

# Модели ЖЦ ПО



Подготовила студентка 3-2П9  
Павлова Элина

г. Кострома, 2020 г.

# Оглавление

- Программное обеспечение (ПО);
- Жизненный цикл ПО;
- Модель жизненного цикла ПО;
- Каскадная модель (водопад);
- V-образная модель;
- Инкрементная модель;
- Спиральная модель;
- Гибкая модель;
- Скрам;
- Итерационная модель;
- Модель хаоса;
- Модель быстрой разработки RAD;
- Заключение.

# Программное обеспечение

**Программное обеспечение (ПО)** — программа или множество программ, используемых для управления компьютером.

# Жизненный цикл ПО

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

# Модель жизненного цикла ПО

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

# Каскадная модель (водопад)



# Каскадная модель (водопад)

Характерные черты каскадной модели:

- завершение каждого этапа проверкой полученных результатов с целью устранить как можно большее число проблем, связанных с разработкой изделия;
- циклическое повторение пройденных этапов (как в классической модели).

# Каскадная модель (водопад)

## Плюсы

- все стадии проекта выполняются в строгой последовательности;
- строгость этапов позволяет планировать сроки завершения всех работ и соответствующие ресурсы (денежные и человеческие);
- требования остаются неизменными в течение всего цикла.

# Каскадная модель (водопад)

## Минусы

- сложности при формулировке четких требований и невозможность их изменения;
- тестирование начинается только с середины развития проекта;
- до завершения процесса разработки пользователи не могут убедиться, качествен ли разрабатываемый продукт.

# V-образная модель



# V-образная модель

В этой модели особое значение придается действиям, направленным на **верификацию** и **аттестацию** продукта. Она демонстрирует, что тестирование продукта обсуждается, проектируется и планируется на ранних этапах жизненного цикла разработки. Данная модель стала последователем каскадной модели, так как с ее помощью можно устранить недостатки, которые были ранее.

# V-образная модель

## Плюсы

- строгая этапизация;
- минимизация рисков и устранение потенциальных проблем за счет того, что тестирование появляется на самых ранних стадиях;
- усовершенствованный тайм-менеджмент.

# V-образная модель

## Минусы

- невозможность адаптироваться к измененным требованиям заказчика;
- длительное время разработки (иногда длится до нескольких лет) приводит к тому, что продукт может быть уже не нужен заказчику, поскольку его потребности меняются;
- нет действий, направленных на анализ рисков.

# Инкрементная модель



# Инкрементная модель

- ПО разрабатывается с линейной последовательностью стадий, но **в несколько инкрементов (версий)**. Таким образом улучшение продукта проходит запланированно все время, пока жизненный цикл разработки ПО не завершится.
- Требования к системе определяются в самом начале работы, после чего процесс разработки проводится в виде последовательности версий, каждая из которых является законченным и работоспособным продуктом.

# Инкрементная модель

## Плюсы

- заказчик может дать свой отзыв касательно каждой версии продукта;
- есть возможность пересмотреть риски, которые связаны с затратами и соблюдением графика;
- привыкание заказчика к новой технологии происходит постепенно.

# Инкрементная модель

## Минусы

- функциональная система должна быть полностью определена в начале жизненного цикла для выделения итераций;
- при постоянных изменениях структура системы может быть нарушена;
- сроки сдачи системы могут быть затянуты из-за ограниченности ресурсов (исполнители, финансы).

# Спиральная модель



# Спиральная модель

- весь процесс создания конечного продукта представлен в виде условной плоскости, разбитой на 4 сектора, каждый из которых представляет отдельные этапы его разработки;
- на выходе из очередного витка мы должны получить готовый протестированный прототип;
- прототип, удовлетворяющий всем требованиям – готов к релизу;
- концентрация на возможных рисках.

# Спиральная модель

## Плюсы

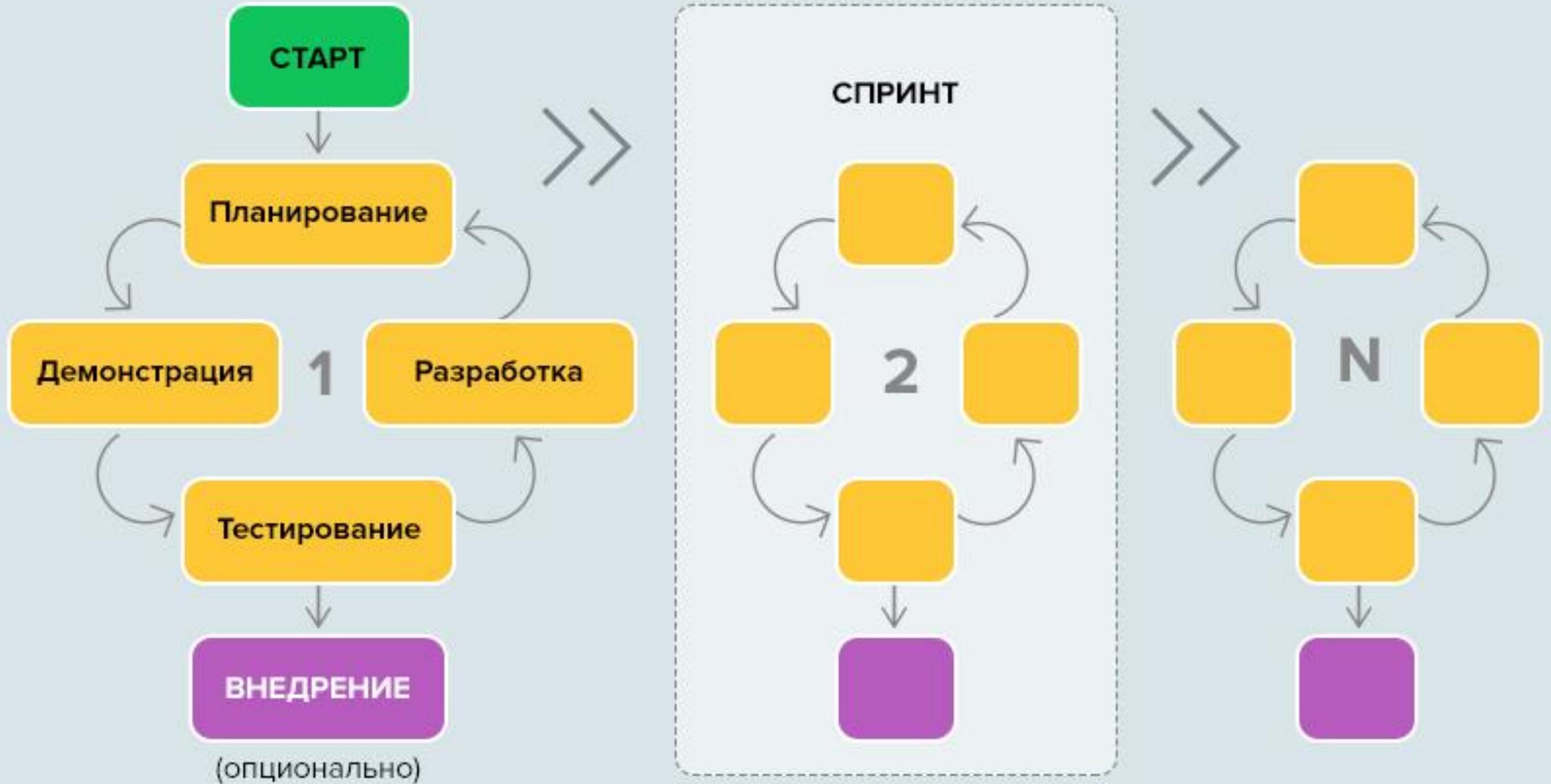
- управлению рисками уделяется особое внимание;
- дополнительные функции могут быть добавлены на поздних этапах;
- есть возможность гибкого проектирования.

# Спиральная модель

## Минусы

- оценка рисков на каждом этапе является довольно затратной;
- постоянные отзывы и реакция заказчика может провоцировать все новые и новые итерации, которые могут приводить к временному затягиванию разработки продукта;
- более применима для больших проектов.

# Гибкая модель



# Гибкая модель

- Представляет собой совокупность различных подходов к разработке ПО.
- Включает серии подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри рабочих групп.
- Отдельная итерация представляет собой миниатюрный программный проект.

# Гибкая модель

## Плюсы

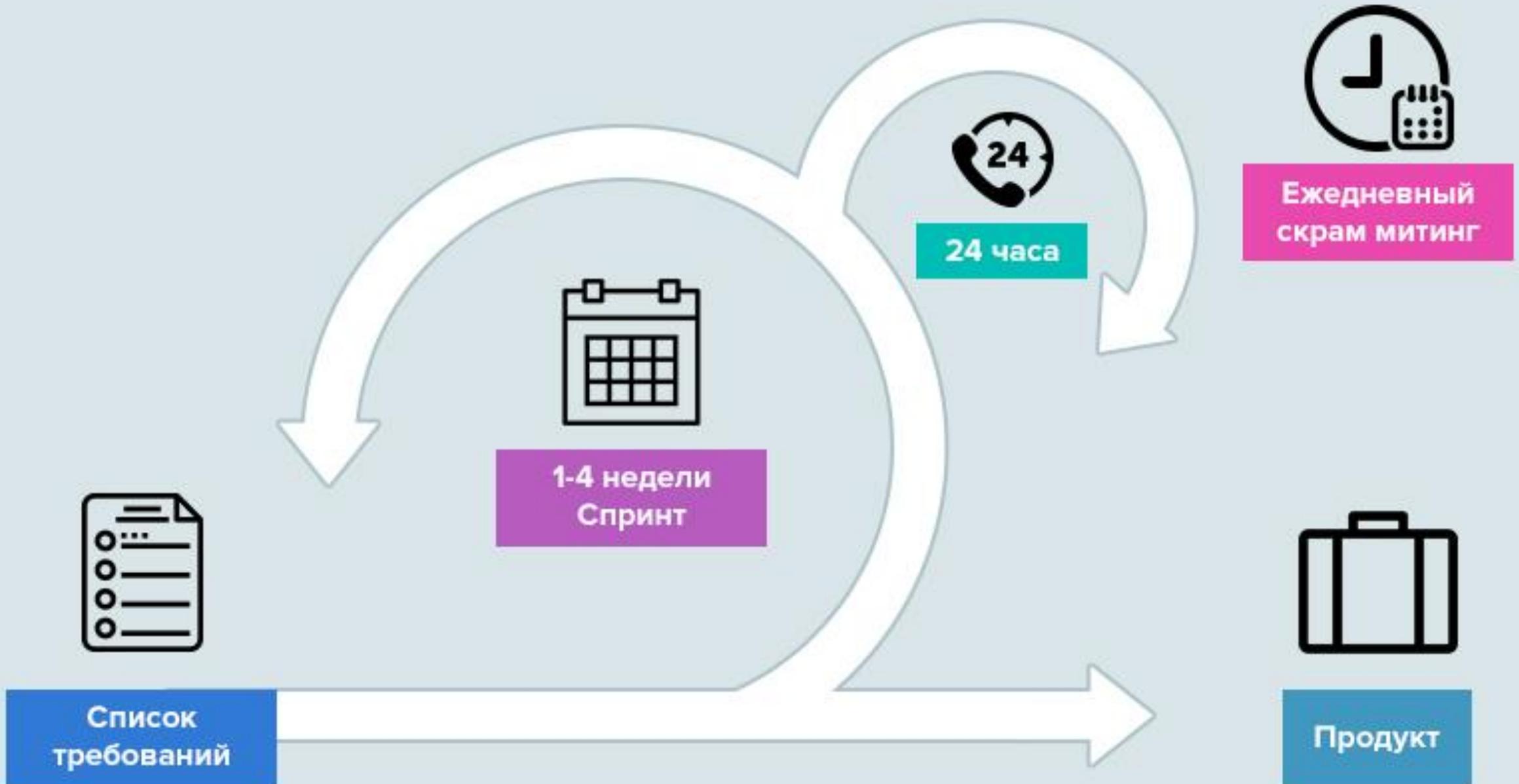
- быстрое принятие решений за счет постоянных коммуникаций;
- минимизация рисков;
- облегченная работа с документацией.

# Гибкая модель

## Минусы

- большое количество митингов и бесед, что может увеличить время разработки продукта;
- сложно планировать процессы, так как требования постоянно меняются;
- редко используется для реализации больших проектов.

# Скрам



# Скрам

**Скрам** – это гибкая модель разработки ПО, в которой делается акцент на качественном контроле процесса разработки.

- Роли в методологии позволяют четко распределить обязанности в процессе разработки.
- Команда – это единое целое, в ней результаты оцениваются не по каждому отдельному участнику, а по тому, что получается в итоге у всех.
- Спринты в данной методологии длятся от 1 до 4 недель. После каждого спринта команда предоставляет вариант законченного продукта.

# Скрам

## Плюсы

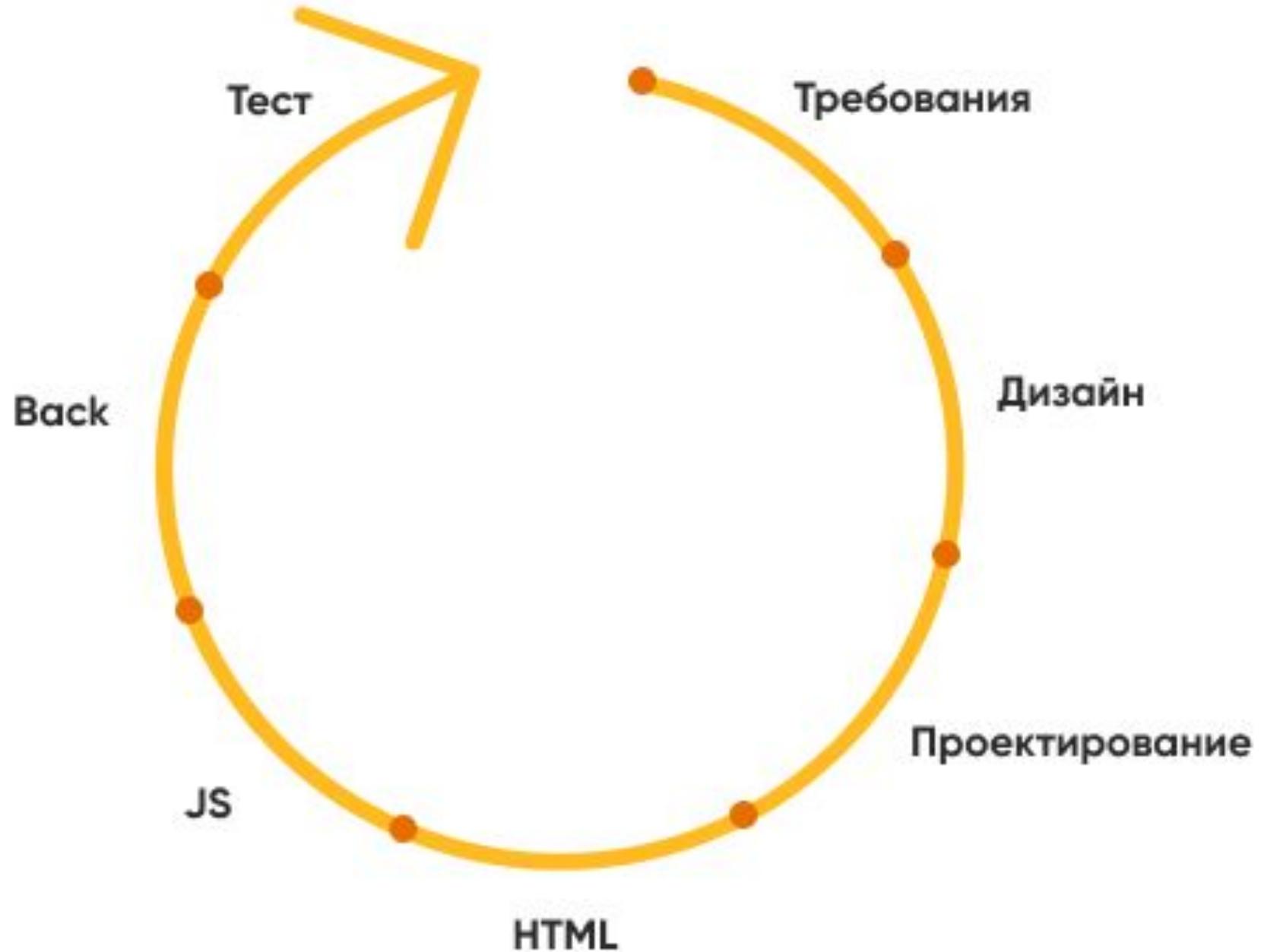
- быстрая обратная связь от специалистов в разных сферах (дизайнеров, архитекторов, тестировщиков и пр.);
- благодаря вовлеченности тестировщика в работу происходит быстрое добавление нового функционала и быстрый запуск продукта с минимальными функциями;
- самостоятельная и самоорганизованная команда.

# Скрам

## Минусы

- некоторые люди, знающие продукт, становятся незаменимыми, так как документация не предоставляется в процессе разработки;
- невозможно спланировать точную дату завершения, так как всё уточняется по результатам предыдущего спринта;
- заказчики не всегда могут понять суть данной методологии и необходимо потратить время на “ликбез”.

# Итерационная модель



# Итерационная модель

- **Итерационная модель** предполагает разбиение проекта на части и прохождение этапов жизненного цикла на каждом из них. **Каждый этап является законченным** сам по себе, совокупность этапов формирует конечный результат.
- На каждой итерации мы работали с одним и тем же продуктом и в конце каждой итерации получали результат, которым можно пользоваться.
- С каждым этапом разработка приближается к конечному желаемому результату или уточняются требования к результату по ходу разработки, и соответственно в любой момент текущая итерация может оказаться последней или очередной на пути к завершению.

# Итерационная модель

## Плюсы

- позволяет бороться с неопределенностью, снимая ее этап за этапом, и проверять правильность технического, маркетингового или любого другого решения на ранних стадиях;
- снижает риски глобального провала и растраты всего бюджета, получение несинхронизированных ожиданий и ошибочного понимания процессов;
- дает возможность завершения разработки в конце любой итерации.

# Итерационная модель

## Минусы

- целостное понимание возможностей и ограничений проекта очень долгое время отсутствует;
- при итерациях приходится отбрасывать часть сделанной ранее работы;
- добросовестность специалистов при выполнении работ снижается, над ними постоянно довлечет ощущение, что «всё равно всё можно будет переделать и улучшить позже».

# Модель хаоса



# Модель хаоса

Основная идея этой модели заключается в том, что программный код представляет собой сложную интеграцию тысяч модулей, функций и строк кода. Этот хаос интеграции требует **метода**, который **определяет интеграцию между всей программой и кодом**, который определяет эту программу.

# Модель хаоса

## Плюсы

- учитывает взаимодействие между членами команды при внесении изменений в код;
- ограничивает риск чрезмерного проектирования решения.
- прозрачность между желаниями руководства высокого уровня и пониманием командой разработчиков проблем и приоритетов.

# Модель хаоса

## Минусы

- критическая необходимость включить единый дизайн на уровне кода, который необходимо выполнить для удовлетворения требований на уровне программы.

# Модель быстрой разработки RAD



# Модель быстрой разработки RAD

Модель RAD, как правило, представляет собой инкрементную модель, в которой множество разработок маленьких кусков выбираются и развиваются одновременно для достижения большей картины. Кроме того, обрабатывается инкрементная модель, в которой основные характеристики, подлежащие разработке, делятся на более мелкие, выполнимые куски. Эти куски затем разрабатываются индивидуально.

# Модель быстрой разработки RAD

## Плюсы

- быстрое развитие продукта;
- разработка многократно мелких компонентов;
- повторный обзор в процессе разработки;
- интеграция повторно используемых компонентов на начальном уровне, следовательно, экономит усилия, несмотря на то, что не добавляются более крупные модули;
- конструктивная реакция.

# Модель быстрой разработки RAD

## Минусы

- требуется много усилий для сбора всех требований на начальном этапе.
- навыки моделирования имеют много зависимостей.
- не подходит для малобюджетного проекта.

# Заключение

Существует множество вариантов моделей разработки ПО. Выбор того или иного варианта зависит от особенностей и требований проекта, моделей оплаты. Частично методологии пересекаются и похожи друг на друга, но тем не менее, каждая находит своих почитателей.

# Список источников

- <https://www.netinbag.com/ru/internet/what-is-the-chaos-model.html>
- <https://ru.photo-555.com/5801526-rad-model>
- <https://evergreens.com.ua/ru/articles/software-development-metodologies.html>
- <https://training.gatestlab.com/blog/technical-articles/popular-software-development-life-cycles/>
- [https://studopedia.ru/7\\_103765\\_iteratsionnaya-model-stadii-dostoinstva-nedostatki.html](https://studopedia.ru/7_103765_iteratsionnaya-model-stadii-dostoinstva-nedostatki.html)
- <https://habr.com/ru/post/111674/>