Программа курса “Введение в тестирование ПО” (20 часов)

Октябрь - Ноябрь, 2017

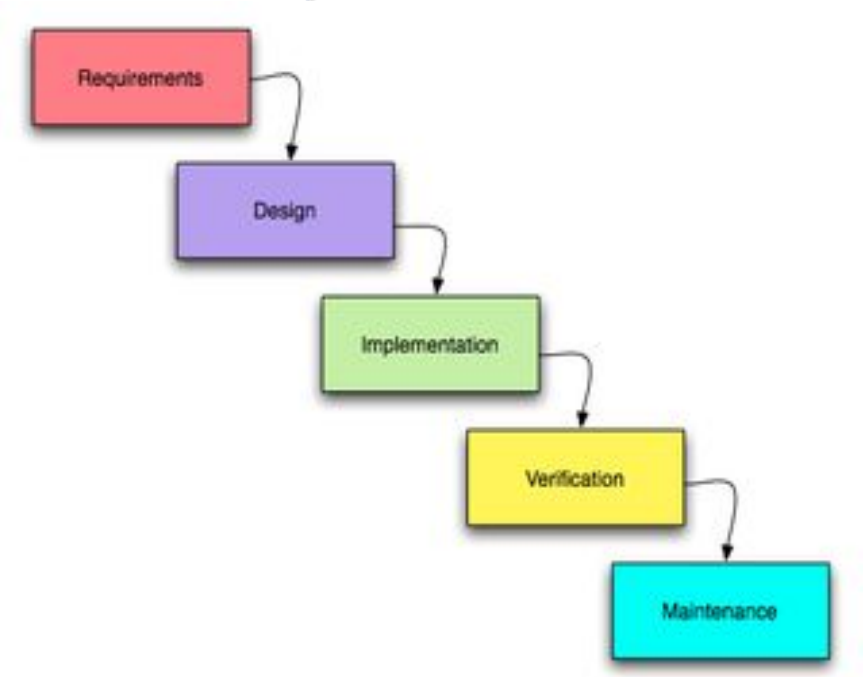
2. Тестирование в жизненном цикле ПО

- Модели разработки ПО

- **Жизненный цикл программного обеспечения** — ряд событий, происходящих с системой в процессе ее создания и дальнейшего использования. Говоря другими словами, это время от начального момента создания какого либо программного продукта, до конца его разработки и внедрения. Жизненный цикл программного обеспечения можно представить в виде моделей.

- Модели разработки ПО

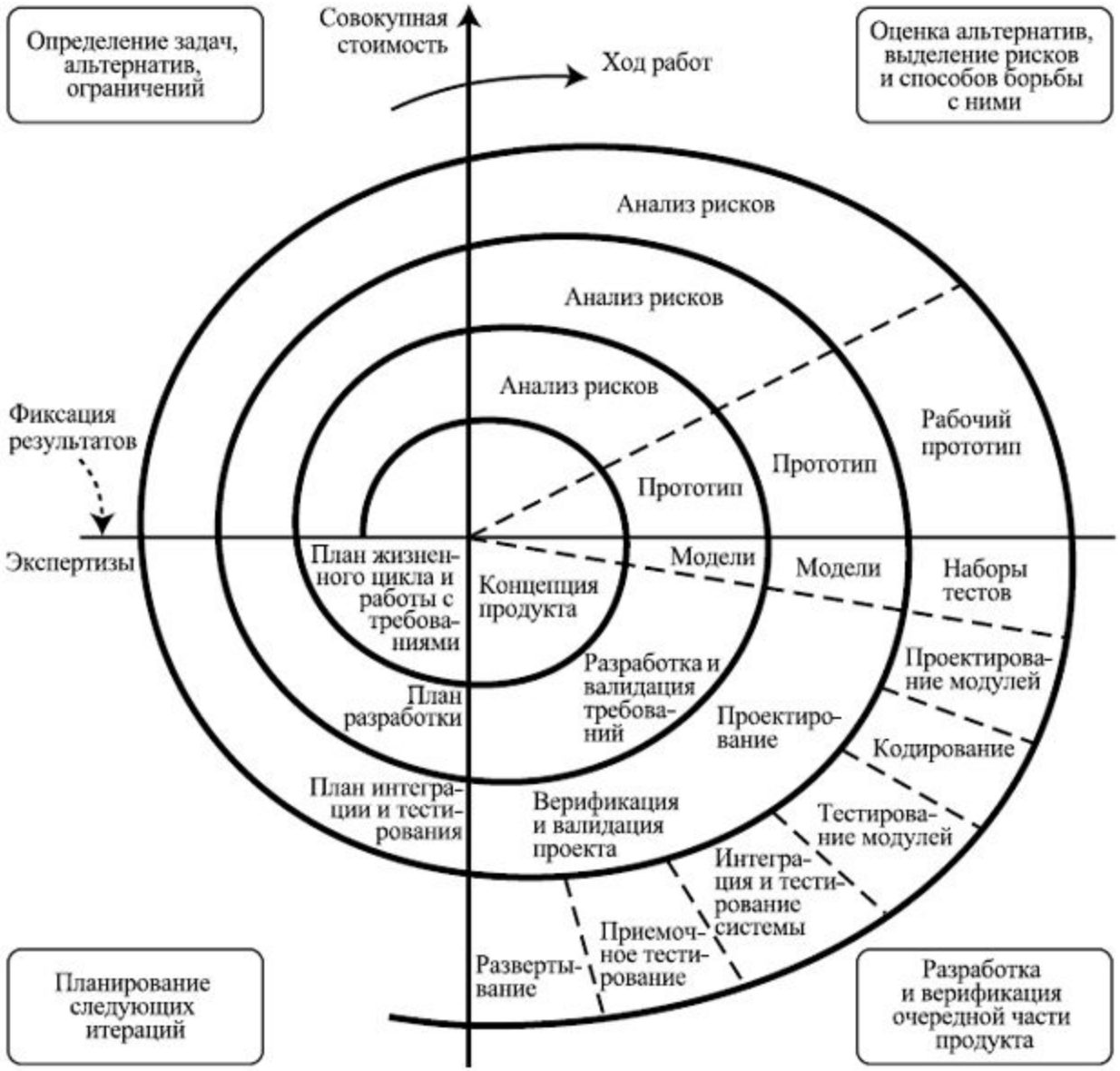
Каскадная разработка или модель водопада (англ. waterfall model) — модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.



- Модели разработки ПО

Спиральная модель, представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции, делающая упор на начальные этапы жизненного цикла: анализ и проектирование. Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла.

- Модели разработки ПО



- Модели разработки ПО

Итеративная разработка (англ. iteration — повторение) — выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы. Проект при этом подходе в каждой фазе развития проходит повторяющийся цикл: Планирование — Реализация — Проверка — Оценка (англ. plan-do-check-act cycle).
В ходе разработки всегда выявляются дополнительные требования или изменяются выявленные ранее. Также появляются новые ограничения, связанные с принятыми техническими решениями. В наиболее полной мере их удается учесть именно в итерационной разработке, поскольку именно при таком подходе руководство проекта в полной мере готово к изменениям.

- Виды требований

Требования к программному обеспечению — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению, в результате анализа требований.

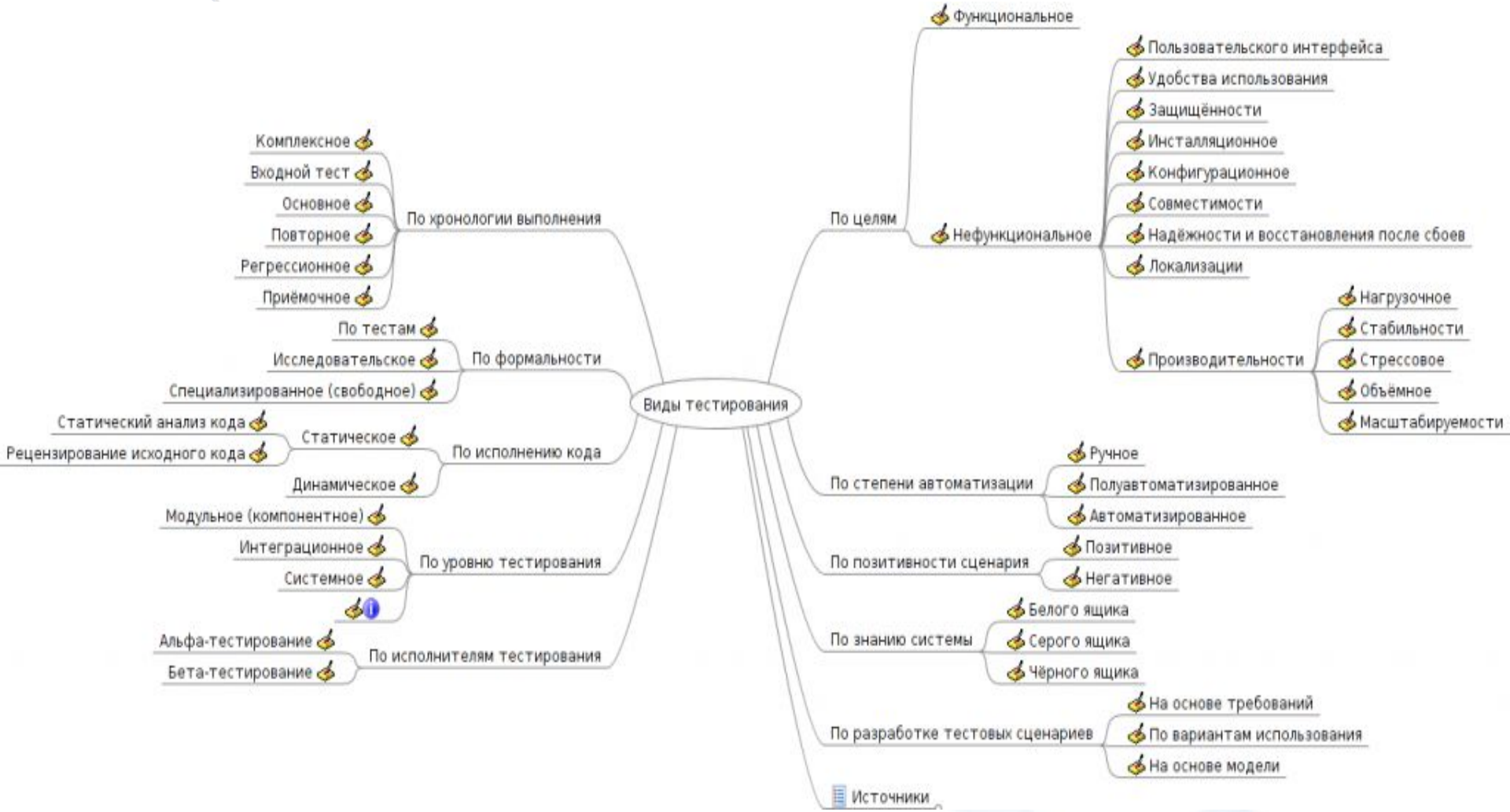
Функциональные требования объясняют, что должно быть сделано. Они идентифицируют задачи или действия, которые должны быть выполнены. Функциональные требования определяют действия, которые система должна быть способной выполнить, связь входа/выхода в поведении системы.

- Виды требований

Нефункциональные требования — требования, которые определяют критерии работы системы в целом, а не отдельные сценарии поведения. Нефункциональные требования определяют системные свойства такие как производительность, удобство сопровождения, расширяемость, надежность, средовые факторы эксплуатации.

Требования к атрибутам качества
Требования к применяемому оборудованию и ПО
Требования к документированию
Требования к дизайну и удобству использования
Требования к безопасности и надёжности
Требования к показателям назначения (производительность, устойчивость к сбоям и т. п.)
Прочие требования и ограничения (внешние воздействия, мобильность, автономность и т. п.).

- Виды тестирования



- Уровни тестирования

Компонентное (модульное) тестирование

проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (**модули программ, объекты, классы, функции и т.д.**).
Обычно компонентное (модульное) тестирование проводится вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки (frameworks - каркасы) для модульного тестирования или инструменты для отладки. Все найденные **дефекты**, как правило исправляются в коде без формального их описания в системе менеджмента багов (Bug Tracking System).
Один из наиболее эффективных подходов к компонентному (модульному) тестированию - это **подготовка автоматизированных тестов** до начала основного кодирования (разработки) программного обеспечения. Это называется разработка от тестирования (**test-driven development**) или подход тестирования вначале (**test first approach**). При этом подходе создаются и интегрируются небольшие куски кода, напротив которых запускаются тесты, написанные до начала кодирования. Разработка ведется до тех пор пока все тесты не будут успешно пройдены.

- Уровни тестирования

Интеграционное тестирование (Integration Testing)
Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).

Уровни интеграционного тестирования:

Компонентный интеграционный уровень (*Component Integration testing*)
Проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.
Системный интеграционный уровень (*System Integration Testing*)Проверяется взаимодействие между разными системами после проведения системного тестирования.

- Уровни тестирования

Подходы к интеграционному тестированию:

Снизу вверх (*Bottom Up Integration*)
Все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования. Данный подход считается полезным, если все или практически все модули, разрабатываемого уровня, готовы. Также данный подход помогает определить по результатам тестирования уровень готовности приложения.

- Уровни тестирования

Сверху вниз (*Top Down Integration*)
Вначале тестируются все высокоуровневые модули, и постепенно один за другим добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем по мере готовности они заменяются реальными активными компонентами. Таким образом мы проводим тестирование сверху вниз.

Большой взрыв (*"Big Bang" Integration*)
Все или практически все разработанные модули собираются вместе в виде законченной системы или ее основной части, и затем проводится интеграционное тестирование. Такой подход очень хорош для сохранения времени. Однако если тест кейсы и их результаты записаны не верно, то сам процесс интеграции сильно осложнится, что станет преградой для команды тестирования при достижении основной цели интеграционного тестирования.

- Уровни тестирования

Системное тестирование (System Testing)
Основной задачей системного тестирования является **проверка как функциональных, так и не функциональных требований в системе в целом**. При этом выявляются **дефекты**, такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д. Для минимизации рисков, связанных с особенностями поведения системы в той или иной среде, **во время тестирования рекомендуется использовать окружение максимально приближенное к тому, на которое будет установлен продукт после выдачи**.

- Уровни тестирования

Можно выделить два подхода к системному тестированию:

на базе требований (*requirements based*)
Для каждого требования пишутся **тестовые случаи (test cases)**, проверяющие выполнение данного требования.
на базе случаев использования (*use case based*)
На основе представления о способах использования продукта создаются случаи использования системы (**Use Cases**). По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся **тест кейсы (test cases)**, которые должны быть протестированы.

- По исполнителям тестирования

Альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.

- По исполнителям тестирования

Бета-тестирование — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

В о п р о с ы ?