Лекция 2

Качество программного обеспечения – это степень, в которой программное обеспечение обладает требуемой комбинацией свойств.

– это совокупность характеристик программного обеспечения, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Обеспечение качества (Quality Assurance – QA) – это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации программного обеспечения (ПО) информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО для обеспечения требуемого уровня качества выпускаемого продукта.

Контроль качества (Quality Control – QC) – это совокупность действий, проводимых над продуктом в процессе разработки для получения информации о его актуальном состоянии в разрезах: "готовность продукта к выпуску", "соответствие зафиксированным требованиям", "соответствие заявленному уровню качества продукта".

QA, QC и тестирование

Quality Assurance	Quality Control	Тестирование
Комплекс мероприятий, который охватывает все технологические аспекты на всех этапах разработки, выпуска и введения в эксплуатацию программных систем для обеспечения необходимого уровня качества программного продукта	Процесс контроля соответствия разрабатываемой системы предъявляемым к ней требованиям	Процесс, отвечающий непосредственно за составление и прохождение тест-кейсов, нахождение и локализацию дефектов и т.д.
Фокус в большей степени на процессы и средства, чем на непосредственно исполнение тестирования системы	Фокус на исполнение тестирования путем выполнения программы с целью определения дефектов с использованием утвержденных процессов и средств	Фокус на исполнение тестирования как такового
Процессно-ориентированный подход	Продуктно-ориентированный подход	Продуктно-ориентированный подход
Превентивные меры	Корректирующий процесс	Превентивный процесс
Подмножество процессов Software Test Life Cycle – цикла тестирования ПО	Подмножество процессов QA	Подмножество процессов QC

Тестирование – часть QC. QC – часть QA.

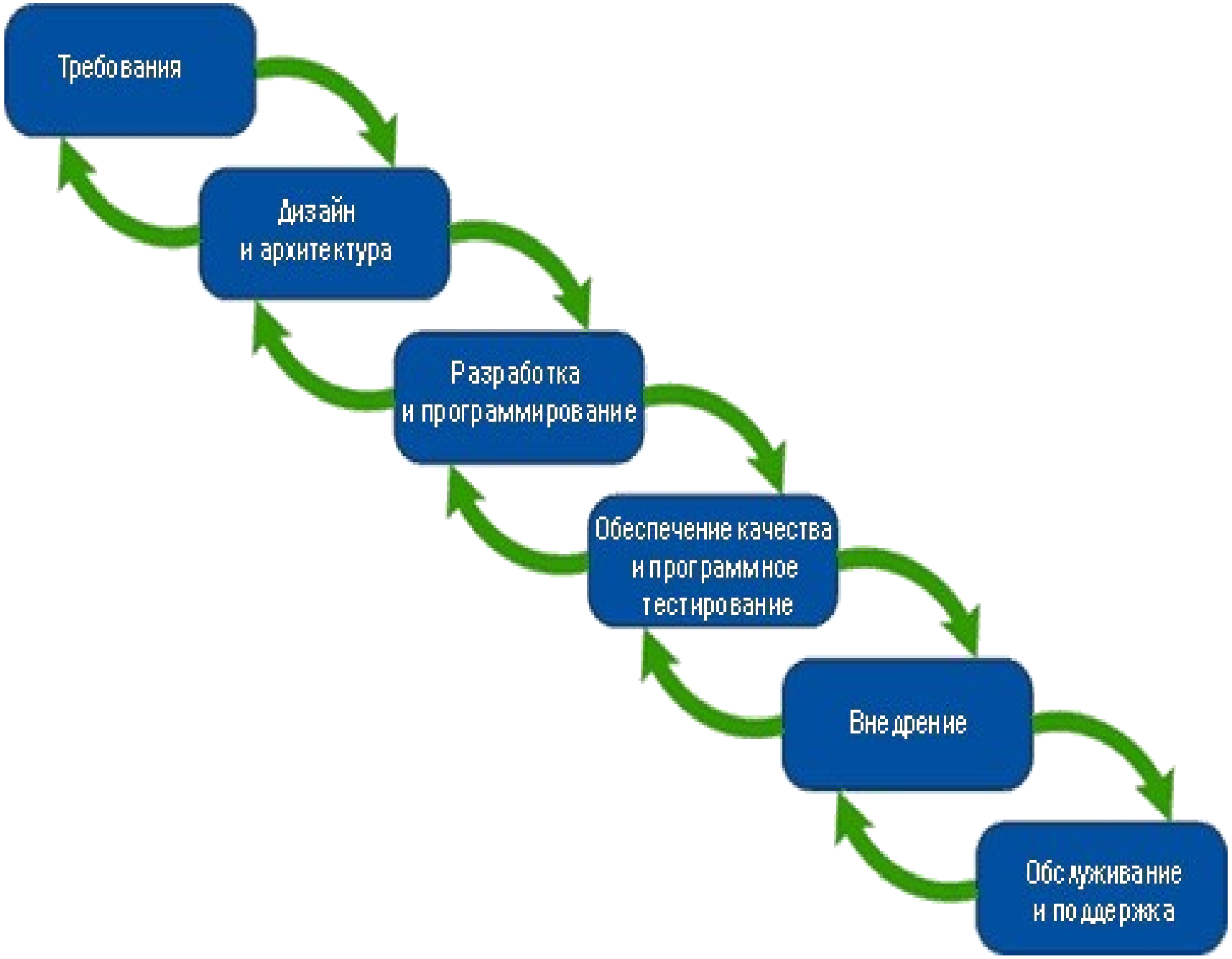
Quality Assurance обеспечивает правильность и предсказуемость процесса, в то время как Quality Control предполагает контроль соблюдения требований. Тестирование же, в свою очередь, обеспечивает сбор статистических данных и внесение их в документы, созданные в рамках QC-процесса.

Жизненный цикл ПО (цикл разработки)

– это условная схема, включающая отдельные этапы, которые представляют стадии процесса создания ПО. При этом, на каждом этапе выполняются разные действия.

Стадии и этапы разработки программного обеспечения определены в Единой системе программной документации (ЕСПД) ГОСТом 19.102-77.

Основная разработка программного обеспечения осуществляется на этапе формирования рабочего проекта. Последовательность и/или одновременность выполнения процессов зависит от применяемой технологии разработки программного обеспечения.


Требования
Дизайн
и архитектура
Разработка
и программирование
Обеспечение качества
и программное
тестирование
Внедрение
Обслуживание
и поддержка

Анализ требований

Цель этой стадии – определение детальных требований к системе. Кроме этого, необходимо убедиться в том, что все участники правильно поняли поставленные задачи и то, как именно каждое требование будет реализовано на практике.

Этот этап предполагает сбор требований к разрабатываемому программному обеспечению, их систематизацию, документирование, анализ, а также выявление и разрешение противоречий.

Проектирование (стадия дизайна и архитектуры)

Программисты и системные архитекторы, руководствуясь требованиями, разрабатывают высокоуровневый дизайн системы.

Дизайн, как правило, закрепляется отдельным документом – дизайн-спецификацией (DSD – Design Specification Document).

Для упрощения визуализации процесса проектирования, используются так называемые нотации– схематическое выражение характеристик системы.

Основные используемые нотации:

- Блок-схемы.
- ER-диаграммы.
- UML-диаграммы.
- Макеты – например, нарисованный прототип сайта.

Разработка и программирование

Начинается написание программистами кода программы в соответствии с ранее определенными требованиями.

Программирование предполагает четыре основных стадии:

- Разработка алгоритмов – создание логики работы программы.
- Написание исходного кода.
- Компиляция – преобразование в машинный код.
- Тестирование и отладка – юнит-тестирование.

Документация

Существует четыре уровня документации:

- Архитектурная (проектная) – например, дизайн-спецификация. Это документы, описывающие модели, методологии, инструменты и средства разработки, выбранные для данного проекта.
- Техническая – вся сопровождающая разработку документация (различные документы, поясняющие работу системы на уровне отдельных модулей).
- Пользовательская – включает справочные и поясняющие материалы, необходимые конечному пользователю для работы с системой (Readme, раздел справки по программе и т.п.).
- Маркетинговая – включает рекламные материалы, сопровождающие выпуск продукта.

Тестирование

- Планирование и управление - действия, направленные на определение основных целей тестирования и задач, выполнение которых необходимо для достижения этих целей; составление тест-стратегии, тест-плана.
- Анализ и проектирование - написание тестовых сценариев и условий на основе общих целей тестирования.
- Внедрение и реализация - написание тест-кейсов, на основе написанных ранее тестовых сценариев, собирается необходимая для проведения тестов информация, подготавливается тестовое окружение и запускаются тесты.
- Оценка критериев выхода и написание отчетов - проверяется было ли проведено достаточное количество тестов, достигнута ли нужная степень обеспечения качества системы.

Основные цели этапа тестирования

- убедиться, что вся запланированная функциональность действительно была реализована;
- проверить, что все отчеты об ошибках, поданные ранее, были, так или иначе, закрыты;
- завершение работы тестового обеспечения, тестового окружения и инфраструктуры;
- оценить общие результаты тестирования и проанализировать опыт, полученный в его процессе.

Внедрение и сопровождение

Когда программа протестирована и в ней больше не осталось серьезных дефектов, приходит время релиза и передачи ее конечным пользователям.

В случае обнаружения пользователями тех или иных пост-релизных багов, информация о них передается в виде отчетов об ошибках команде разработки, которая, в зависимости от серьезности проблемы, либо немедленно выпускает исправление (hot-fix), либо откладывает его до следующей версии программы.

Модели разработки ПО

- Каскадная (водопадная)
- V-образная
- Итерационная
- Инкрементальная
- Спиральная
- Гибкая

Каскадная модель

- высокий уровень формализации процессов;
- большое количество документации;
- жесткая последовательность этапов жизненного цикла без возможности возврата на предыдущий этап.

Минусы:

- Waterfall-проект должен постоянно иметь актуальную документацию. Обязательная актуализация проектной документации. Избыточная документация.
- Очень не гибкая методология.
- У заказчика нет возможности ознакомиться с системой заранее и даже с «Пилотом» системы.
- У пользователя нет возможности привыкать к продукту постепенно.
- Все требования должны быть известны в начале жизненного цикла проекта.
- Возникает необходимость в жёстком управлении и регулярном контроле, иначе проект быстро выбьется из графиков.
- Отсутствует возможность учесть переделку, весь проект делается за один раз.

Плюсы:

- Высокая прозрачность разработки и фаз проекта.
- Чёткая последовательность.
- Стабильность требований.
- Строгий контроль менеджмента проекта.
- Облегчает работу по составлению плана проекта и сбора команды проекта.
- Хорошо определяет процедуру по контролю качества.

V модель — разработка через тестирование

Данная модель имеет более приближенный к современным методам алгоритм, однако все еще имеет ряд недостатков. Является одной из основных практик экстремального программирования и предполагает регулярное тестирование продукта во время разработки.

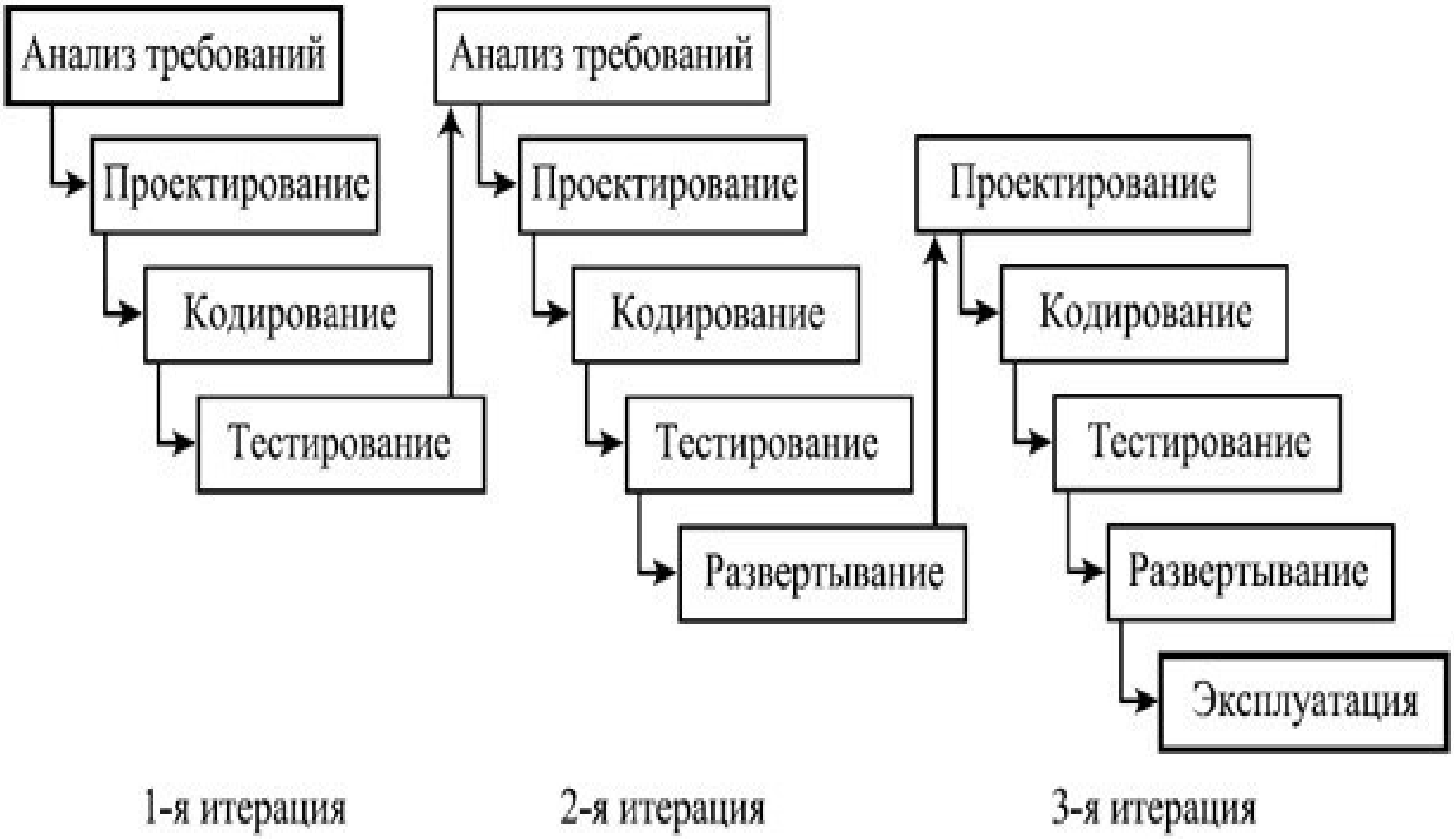

Планирование проектов и требований
Производство, эксплуатация и сопровождение
Анализ требований продукта и спецификаций
Системное и приемочное тестирование
Разработка архитектурного проекта на высшем уровне
Интеграция и тестирование
Детализированная разработка проекта
Модульное тестирование
Кодирование

Обеспечивает поддержку в планировании и реализации проекта. В ходе проекта ставятся следующие задачи:

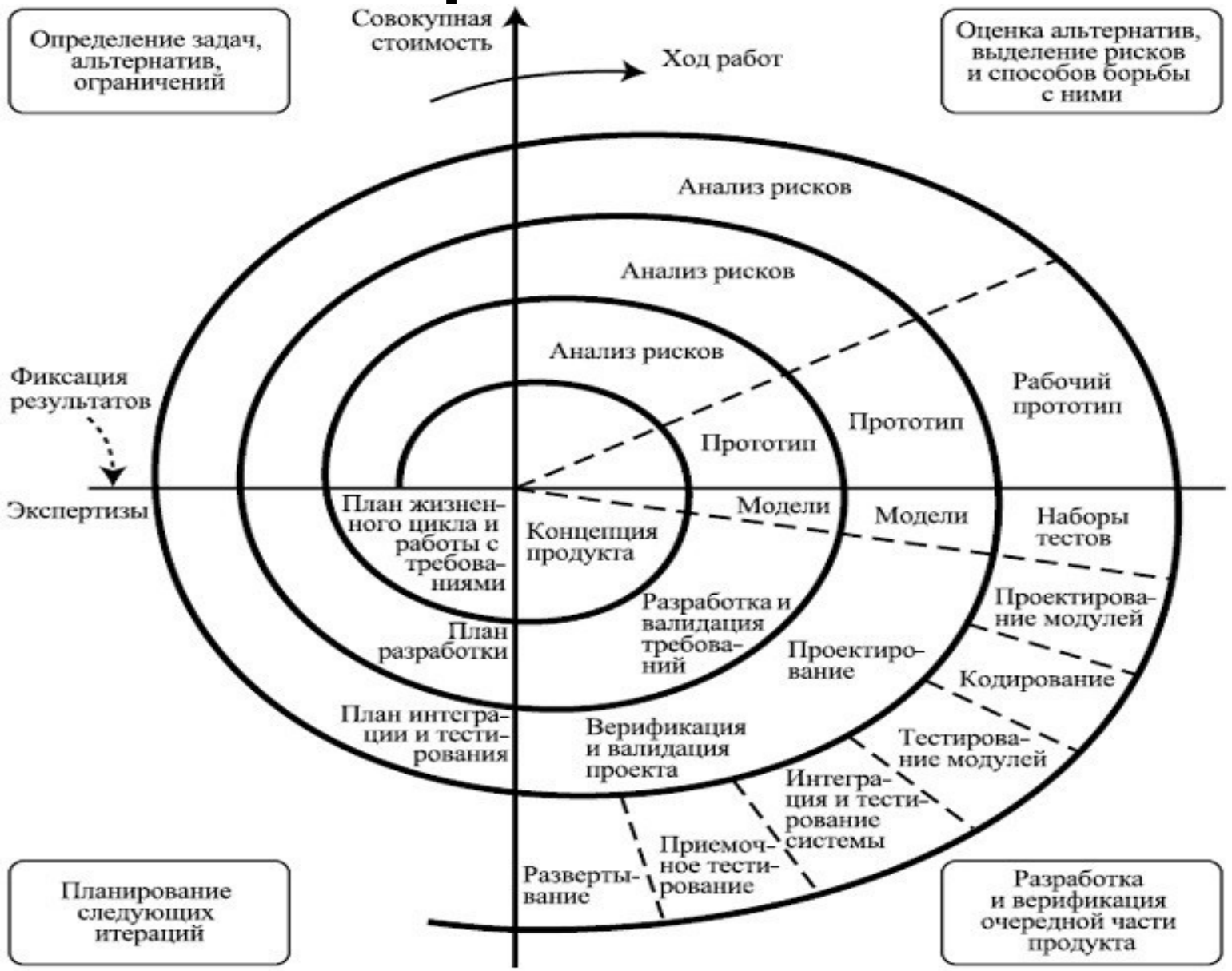
- **Минимизация рисков:** делает проект более прозрачным и повышает качество контроля проекта путём стандартизации промежуточных целей и описания соответствующих им результатов и ответственных лиц.
- **Повышение и гарантии качества:** Промежуточные результаты могут быть проверены на ранних стадиях. Универсальное документирование облегчает читаемость, понятность и проверяемость.
- **Уменьшение общей стоимости проекта:** ресурсы на разработку, производство, управление и поддержку могут быть заранее просчитаны и проконтролированы.
- **Повышение качества коммуникации между участниками проекта:** универсальное описание всех элементов и условий облегчает взаимопонимание всех участников проекта.

Итеративная модель

Итеративные или инкрементальные модели предполагают разбиение создаваемой системы на набор кусков, которые разрабатываются с помощью нескольких последовательных проходов всех работ или их части.


Анализ требований
Проектирование
Кодирование
Тестирование
Анализ требований
Проектирование
Кодирование
Тестирование
Развертывание
Проектирование
Кодирование
Тестирование
Развертывание
Эксплуатация
1-я итерация
2-я итерация
3-я итерация

Спиральная модель



Преимущества модели:

- Результат достигается в кратчайшие сроки.
- Конкурентоспособность достаточно высокая.
- При изменении требований не придется начинать все с «нуля».

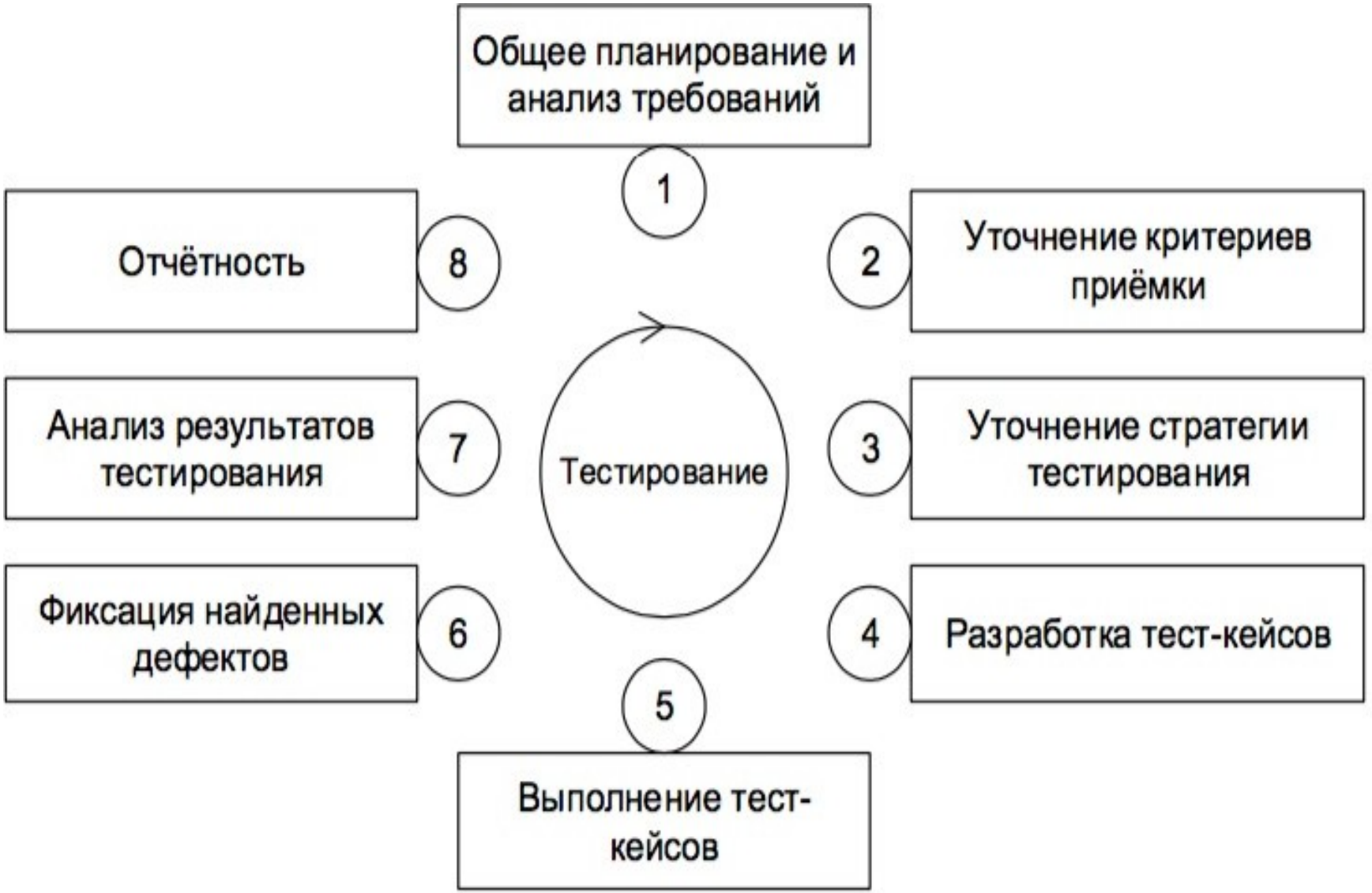
Но у этой модели есть один существенный **недостаток:**

- **невозможность регламентирования стадий выполнения.**

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.

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.	

Жизненный цикл тестирования



- **Стадия 1** (общее планирование и анализ требований) объективно необходима, как минимум, для того, чтобы иметь ответ на такие вопросы, как: что нам предстоит тестировать; как много будет работы; какие есть сложности; всё ли необходимое у нас есть и т.п. Как правило, получить ответы на эти вопросы невозможно без анализа требований, т.к. именно требования являются первичным источником ответов.

- **Стадия 2** (уточнение критериев приёмки) позволяет сформулировать или уточнить метрики и признаки возможности или необходимости начала тестирования, приостановки и возобновления тестирования, завершения или прекращения тестирования.
- **Стадия 3** (уточнение стратегии тестирования) представляет собой ещё одно обращение к планированию, но уже на локальном уровне: рассматриваются и уточняются те части стратегии тестирования, которые актуальны для текущей итерации.

- **Стадия 4** (разработка тест-кейсов) посвящена разработке, пересмотру, уточнению, доработке, переработке и прочим действиям с тест-кейсами, наборами тест-кейсов, тестовыми сценариями и иными артефактами, которые будут использоваться при непосредственном выполнении тестирования.

- **Стадия 5** (выполнение тест-кейсов) и **стадия 6** (фиксация найденных дефектов) тесно связаны между собой и фактически выполняются параллельно: дефекты фиксируются сразу по факту их обнаружения в процессе выполнения тест-кейсов. Однако зачастую после выполнения всех тест-кейсов и написания всех отчетов о найденных дефектах проводится явно выделенная стадия уточнения, на которой все отчеты о дефектах рассматриваются повторно с целью формирования единого понимания проблемы и уточнения таких характеристик дефекта, как важность и срочность.

- **Стадия 7** (анализ результатов тестирования) и **стадия 8** (отчетность) также тесно связаны между собой и выполняются практически параллельно. Формулируемые на стадии анализа результатов выводы напрямую зависят от плана тестирования, критериев приёмки и уточненной стратегии, полученных на стадиях 1, 2 и 3. Полученные выводы оформляются на стадии 8 и служат основой для стадий 1, 2 и 3 следующей итерации тестирования. Таким образом, цикл замыкается.