



# ЖИЗНЕННЫЙ ЦИКЛ ПО



# ЧТО ТАКОЕ ЖИЗНЕННЫЙ ЦИКЛ?

**Жизненный цикл программного обеспечения** (Software Life Cycle Model, ЖЦ) – это *период времени*, который начинается с момента принятия решения о создании программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Этот цикл – процесс построения и развития ПО.

# ОСНОВНЫЕ ЭТАПЫ ЖЦ

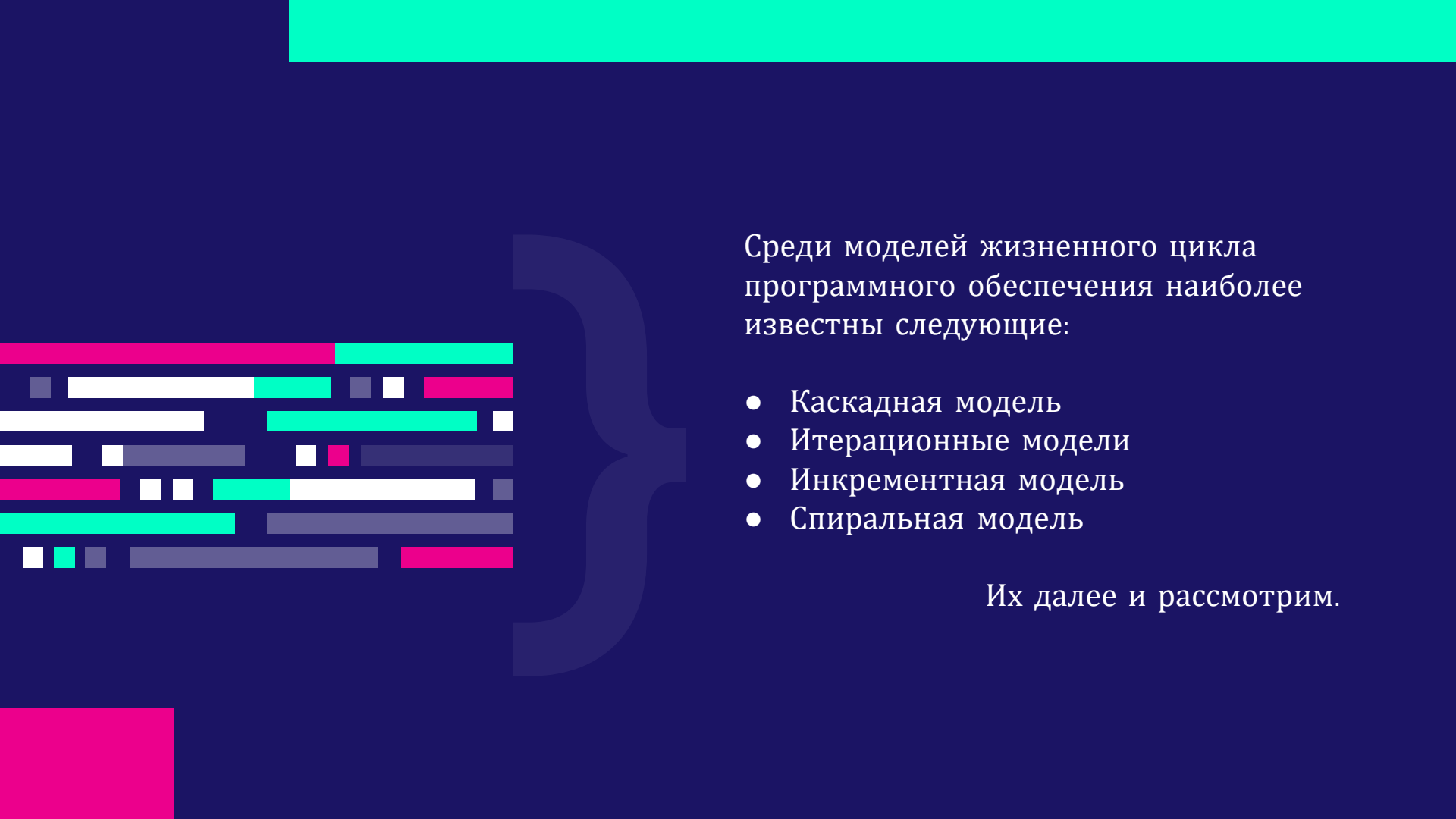
- Анализ требований
- Проектирование
- Программирование
- Тестирование и отладку
- Эксплуатацию,  
сопровождение и поддержку

и т.д.



# Модели

Жизненного цикла



Среди моделей жизненного цикла программного обеспечения наиболее известны следующие:

- Каскадная модель
- Итерационные модели
- Инкрементная модель
- Спиральная модель

Их далее и рассмотрим.

# Waterfall (каскадная модель)

Основная суть модели Waterfall в том, что этапы зависят друг от друга и следующий начинается, когда закончен предыдущий, образуя таким образом поступательное (каскадное) движение вперед.

Параллелизм этапов в каскадной модели, хоть и ограничен, но возможен для абсолютно независимых между собой работ. При этом интеграция параллельных кусков все равно происходит на каком-то следующем этапе, а не в рамках одного.

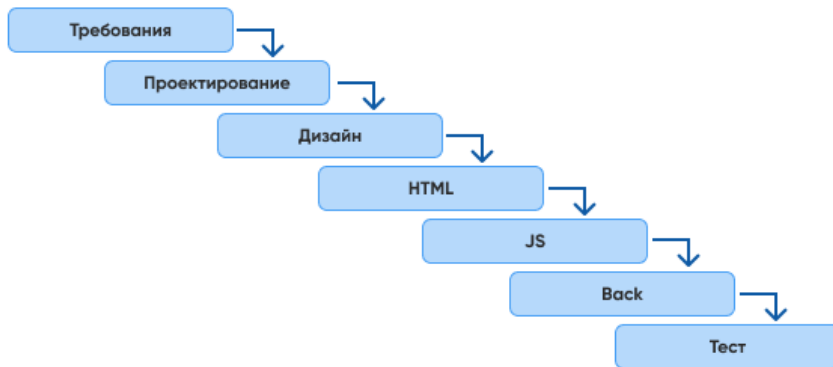
Команды разных этапов между собой не коммуницируют, каждая команда отвечает четко за свой этап.

Недостатками этой модели являются получение результата по прохождению всех этапов и сложность выявления ошибок. Возвращаться назад трудно. Не понятно что возвращать: если произошел сбой на каком-то этапе, его последствия видны только в конце.

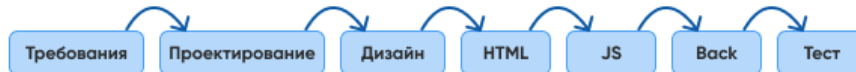
Данная модель понятно и чисто укладывается в документы, например в договора и роадмапы при наличии четко обозначенных контрольных точек. В любой момент времени можно легко понять была ли пройдена та или иная точка контроля или нет, и соблюдены ли сроки. По этим причинам долговременные и особо крупные проекты, рассчитанные на десятилетия и вовлечение большого числа организаций-участников, руководствуются преимущественно waterfall.

## Каскадная модель

Как работает изнутри



Как видит клиент



# Итерационная модель

**Итерационная модель** предполагает разбиение проекта на части (этапы, итерации) и прохождение этапов жизненного цикла на каждом из них. Каждый этап является законченным сам по себе, совокупность этапов формирует конечный результат.

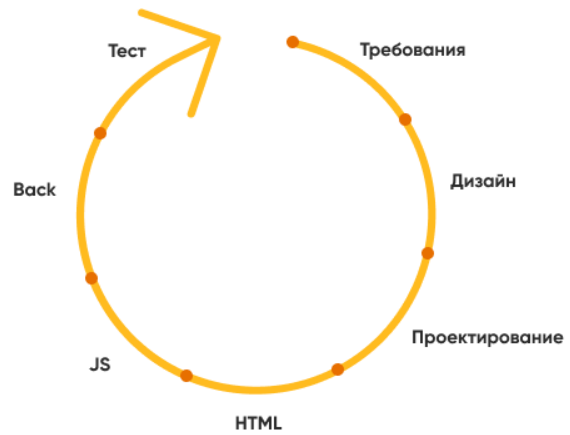
С каждым этапом разработка приближается к конечному желаемому результату или уточняются требования к результату по ходу разработки, и соответственно в любой момент текущая итерация может оказаться последней или очередной на пути к завершению.

Данный подход позволяет бороться с неопределенностью, снимая ее этап за этапом, и проверять правильность технического, маркетингового или любого другого решения на ранних стадиях.

Использование итерационной модели снижает риски глобального провала и растраты всего бюджета, получение несинхронизированных ожиданий и ошибочного понимания процессов как клиентом, так и каждым участником команды разработки. Оно также дает возможность завершения разработки в конце любой итерации (в каскадной модели вы должны прежде завершить все этапы).



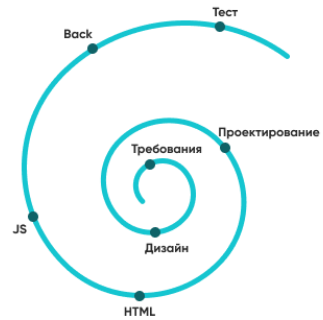
## Итерационная модель



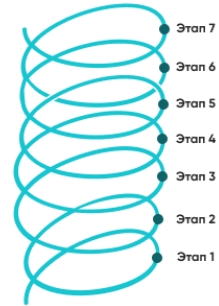
# СПИРАЛЬНАЯ МОДЕЛЬ

Все этапы жизненного цикла при спиральной модели идут витками, на каждом из которых происходят проектирование, кодирование, дизайн, тестирование и т. д. Такой процесс отображает суть названия: поднимаясь, проходится один виток (цикл) спирали для достижения конечного результата. Причем не обязательно, что один и тот же набор процессов будет повторяться от витка к витку. Но результаты каждого из витков ведут к главной цели.

## Спиральная модель



Как работает изнутри



Как видит клиент

# ИНКРЕМЕНТНАЯ МОДЕЛЬ

Принцип, который лежит в основе инкрементной модели, подразумевает расширение возможностей, достраивание модулей и функций приложения. Буквальный перевод слова инкремент: «увеличение на один». Это «увеличение на один» применяется в том числе для обозначения версий продукта.

Если в каскадной модели по сути есть два состояния продукта: «ничего» и «готовый продукт», то с появлением итерационных моделей стало применяться версионирование продукта. Каждая итерация обозначается цифрой: 1,2,3 и соответственно продукт после каждой итерации имеет версию с соответствующим номером: v.1, v.2, v.3. Числами после слова версия обозначают масштабные изменения в ядро продукта.

А когда одна из версий эксплуатируется, следующая, учитывая недочеты предыдущей, только планируется или уже разрабатывается, а улучшения заказчику и пользователю хочется доставить прямо сейчас, тогда появляются минорные версии. Туда попадают изменения, которые не влияют на ядро разработки и представлены как под-версии 1.1,1.2,1.3 или релизы 1.1.1, 1.1.2 и т.п.

В совокупности такие поэтапные релизы приводят к полноценной версии 2.0.

