классов и компонент, обычно применяется самими разработчиками)
* Системное тестирование (тестирование все системы на ее соответствие требованиям)

**По времени проведения тестирования:**

* Альфа тестирование
  + Тестирование при приёмке
  + Тестирование новых функциональностей
  + Регрессионное тестирование (показывает, что внесенные изменения не повлияли на работоспособность уже проверенной функциональности)
  + Тестирование при сдаче
* Бета тестирование (для этой стадии используются группы людей, которые могут стать потенциальными пользователями продукта – продукт между ними распространяется в виде ограниченной версии. Эта группа лиц должна подтвердить наличие малого количества ошибок в программе, а также сформировать первые отзывы о программе в целом).

**По признаку позитивности сценариев:**

* Позитивное тестирование
* Негативное тестирование

**Управление разработкой программного обеспечения**

Управление разработкой ПО – это сложный и многогранный процесс, требующий от управленца знания многих вещей, на первый взгляд не причастных области разработки ПО. На плечи менеджера ложаться такие сложные задачи, как организация команды, управление сроками исполнения проекта, выбор методологии разработки, общение с заказчиком, оценка рисков и т.д.