Testing processes and software development
Процессы тестирования и разработки ПО

What is project?

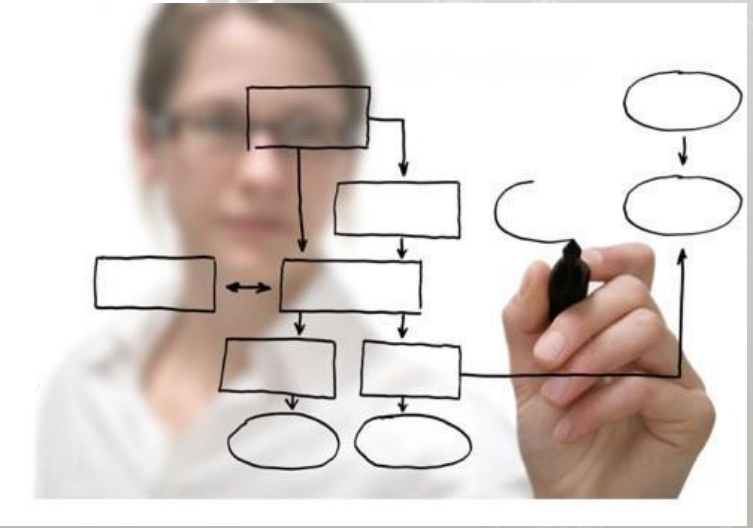
Проект (*project*) - это деятельность по достижению нового результата в рамках установленного времени с учетом определенных ресурсов;

- это временное предприятие, направленное на создание уникальных продуктов, услуг или результатов;

- это все, что имеет «начало, конец и цель».

Признаки проекта

1. Есть конкретная дата начала
2. Есть конкретная дата конца или конечный результат
3. Результат проекта уникален
4. Ресурсы ограничены



Software Development Life Cycle

Жизненный цикл программного обеспечения (*Software Life Cycle*) — ряд событий, происходящих с системой с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

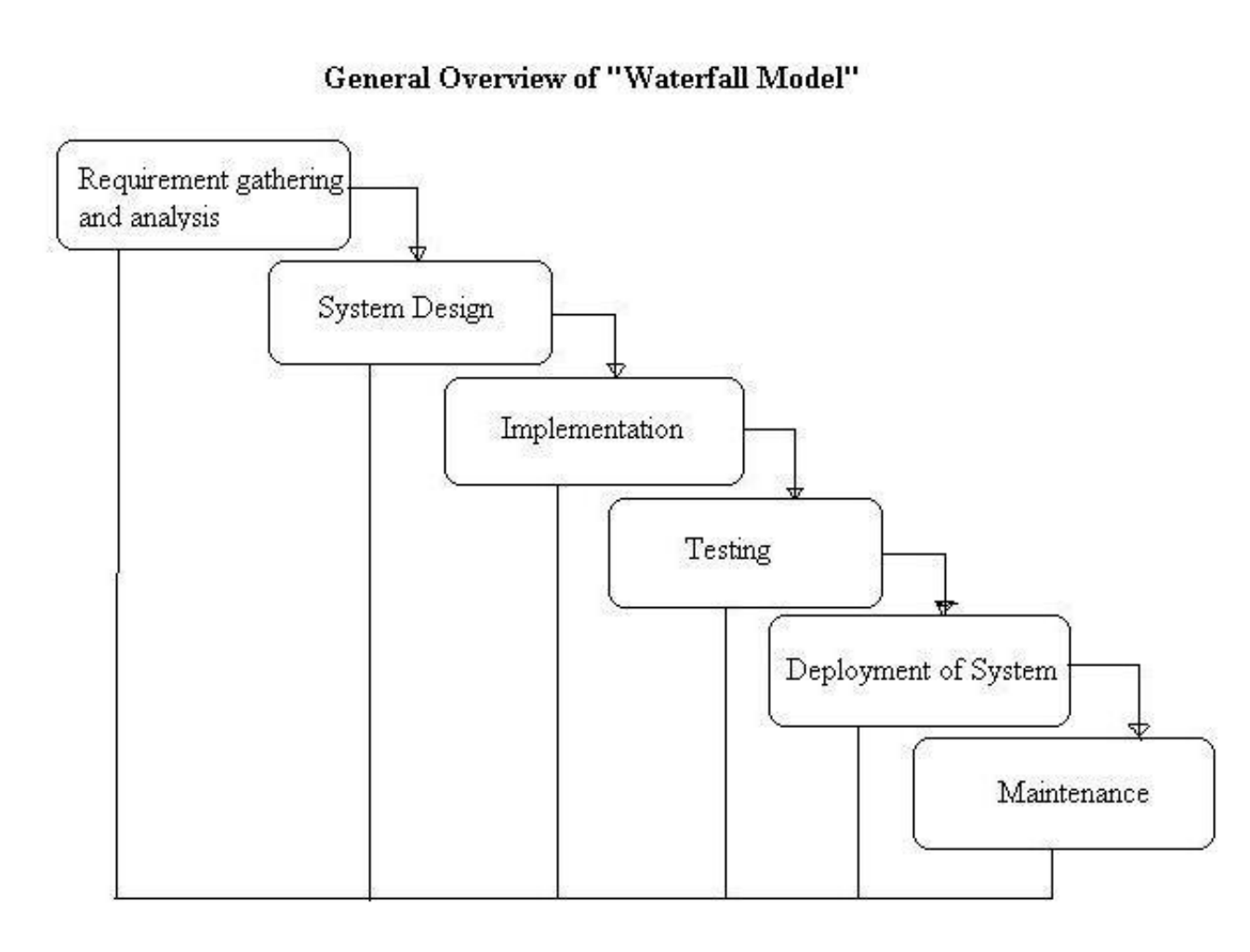
Разработка ПО (***Software Development***) является лишь частью жизненного цикла ПО

Software Development Model

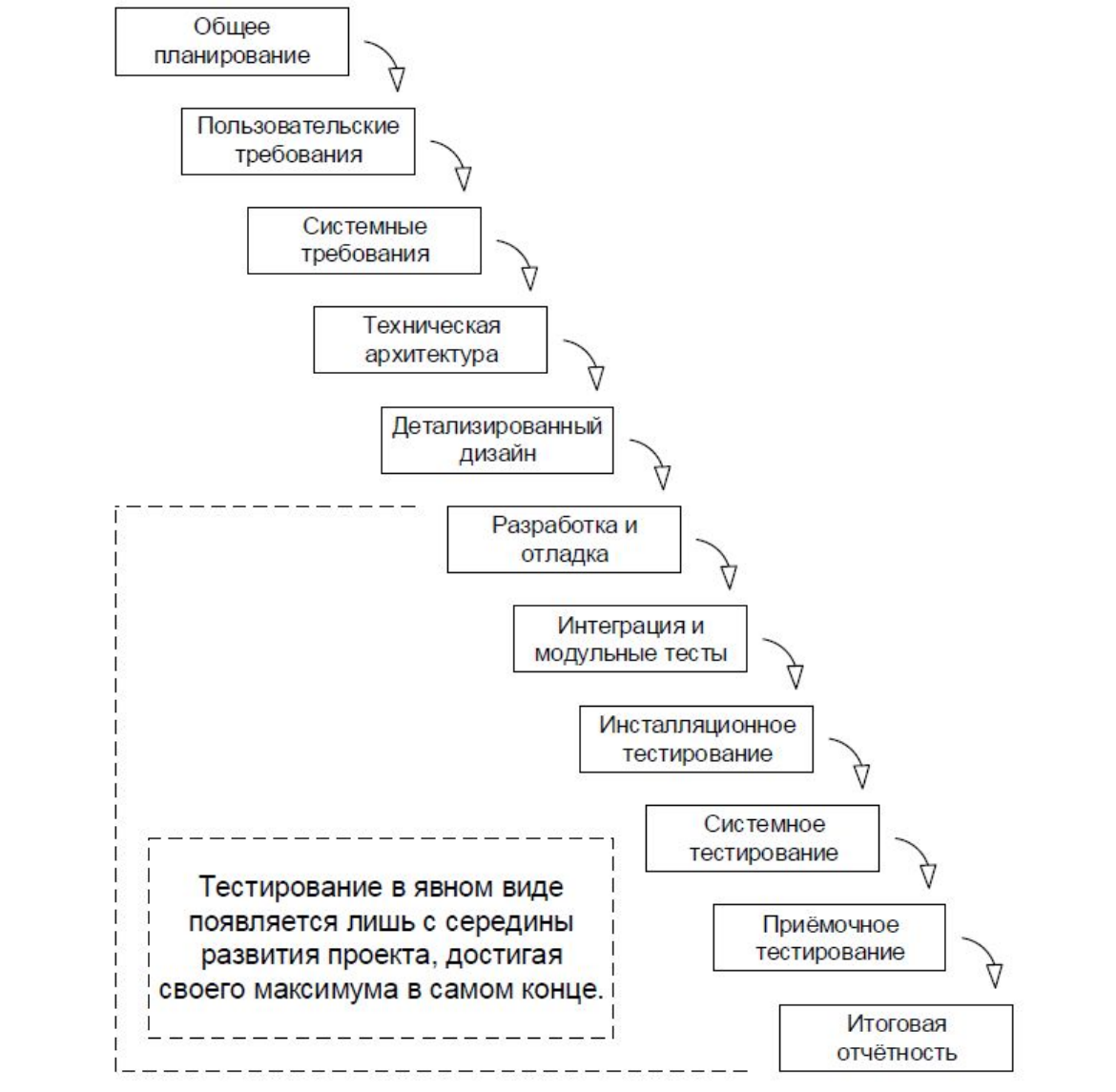
Модель разработки ПО (**Software Development Model, SDM**) — структура, систематизирующая различные виды проектной деятельности, их взаимодействие и последовательность в процессе разработки.

Эти модели можно разделить на 3 **основных группы**:
- Каскадные модели, ***cascade models*** (Waterfall, V-model)
- Итеративные модели, ***iterative models*** (incremental model, RUP, Spiral model)
- Гибкие модели, ***flexible models*** *(*XP, Agile, Scrum, Kanban)

Waterfall model
(Linear-Sequential life cycle model)



Waterfall model



Waterfall model

- Проста для понимания и использования
- Однократное выполнение каждой из фаз проекта
- Каждая фаза должна быть полностью закончена перед началом следующей.
- Применима на небольших проектах
- Все требования должны быть определены
- В завершении каждой фазы проводится обзор (review)
- Тестирование начинается когда разработка полностью завершена
- Фазы проекта не перекрываются

Advantages and disadvantages of waterfall model

Преимущества

- Простота понимания и использования
- Легкость управления из-за жесткости модели – каждая фаза имеет конкретные результаты и процесс обзора
- Фазы не перекрываются и находятся в стадии выполнения и завершения единожды

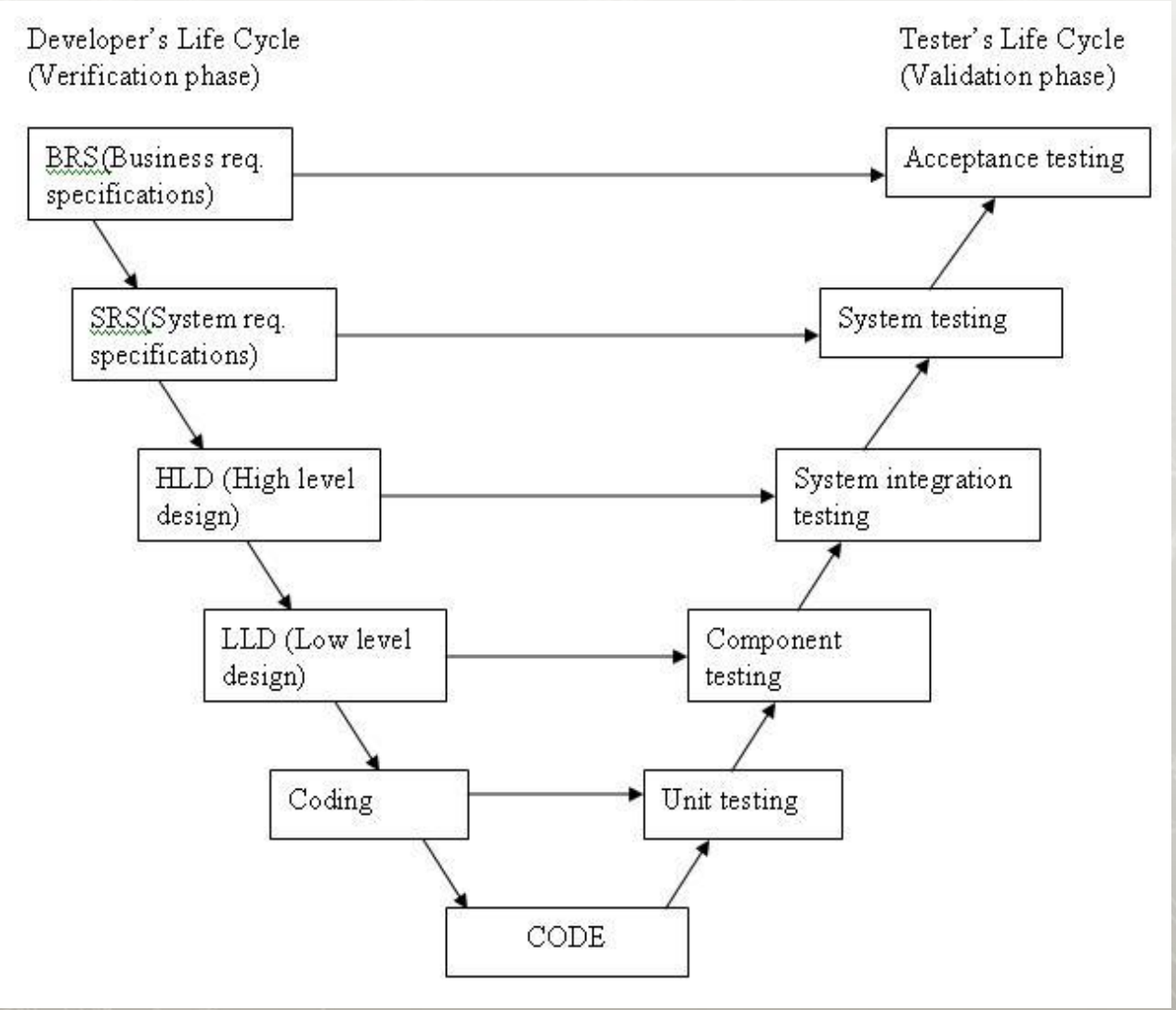
Недостатки

- Во время тестирования сложно исправить что-то на концептуальном уровне
- Работоспособное ПО появляется только на последних
- Высокая степень риска и неопределенности
- Не подходит для длительных, сложных и непрерывно меняющихся проектов
- Не подходит для проектов с высокой вероятностью изменений требований

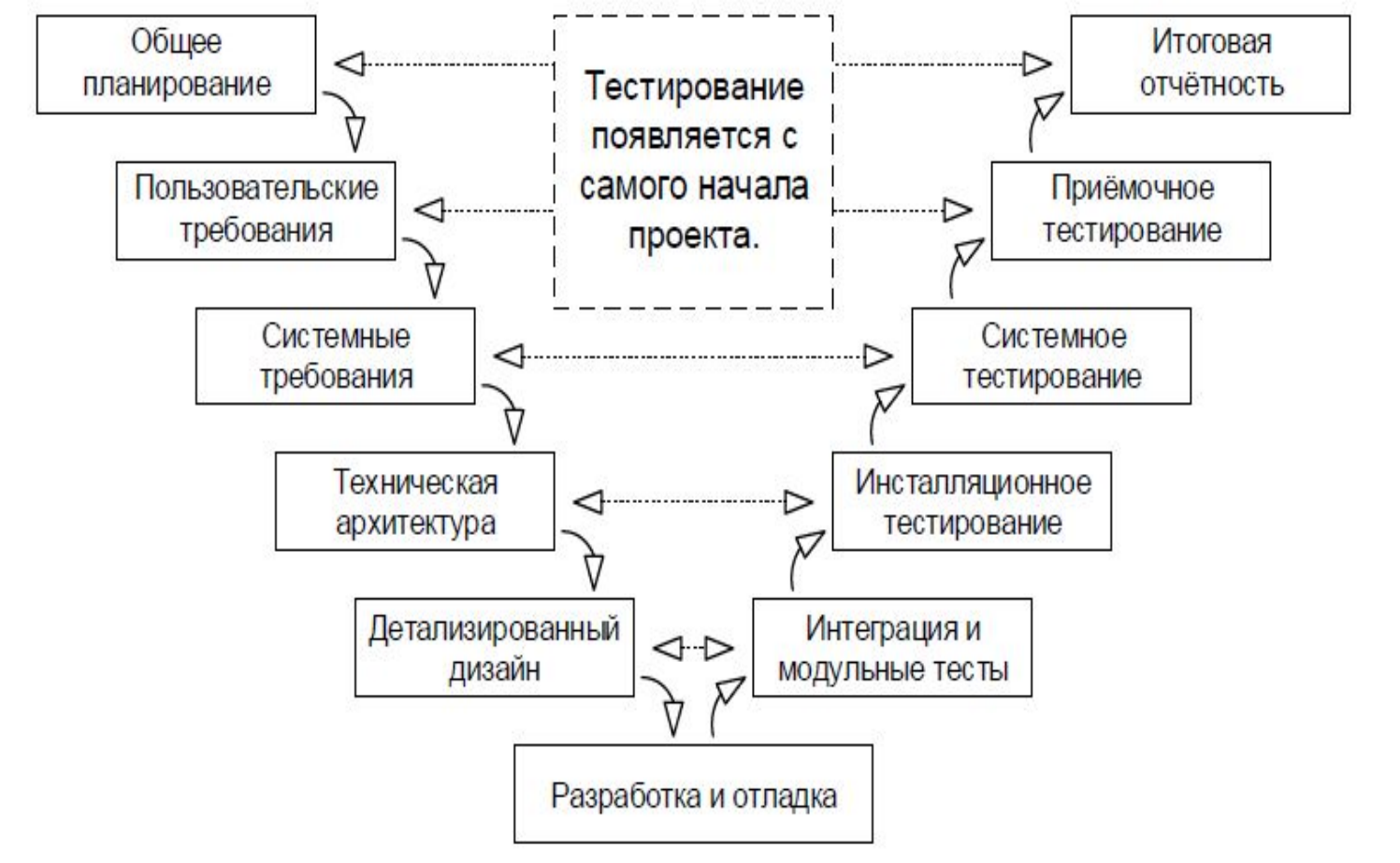
When to use the waterfall model?

- Эта модель используется когда требования четко известны, понятны и зафиксированы
- Разрабатываемые характеристики стабильные
- Технология понятна
- Нет неоднозначных требований
- Проект короткий
- Ресурсов с необходимым опытом достаточно в свободном доступе

V-model


Developer's Life Cycle
(Verification phase)
Tester's Life Cycle
(Validation phase)
BRS(Business req. specifications)
Acceptance testing
SRS(System req. specifications)
System testing
HLD (High level design)
System integration testing
LLD (Low level design)
Component testing
Coding
Unit testing
CODE

V-model


Общее планирование
Пользовательские требования
Системные требования
Техническая архитектура
Детализированный дизайн
Разработка и отладка
Тестирование появляется с самого начала проекта.
Итоговая отчётность
Приёмочное тестирование
Системное тестирование
Инсталляционное тестирование
Интеграция и модульные тесты

Phases of the V-model

The various phases of the V-model are as follows:

Requirements like **BRS** and **SRS** begin the life cycle model just like the waterfall model. System plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The **high-level design** (**HLD**) phase focuses on system architecture and design. An integration test plan is created.

The **low-level design** (**LLD**) phase is where the actual software components are designed. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created.

The **implementation** phase is where all coding takes place. **Coding** is at the bottom of the V-Shape model. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Advantages and disadvantages of V-model

Преимущества

- Простота и легкость использования
- Деятельность по тестированию происходит задолго до кодирования
- Дефекты могут быть обнаружены на ранних стадиях
- Избегание нисходящих дефектов
- Хорошо работает для небольших проектов, где требования хорошо понимаемы.

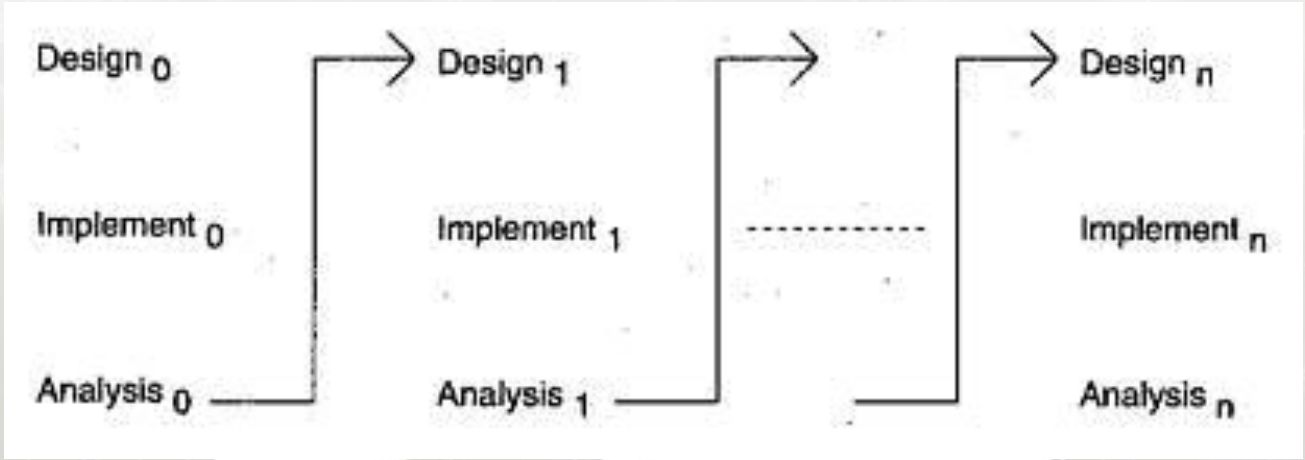
Недостатки

- Жесткость этапов
- Ранние прототипы программного обеспечения не производится
- Если какие-либо изменения происходят на полпути, то тестовые документы наряду с требованиями должны быть обновлены

When to use the V-model?

- V-образная модель может быть использована для малых и средних проектов, где требования четко определены и зафиксированы
- V-образная модель может быть выбрана, когда доступно достаточно технических ресурсов с необходимой подготовкой

Iterative model



Advantages and disadvantages of Iterative model

Преимущества:

- Строим и улучшаем продукт шаг за шагом, следовательно можем отслеживать дефекты на ранних стадиях. Это позволяет избежать нисходящего потока дефектов.
- Надежная обратную связь с пользователем.
- Меньше времени на документацию – больше на дизайн.

Недостатки:

Каждая фаза итерации жесткая и этапы не перекрываются
Возможны повышенные расходы на системную архитектуру или вопросы дизайна, потому что не все требования предварительно собраны для всего жизненного цикла.

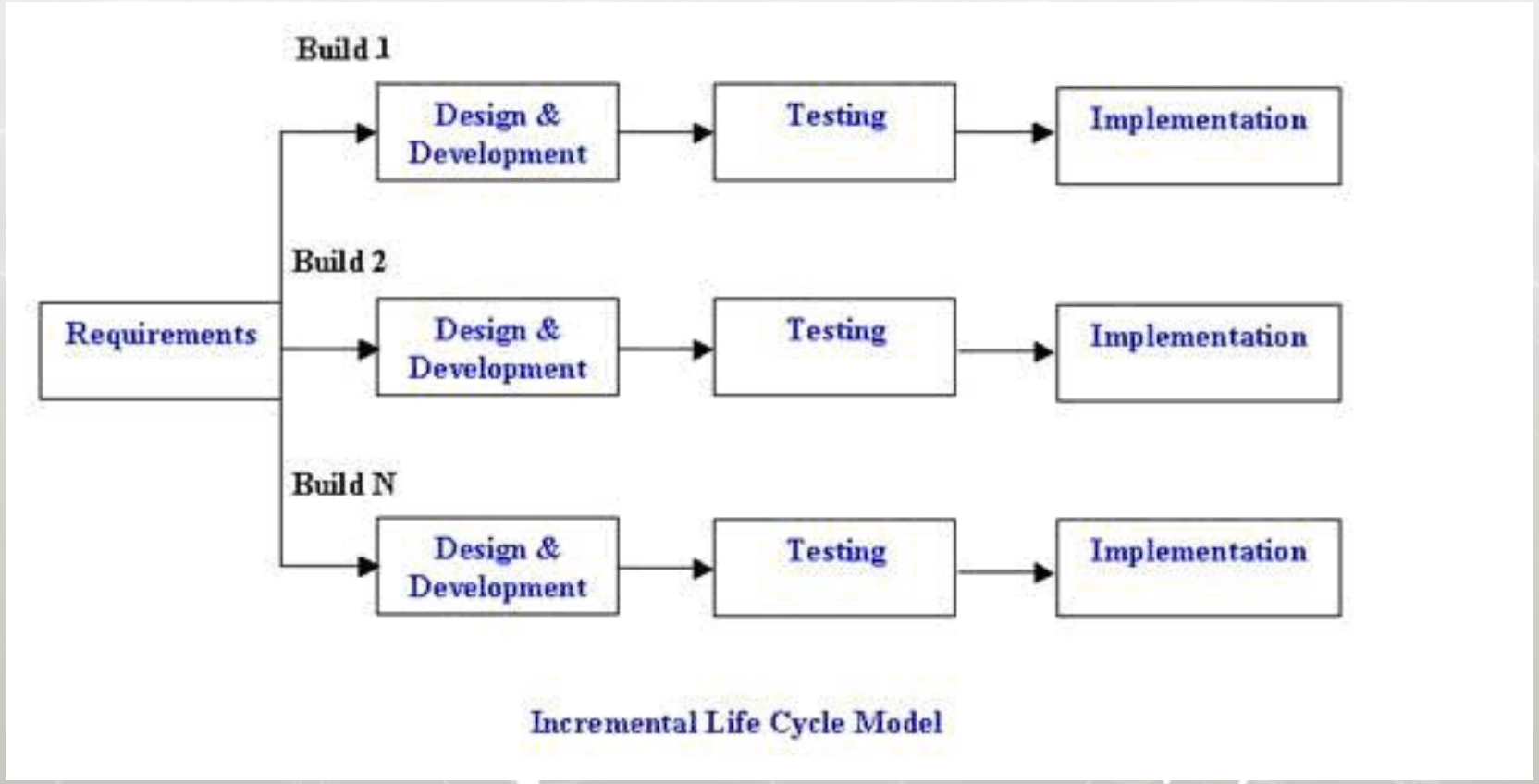
When to use iterative model?

Требования в целом для системы определены и понятны

Подходит для больших проектов

Основные требования должны быть определены, однако некоторые детали могут добавляться со временем

Incremental model



Incremental Life Cycle Model

Incremental model

Когда мы работаем инкрементально, мы добавляем кусок за куском, ожидая что каждый кусочек является полностью законченным. Таким образом продолжается добавление кусочков до завершения системы.

Advantages of Incremental model

Преимущества инкрементальной модели:

Быстрое создание работающего ПО на ранних стадиях жизненного цикла ПО

Эта модель является более гибкой - менее дорогостояще изменить объем работ (scope) и требования.

Тестирования и отлаживать код легче в рамках более мелких итераций

В этой модели заказчик может реагировать на каждый билд

Более низкая стоимость исходной поставки ПО

Легче управлять рисками, потому что рискованные части идентифицируются и обрабатываются во время соответсвующей итерации.

Disadvantages of Incremental model

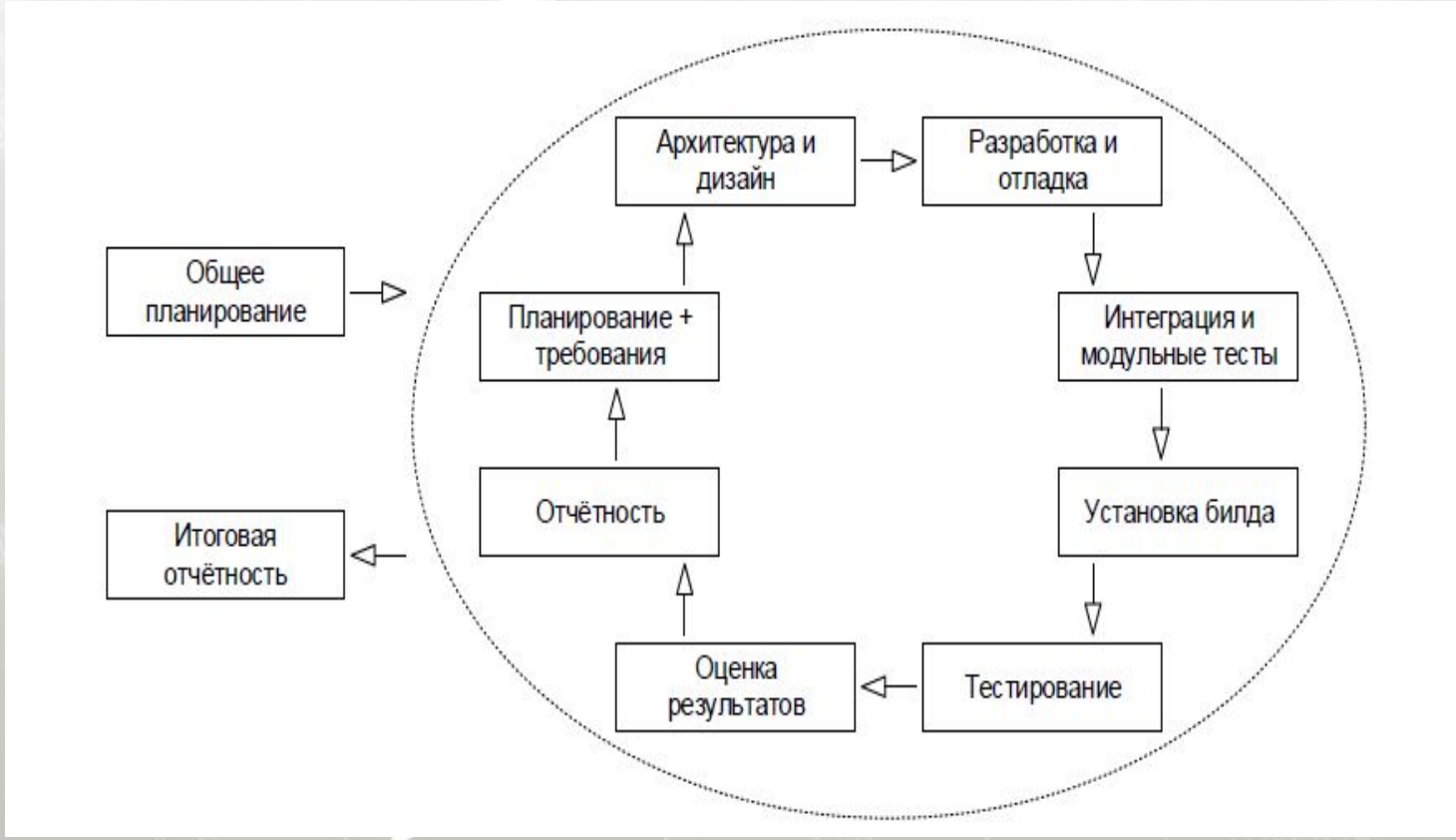
Недостатки инкрементальной модели:

- Требуется хорошее планирование и дизайн
- Требуется четкое и полное определение всей системы, прежде чем она может быть разбита на части и построена инкрементально
- Общая стоимость выше, чем водопадной модели.

When to use the Incremental model?

- Данная модель может быть использована, когда требования для всей системы четко определены и понятны
- Должны быть определены основные требования; Тем не менее, некоторые детали могут вовлекаться позже
- Существует необходимость раннего выхода продукта на рынок
- Новые технологии используются
- Ресурсы с требуемым набором навыков не доступны
- Есть некоторые рисковые функции и цели

Iterative - Incremental model



Iterative - Incremental model

Итерационная инкрементальная модель (***iterative model, incremental model***) является фундаментальной основой современного подхода к разработке ПО.
Ей свойственна определённая двойственность:
- с точки зрения жизненного цикла модель является **итерационной**, т.к. подразумевает многократное повторение одних и тех же стадий;
- с точки зрения развития продукта (приращения его полезных функций) модель является **инкрементальной** (наращиваемой).

Spiral model

Спиральная модель (***spiral model***) представляет собой частный случай итерационной инкрементальной модели, в котором особое внимание уделяется управлению рисками, в особенности влияющими на организацию процесса разработки проекта и контрольные точки.

Ключевые фазы:

- проработка целей, альтернатив и ограничений (**Planning**);
- анализ рисков и прототипирование (**Risk Analysis**);
- разработка (промежуточной версии) продукта (**Engineering**);
- оценка результатов и планирование следующего цикла (**Evaluation**).

Spiral model

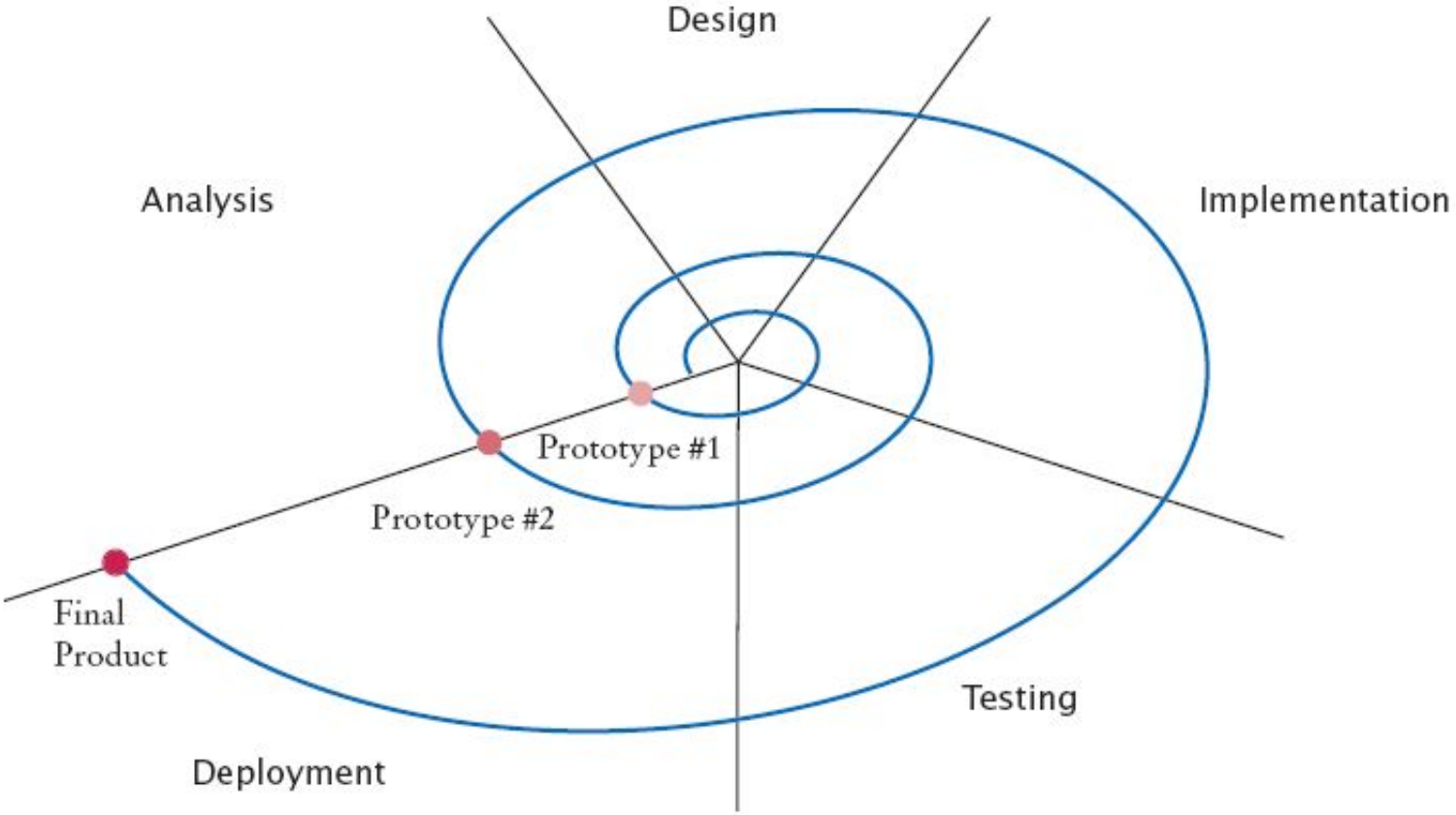


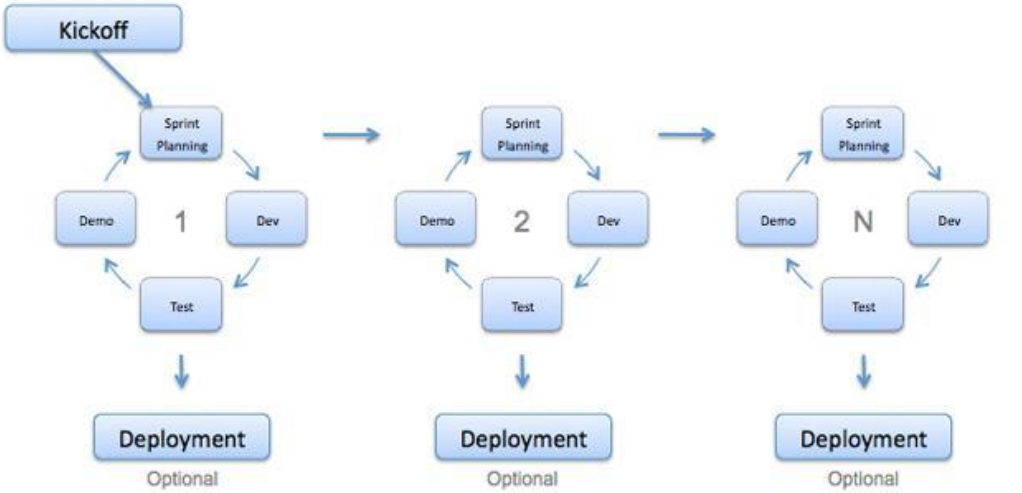
Figure 2 A Spiral Model

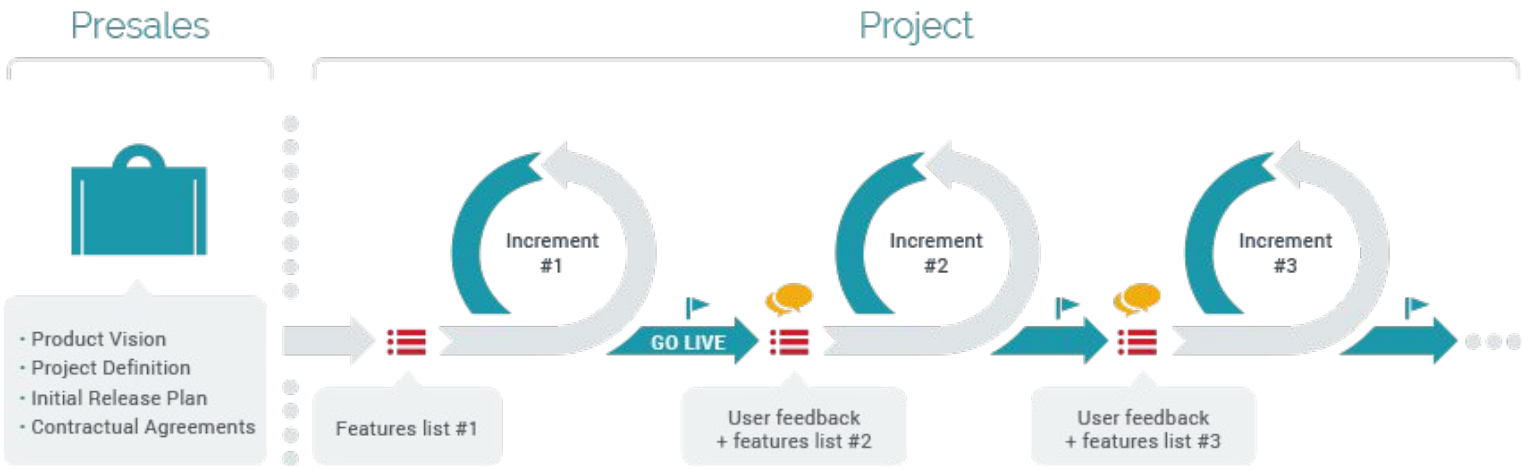
Agile model

Гибкая модель (***Agile model***) - представляет собой совокупность различных подходов к разработке ПО Базируется на т.н. «agile-манифесте»:

- Люди и взаимодействие важнее процессов и инструментов.
- Работающий продукт важнее исчерпывающей документации.
- Сотрудничество с заказчиком важнее согласования условий контракта.
- Готовность к изменениям важнее следования первоначальному плану.

Diagram of Agile model





Advantages of Agile model

Удовлетворение заказчика через быструю и непрерывную поставку жизнеспособного ПО.

Люди и взаимодействие важнее процессов и инструментов. Заказчик, разработчики и тестировщики постоянно взаимодействуют друг с другом.

Работающее ПО поставляется часто (неделями, а не месяцами).

Общение Face-to-face (лицом к лицу) является лучшей формой общения (communication is key to success).

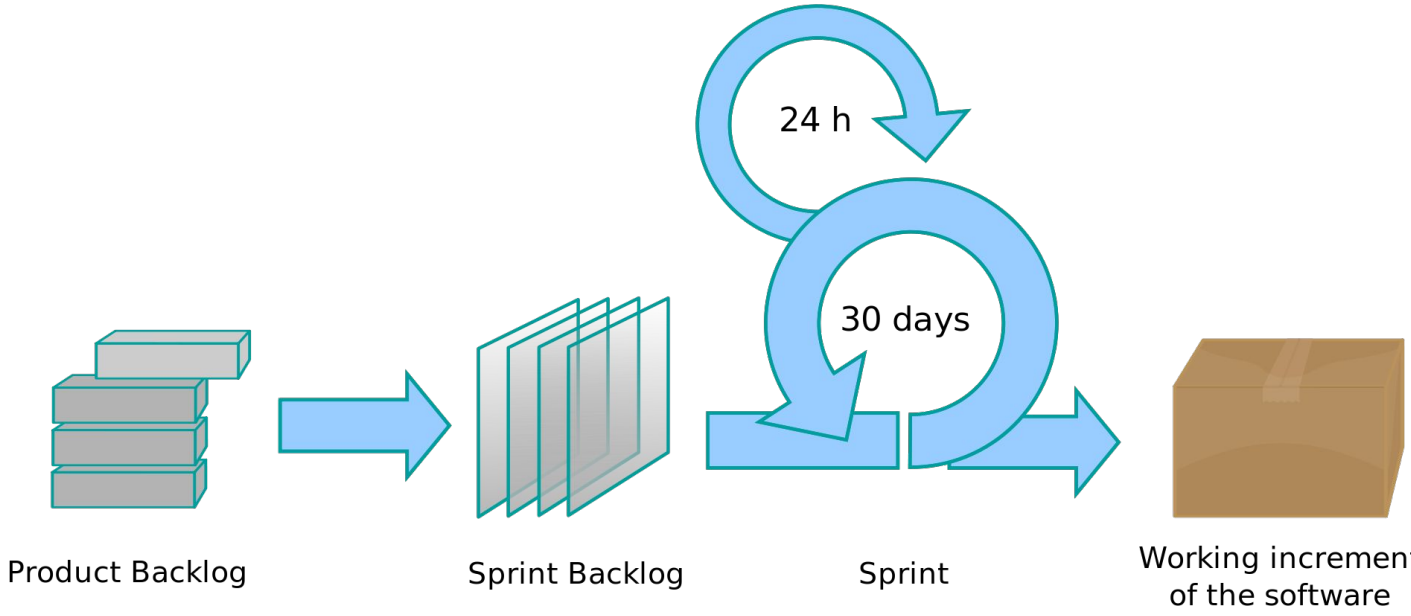
Тесное ежедневное сотрудничесво между бизнес аналитиками и программистами.

Постоянное внимание к техническому совершенству и хорошему дизайну.

Регулярная адаптация к изменяющимся обстоятельствам.

Даже поздние изменения в требованиях приветствуются

Scrum


24 h
30 days
Product Backlog
Sprint Backlog
Sprint
Working increment
of the software

Basic concepts of Scrum

Роли

Scrum Master

Product Owner

Team

Артефакты

Product Backlog - - это приоритезированный список имеющихся на данный момент бизнес-требований и технических требований к системе.

Включает в себя use cases, defects, enhancements(улучшения), technologies, stories, features, issues, и т.д. Product backlog также включает задачи, важные для команды, например "провести тренинг", "добить всем памяти"...

Basic concepts of Scrum

Sprint Backlog содержит функциональность, выбранную Product Owner из Product Backlog. Все функции разбиты по задачам, каждая из которых оценивается командой. Каждый день команда оценивает объем работы, который нужно проделать для завершения задач.
Сумма оценок оставшейся работы может быть построена как график зависимости от времени. Такой график называется **Sprint Burndown chart**. Он демонстрирует прогресс команды по ходу спринта.

Спринт (Sprint) - итерация в Scrum. Ее длительность составляет 1 месяц (30 дней). Результатом Sprint является готовый продукт (**build**), который можно передавать (**deliver**) заказчику (по крайней мере, система должна быть готова к показу заказчику).

Basic concepts of Scrum

Жизненный цикл спринта

1. **Планирование спринта:**

Митинг первый

Цель: Определить цель спринта (**Sprint Goal**) и **Sprint Backlog** - функциональность, которая будет разработана в течение следующего спринта для достижения цели спринта.

Митинг второй

Цель: определить, как именно будет разрабатываться определенная функциональность для того, чтобы достичь цели спринта. Для каждого элемента Sprint Backlog определяется список задач и оценивается их продолжительность. В Sprint Backlog появляются задачи

Basic concepts of Scrum

2. Если в ходе спринта выясняется, что команда не может успеть сделать запланированное на спринт, то Скрам Мастер, Product Owner и команда встречаются и выясняют, как можно сократить **scope** работ и при этом достичь цели спринта.

3. Остановка спринта (Sprint Abnormal Termination) - если команда понимает, что не может достичь цели спринта в отведенное время или если необходимость в достижении цели спринта исчезла.

4. Daily Scrum Meeting - проходит каждое утро в начале дня. Цель митинга - поделиться информацией.

5. Демо и ревью спринта - команда демонстрирует инкремент продукта, созданный за последний спринт.

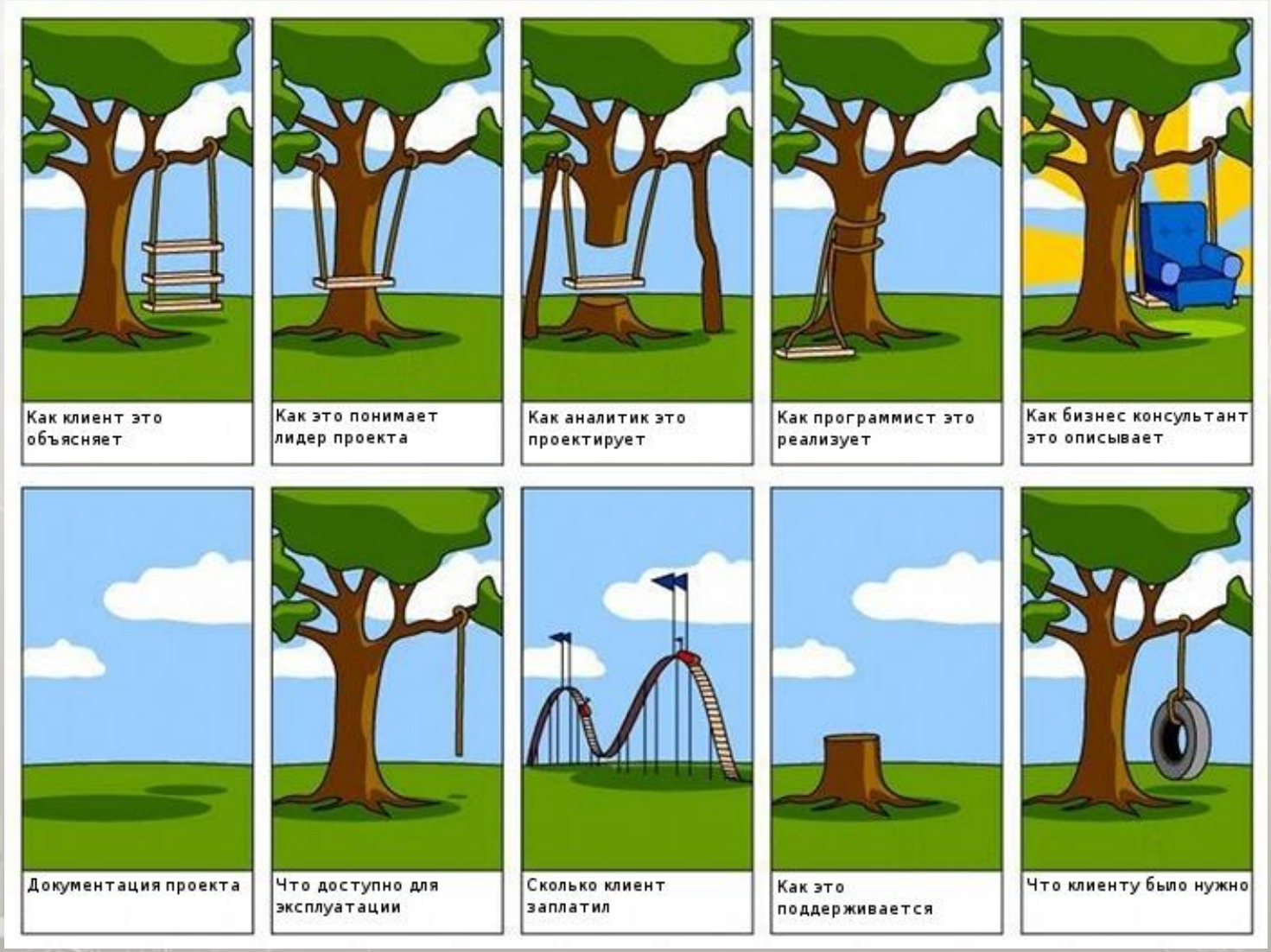
Summary table

Модель	Преимущества	Недостатки	Тестирование
Водопадная	• У каждой стадии есть чёткий проверяемый результат. • В каждый момент времени команда выполняет один вид работы. • Хорошо работает для небольших задач.	• Полная неспособность адаптировать проект к изменениям в требованиях. • Крайне позднее создание работающего продукта.	• С середины проекта.
V-образная	• У каждой стадии есть чёткий проверяемый результат. • Внимание тестированию уделяется с первой же стадии. • Хорошо работает для проектов со стабильными требованиями.	• Недостаточная гибкость и адаптируемость. • Отсутствует раннее прототипирование. • Сложность устранения проблем, пропущенных на ранних стадиях развития проекта.	• На переходах между стадиями.

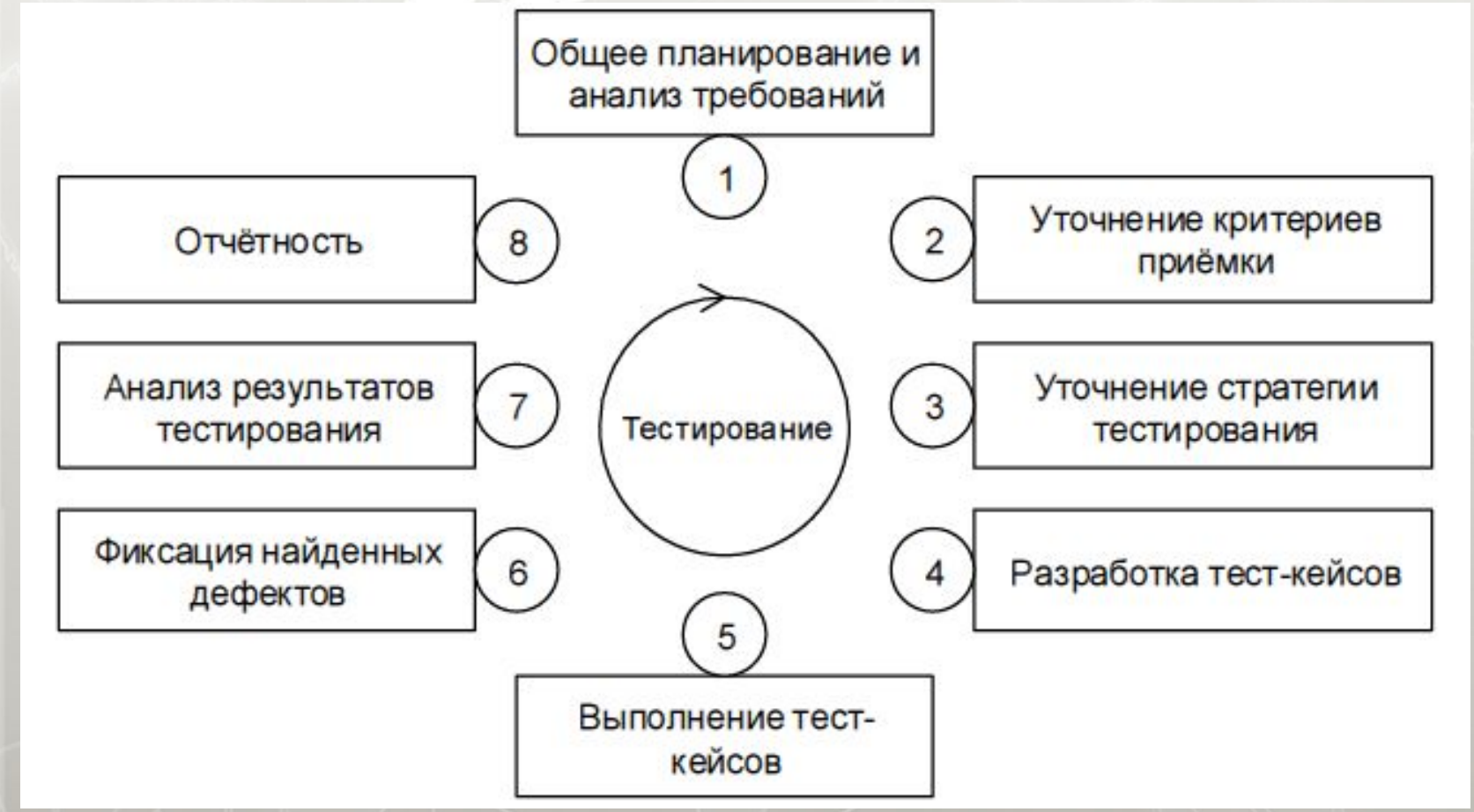
Summary table

Итерационная инкрементальная	• Достаточно раннее прототипирование. • Простота управления итерациями. • Декомпозиция проекта на управляемые итерации.	• Недостаточная гибкость внутри итераций. • Сложность устранения проблем, пропущенных на ранних стадиях развития проекта.	• В определённые моменты итераций. • Повторное тестирование (после доработки) уже проверенного ранее.
Спиральная	• Глубокий анализ рисков. • Подходит для крупных проектов. • Достаточно раннее прототипирование.	• Высокие накладные расходы. • Сложность применения для небольших проектов. • Высокая зависимость успеха от качества анализа рисков.	
Гибкая	• Максимальное вовлечение заказчика. • Много работы с требованиями. • Тесная интеграция тестирования и разработки. • Минимизация документации.	• Сложность реализации для больших проектов. • Сложность построения стабильных процессов.	• В определённые моменты итераций и **в любой необходимый момент**.

In conclusion...



Testing Life Cycle



Testing Life Cycle

Стадия 1 (общее планирование и анализ требований) - что нам предстоит тестировать; как много будет работы; какие есть сложности; всё ли необходимое у нас есть и т.п.
Стадия 2 (уточнение критериев приёмки) - формулирование или уточнение метрик и признаков возможности или необходимости начала тестирования (***entry criteria***), приостановки (***suspension criteria***) и возобновления (***resumption criteria***) тестирования, завершения или прекращения тестирования (***exit criteria***).
Стадия 3 (уточнение стратегии тестирования) - рассмотрение и уточнение тех части стратегии тестирования (***test strategy***), которые актуальны для текущей итерации.
Стадия 4 (разработка тест-кейсов) - разработка, пересмотр, уточнение, доработка, переработка и прочие действиям с тест-кейсами.

Testing Life Cycle

Стадия 5 (выполнение тест-кейсов) и **стадия 6** (фиксация найденных дефектов) - выполняются параллельно: дефекты фиксируются сразу по факту их обнаружения в процессе выполнения тест-кейсов.

Стадия 7 (анализ результатов тестирования) и **стадия 8** (отчётность) - выполняются параллельно. Формулируемые на стадии анализа результатов выводы напрямую зависят от плана тестирования, критериев приёмки и уточнённой стратегии, полученных на стадиях 1, 2 и 3.

Полученные выводы оформляются на стадии 8 и служат основной для стадий 1, 2 и 3 следующей итерации тестирования.

Таким образом, цикл замыкается.