***Жизненный цикл программного обеспечения (Software Life Cycle Model)*** — это период времени, который начинается с момента принятия решения о создании программного продукта (средства) и заканчивается в момент его полного изъятия из эксплуатации.

В источниках могут встречаться разные наименования стадий, но, как правило, они сводятся к следующим:

* 1. Идея
  2. Анализ требований
  3. Проектирование
  4. Разработка
  5. Тестирование
  6. Стабилизация и релиз
  7. Поддержка и сопровождение

Определенная последовательность стадий, зависящая от ряда условий, в том числе от финансовой составляющей, времени, наличия компетенций в команде и так далее характеризует разные модели жизненного цикла. Далее в этом модуле мы будем рассматривать подробно некоторые из них:

* + **Каскадная** (водопад) — модель, представляющая собой последовательное выполнение каждого из этапов жизненного цикла.
  + **V-модель**— разработка через тестирование, тесты внедряются в каждый из этапов производства.
  + **Спиральная модель** — предполагает цикличное повторение каждой стадии производства из раза в раз, вплоть до выпуска продукта.
  + **Agile**— гибкий подход к процессам разработки и производства, в основе которого лежат быстрый результат и взаимодействие с людьми.
  + **Scrum** — гибкая методика управлениями всеми стадиями жизненного цикла.

**Тест-кейсы** — профессиональная документация тестировщика, последовательность действий, направленная на проверку какого-либо функционала, описывающая, как прийти к фактическому результату.

Вторая модель, которую мы рассмотрим подробно — **спиральная**. Она представляет собой процесс разработки ПО, сочетающий в себе цикличное повторение проектирования, анализа и разработки. Модель была предложена Барри Боэмом в 1986 годы, и на тот момент считалась новаторской и прорывной.

Суть спиральной модели в том, что весь процесс создания конечного продукта представлен в виде условной плоскости, разбитой на четыре сектора:

* + определение целей;
  + оценка рисков;
  + разработка и тестирование;
  + планирование новой итерации.

При прохождении каждого цикла спирали на выходе получаем прототип. Так функционал продукта наращивается с каждым витком, и в итоге доводится до нужного (запланированного) состояния, вплоть до релиза (конец спирали).

**Главная особенность спиральной модели** — концентрация на возможных рисках. Для их оценки даже выделена соответствующая стадия.

Основные **типы рисков**, которые могут возникнуть в процессе разработки ПО:

* 1. Нереалистичный бюджет и сроки;
  2. Дефицит специалистов;
  3. Частые изменения требований;
  4. Чрезмерная оптимизация;
  5. Низкая производительность системы;
  6. Несоответствие уровней квалификации специалистов разных отделов.

Использование спиральной модели будет уместно:

* + при внедрении крупных долгосрочных проектов с отсутствием чётких требований или вероятностью их динамического изменения;
  + когда важен высокопрофессиональный анализ рисков и затрат;
  + при разработке новой линейки продуктов.
* **Основополагающие принципы Agile-манифеста**
* *Мы следуем таким принципам:*
* Наивысшим приоритетом для нас является удовлетворение потребностей   
  заказчика, благодаря регулярной и ранней поставке ценного программного   
  обеспечения.
* Изменение требований приветствуется, даже на поздних стадиях разработки.   
  Agile-процессы позволяют использовать изменения для обеспечения заказчику  
  конкурентного преимущества.
* Работающий продукт следует выпускать как можно чаще, с периодичностью    
  от пары недель до пары месяцев.
* На протяжении всего проекта разработчики и представители бизнеса должны   
  ежедневно работать вместе.
* Над проектом должны работать мотивированные профессионалы. Чтобы   
  работа была сделана, создайте условия, обеспечьте поддержку и полностью   
  доверьтесь им.
* Непосредственное общение является наиболее практичным и эффективным   
  способом обмена информацией как с самой командой, так и внутри команды.
* Работающий продукт — основной показатель прогресса.
* Инвесторы, разработчики и пользователи должны иметь возможность   
  поддерживать постоянный ритм бесконечно. Agile помогает наладить такой   
  устойчивый процесс разработки.
* Постоянное внимание к техническому совершенству и качеству   
  проектирования повышает гибкость проекта.
* Простота — искусство минимизации лишней работы — крайне необходима.
* Самые лучшие требования, архитектурные и технические решения рождаются   
  у самоорганизующихся команд.
* Команда должна систематически анализировать возможные способы   
  улучшения эффективности и соответственно корректировать