**IT-продукт**

Совокупность программно-аппаратных средств (программное решение, основанное на технологической базе), которое предоставляет выполнение заложенных функций, определяющих его суть.

**Жизненный цикл продукта**

Подготовка – Проектирование – Создание - Поддержка

|  |  |  |  |
| --- | --- | --- | --- |
| Подготовка | Проектирование | Создание | Поддержка |
| Анализ конкурентов Сбор инфы Трафик Функционал | Архитектура дизайн  Выбор подрядчика | Разработка отрисовка дизайна  документация | Исправление багов  Новый функционал добавление ассортимента |

**Модель**

Описывает какие стадии жизненного цикла проходит продукт и что происходит на каждой жизненной стадии.

Модели разработки ПО :  
✔ «Waterfall Model»   
✔ V-образная модель   
✔ Спиральная модель   
✔ Инкрементная модель  
 ✔ Итеративная или итерационная модель   
✔ Гибкие модели, методологии и подходы (Agile, Scrum, Kanban и другие)

**«Waterfall Model» (каскадная модель или «водопад»)**

Waterfall  
● Впервые описана в 1970 г. в статье Уинстона Ройса (американский учёный в области информатики, пионер в области разработки ПО)

● Традиционная и самая старейшая модель управления разработкой продукта

■ Предполагает последовательное выполнение всех фаз проекта

■ Итоговый продукт будет получен после завершения всех фаз проекта

● Долгое время рассматривалась как основной способ регулярной разработки ПО

Когда можно использовать Waterfall?

✔ Когда требования известны, понятны и зафиксированы

✔ Нет противоречивых требований к функционалу продукта

✔ Нет проблем с доступностью программистов нужной квалификации

✔ В относительно небольших проектах

Используется:

В медицинской отрасли

В космической отрасли

В строительстве

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| ✔ Старт реализации проекта должен быть отложен до выяснения всех целей и задач проекта и сбора полных требований к нему  ✔ Разработка продукта должна быть хорошо скоординированной, подчиняться разумному планированию и управлению | ✖ Сильный рост стоимости незапланированных изменений требований к продукту на каждом следующем этапе работ ✖ Рост количества проблем в процесс |

**V-образная модель (разработка через тестирование)**

V-образная модель:

● Усовершенствованная каскадная модель

● Заказчик с командой программистов одновременно составляют требования к системе и описывают, как будут тестировать её на каждом этапе



|  |  |
| --- | --- |
| Преимущества | Недостатки |
| ✔ Количество ошибок в архитектуре ПО сводится к минимуму | ✖ Если при разработке архитектуры была допущена ошибка, то вернуться и исправить её будет стоить также дорого, как и в Waterfall |

Когда можно использовать V-модель?

✔ Если требуется тщательное тестирование продукта (validation & verification)

✔ Для малых и средних проектов, где требования чётко определены и фиксированы

✔ В условиях доступности инженеров необходимой квалификации, особенно тестировщиков

**Спиральная модель**



Впервые описана в 1988г. Барри Боемом (американский инженер-программист, профессор программной инженерии) в статье «A Spiral Model of Software Development and Enhancement»

● Работы над проектом представляются как цикл (спираль)

● Каждый виток спирали является водопадной моделью

**Особенности спиральной модели**

Цикл начинается с этапа сбора требований к предполагаемым изменениям, вносимым на данном витке, и завершается реализацией прототипа решается основная проблема традиционных моделей о невозможности изменения требований к продукту

● Заказчик и команда разработчиков серьёзно анализируют риски проекта и выполняют проект итерациями

● Последующая стадия основывается на предыдущей

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| ✔ Уделяется большое внимание проработке рисков | ✖ Есть риск застрять на начальном этапе (бесконечно совершенствовать первую версию продукта и не продвинуться к следующим)  ✖ Разработка длится долго и стоит дорого |

**Инкрементная модель**

Полные требования к ПО делятся на различные сборки

● Поэтапная сборка ПО

● Несколько циклов разработки вместе составляют жизненный цикл «мультиводопад»

● Цикл разделён на более мелкие легко создаваемые модули

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

● Предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций – «инкрементов»

● Процесс продолжается пока не будет создана полная система

**Инкрементная модель.**

Создание соц сети

Пример

1. Заказчик написал ТЗ.

2. Программисты предложили реализовать основные функции, а затем протестировать на пользователях, «взлетит или нет».

3. Продукт разработан и его запускают на рынок.

4. Если пользователям соцсеть нравится, работа над ней продолжается, но уже по частям.

5. Программисты параллельно создают функциональность для загрузки фотографий, обмена документами, прослушивания музыки и других действий, согласованных с заказчиком.

6. Продукт совершенствуется инкремент за инкрементом, приближаясь к изначально созданному ТЗ.

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| ✔ не нужно сразу вкладывать много денег  ✔ можно быстро получить фидбэк от пользователей и обновить ТЗ  ✔ снижается риск создать продукт, который никому не нужен  ✔ ошибка стоит дешевле | ✖ над продуктом работают несколько команд программистов – каждый видит продукт по своему  ✖ разработчики будут оттягивать доработку основной функциональности и «пилить мелочёвку»  ✖ подходит для проектов, в которых точное ТЗ прописано уже на старте, а продукт должен быстро |

**Итеративная или итерационная модель**

Эта модель даёт начало гибким методологиям

Разработка ПО как последовательность итераций

● Каждая итерация сама по себе является небольшим проектом в рамках общей задачи

● Каждая итерация предполагает измеримый прирост ценности продукта по завершении итерации

● Эта модель даёт начало гибким методологиям

● Фокус смещается с обеспечения полноты требований к продукту на формирование процессов слаженной работы команды

Почему всем так понравилась итеративная модель

● Не требует для начала проекта полной спецификации требований

● Создание начинается с реализации части функционала

● Эта часть функционала становится базой для определения дальнейших требований

● Этот процесс повторяется

● Версия может быть неидеальна, главное, чтобы она работала

● Понимая конечную цель, мы стремимся к ней так, чтобы каждый шаг был результативен, а каждая версия — работоспособна

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| ✔ Быстрый выпуск MVP (мини версия проекта)  ✔ Быстрая обратная связь от заказчика и пользователей  ✔ Фокусировка на наиболее важных функциях ПО и улучшении их в соответствии с требованиями рынка и пожеланиями клиента ✔ Постоянное тестирование пользователями позволяет быстро обнаруживать и устранять ошибки | ✖ Использование на начальном этапе баз данных или серверов — первые сложно масштабировать, а вторые не выдерживают нагрузку. Возможно, придётся переписывать большую часть приложения.  ✖ Отсутствие фиксированного бюджета и сроков – заказчик детально не знает, как выглядит конечная цель и когда закончится разработка |

Особенность итеративной модели

✔ Подходит для работы над большими проектами с неопределёнными требованиями

✔ Подходит для задач с инновационным подходом, когда заказчик не уверен в результате

**Гибкие методологии**

Была создана на основе итеративной модели

● Это не модель и не методология, а подход к разработке

● Переводится с английского как «гибкий»

● Включает в себя практики, подходы и методологии, которые помогают создавать ИТ-продукт более эффективно

**Agile**

Экстремальное программирование (Extreme Programming, XP)

✔ Бережливую разработку программного обеспечения (Lean)

✔ Фреймворк для управления проектами Scrum

✔ Разработку, управляемую функциональностью (Feature-driven development, FDD)

✔ Разработку через тестирование (Test-driven development, TDD)

✔ Методологию «чистой комнаты» (Cleanroom Software Engineering)

✔ Итеративно-инкрементальный метод разработки (OpenUP)

✔ Методологию разработки Microsoft Solutions Framework (MSF)

✔ Метод разработки динамических систем (Dynamic Systems Development Method, DSDM)

✔ Метод управления разработкой Kanban

**Scrum**

● Фреймворк – это более сформированная методология со строгими правилами

● Все роли и процессы чётко прописаны

**Kanban**

Cегодня одна из наиболее популярных методологий разработки ПО

Принцип Kanban:

● Команда ведёт работу с помощью виртуальной доски, которая разбита на этапы проекта

● Каждый участник видит, какие задачи находятся в работе, какие — застряли на одном из этапов, а какие уже дошли до его столбца и требуют внимания.

● В отличие от Scrum, при использовании Kanban-подхода можно взять срочные задачи в разработку сразу, не дожидаясь начала следующего спринта.

**Отличия гибких методологий от традиционных (жёстких)**



**Выводы**

В современной практике модели разработки ИТ-продуктов многовариантны

● Нет единственно верной для всех проектов, стартовых условий и моделей оплаты

● Даже гибкие методологии не всегда могут стать универсальным решением всех проблем

● Опытный руководитель проекта / разработки должен найти подходящую методологию или совместить несколько