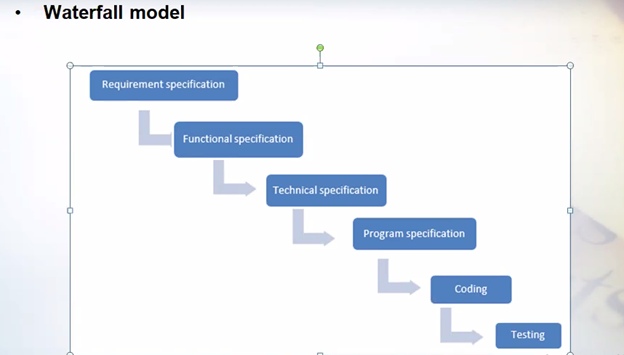
**waterfall model**



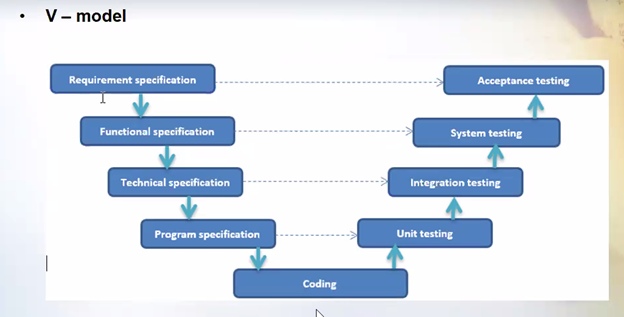
Requirement specification-определение требований пользователя и заказчика   
Functional specification-определение функциональности,необходимой для достижения требований   
Technical specification-определение технического дизайна и архитектуры   
Program specification -определение детального дизайна каждого модуля   
Coding-разработка   
Testing-тестирование

**Преимущества данной модели:**  
1)Понятна и проста в использовании   
2)Хорошо работает для небольших проектов   
3)Фазы разработки не пересекаются   
4)Хорошо срабатывает когда требования к качеству доминируют над требованиями к затратам и графику проекта   
  
**Недостатки данной модели:**  
1)Каждая попытка вернуться на 1 или 2 фазы назад чтобы исправить проблему приведет к увеличению затрат и сбою в графике   
2)Высокая степень появления рисков   
3)Не подходит для комплексных и больших проектов   
4)Рабочая версия продукта появляется только на последних этапах   
5)Клиент не может убедиться в качестве продукта до окончания процесса разработки   
  
**Когда используется данная модель:**  
1)Когда requirement-ы четкие и ясные   
2)Когда цели и определения продукта четко установлены и стабильны   
3)Для небольших проектов

**Область применения спиральной модели:**

1. При разработки новой функции или новой серии продукта
2. Когда требования слишком сложные
3. Для проекта со средними и высокими рисками
4. Для больших проектов

**V-model:**



**Unit testing –** выполняется проверка каждого модуля(кода) на наличие шибок

**Unit testing –** проверка взаимосвязи между группами ранееразработанных и протестированных модулей

**System testing –** выполняется проверка функционирования программы в целом

**Acceptance testing –** прием тестирования заказчиком

**Плюсы:**

1. кождому этапу разработки соответствует этап тестирования
2. особое значение придается тестированию
3. хорошо работает для небольших проектов с четкими требованиями

**Минусы:**

1)не входят в действие направленные на анализ рисков

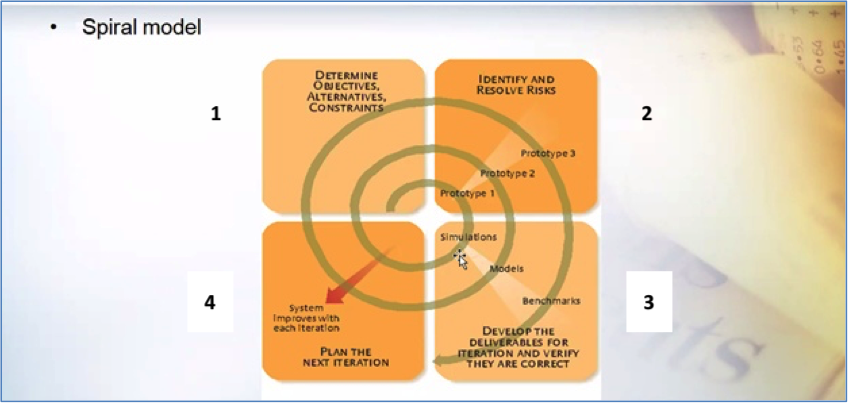
2)строгая последовательность этапов

3)готовая версия продукта появляется на последних стадиях

4)если появляются изменения в середине разработки, то необходимо вернутся к обновлению требований(в начале)

**Применение:** применяется для систем, в которых требуется высокая надежность

**Spiral model**



**Spiral model-** за счет спиралей(циклов) предоставляется версии продукта заранее

1. определение целей проекта
2. определение и разрешение рисков. Данный этап повторяется для каждой итерации(версии продукта). Определение технического дизайна итерации
3. кодинг и тестинг ПО
4. оценка результатов и определение правильности следующей итерации

каждый цикл представляет собой набор спиралей,6 которые соответствует такое же количество стадий как в каскадном модуле

каждый виток спирали соответствует созданию фрагмента или версии продукта

**Плюсы:**

1. анализ рисков
2. подходит для больших проектов и проектов с высокими рисками
3. контроль документации по проекту
4. дополнительный функционал может быть добавлен в последующих итерациях(витках)
5. модель разрешает клиенту и пользователю увидеть систему на ранних тапах
6. не нужно распределять заранее все финансовые ресурсы

**Недостатки:**

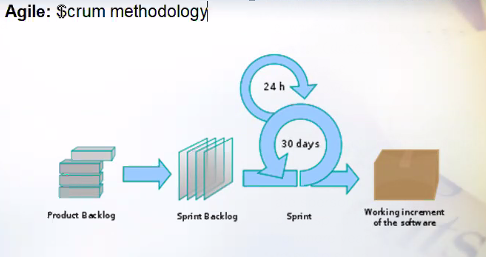
1)может оказаться дорогостоящем (на проекте с низкой степенью рисков или на небольшом проекте)(расчет рисков связан с затратами)

2)не работает для маленьких проектов

3)модуль имеет усложненную структуру

2 тема:

**Методология скрам**



Product Backlog- список требований к функциональности

Sprint Backlog- выбранные задания на спринт

Sprint

Working increment of the software

Методология Agile заключается в серии коротких циклов(итерации, спринты), которые обычно длятся 1-2 недели

Каждый день проводятся митинги, где члены команды обсуждают прогресс по выполненным заданиям

В команде присутствует скрам-мастер(scrum master) – руководитель, представитель заказчик. Он помогает команде правильно двигать задания и не забывать задачи, помогает определить приоритеты в заданиях.

Плюсы методологии Agile:

1. Рабочая версия продукта появляется в начальных стадиях
2. Ежедневное и тесное общение членов команды
3. Постоянное усовершенствование технического дизайна и требований
4. Адаптация к изменяющимся состояниям
5. Разрешаются поздние изменения в требованиях

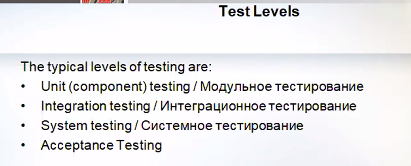
Недостатки :

1. Недостаточное внимание уделяется документации
2. Проект может быть отменен или закрыт, если клиент не удовлетворен продуктом

Когда используется Agile:

1. когда нужно внедрять изменения
2. На проектах с невысокими рисками
3. Когда на проекте достаточно минимального планирования
4. Изменения могут быть не дорогостоящими, благодаря частоте инкременции продукта( частоте спринотов)

**Уровни тестирования**

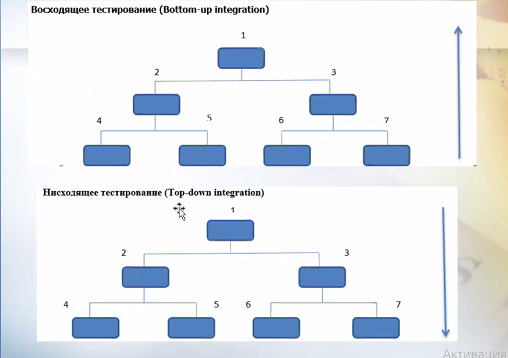
****

Модульное тестирование- его цель удостоверится в соответствии требований каждого отдельного модуля системы перед тем как они будут интегрированы друг в друга( обычно проводятся программистами, которые написали код) Чаще всего баги выявленные и починенные в это время не отмечаются в бак-трекере

Интеграционное тестирование проводится после модульного над компонентами которые уже были протестированы и как они теперь интегрируются вместе

**Типы интеграционного тестирования: (три подхода)**

1. Bottom-up (восходящее тестирование)
2. Top-down (нисходящее)
3. Big-bang



**Цель интеграционного тестирования –** проверить как объедененые модули взаимодействуют друг с другом

**Bottom-up (восходящее тестирование)** сперва собираются и тестируются низкоуровневые модули, а затем постепенно добавляются модули более высокого уровня. Для этого девелопер использует заглушки и драйвера на месте интеграции модулей. На создание драйвера может уйти большое количество времени.

**Top-down (нисходящее)** этот процес тестирования движется следом за разработкой. Сначала тестируют самый верхний( управляющий модуль), затем постепенно интергируются более низкоуровневые модули. В данном методе используются только заглушки.

**Big-bang** предполагается, что все компоненты интегрируются одновременно, не нужны ни драйвера ни заглушки, потому что все интегрируется и проверяется одновременно. И появляются недостатки:

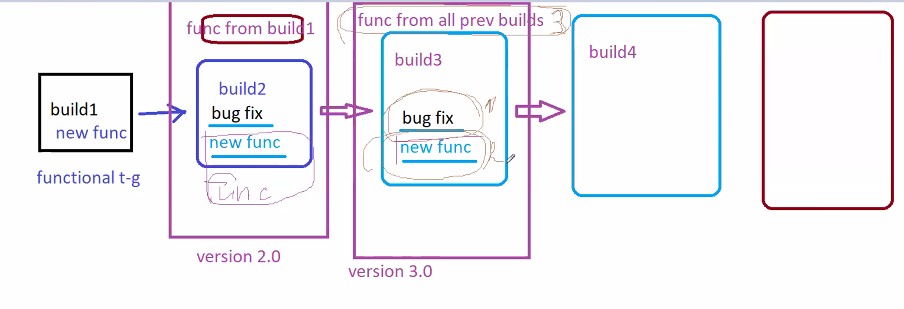
1. Трудно выявить источник ошибки(место соединения) в каком модуле проблема или в каком месте их соединения
2. Трудно организовать исправления ошибки, когда модули написаны разными людьми и сложнее найти ответственного за поиск и устранения дефекта

**System testing(системное тестирование)**

**Виды системного тестирования:**

* функциональное,
* регрессионное
* нефункциональное тестирование

Системного тестирования выполняется после интеграционного тестирования. На этом этапе проводится функциональное тестирование, оценка характеристик качества системы, его устойчивость, надежность, безопасность, производительность и тд. Системное тестирование проводится в несколько фаз, на каждой из которых проверяется один из аспектов ее поведения, то есть проводится один из типов системного тестирования. Все фазы могут протекать одновременно или последовательно. (Регрессионное тестирование всегда проходит после функционального/нефункционального тестирования. А функциональное может проходить отдельно или одновременно с нефункциональным)



**функциональное тестирование** – на этом этапе проверяются все функции системы, как система в целом ведет себя в соответствии с ожиданиями пользователь, в соотв с требованиями. На этом проверяется весь фукционал системы.

Критерии полноты тестирования:

1. все функциональные требования должны быть протестированы
2. все классы допустимых входных данных должны корректно обрабатываться системой
3. все классы недопустимых данных должны быть обработаны и отброшены системой
4. в тестах должны генерироваться все возможные классы входных данных
5. система должна побывать во всех своих состояниях(диаграмма состояний)

**Регрессионное тестирование** направлено на обнаружение ошибок в уже протестированных участках программ.

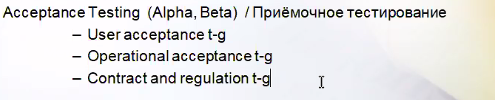
Включает в себя:

1. проверка исправления найденного ранее дефекта (new bug fix)
2. проверка, что найденного и исправленный ранее дефект не воспроизводится в системе снова( old bug fix)
3. проверка того, что ранее работающий функционал продолжает работать и изменение в новом билде не затронули его

Регрессионное тестирование включает повторные прогоны предыдущих тестов, а также проверку не попали ли регрессионные ошибки очередную версию в результате слияния кода

**Уровни тестирования**

**Приемочное тестирование**



**Приемочное тестирование** сконцентрирован на документации верхнего уровня, документации. Часто проводится заказчиком или будущими пользователями.

**User acceptance testing** - Тестирование с целью удостовериться, что продукт соотв нуждам и ожиданиям пользователя (могут быть alpha (выполняет qa или заказчик) , beta (выполняет пользователь)) ust(User acceptance testing) alpha выполняется в тестовой среде разработки, продукт может тестироваться qa и заказчиком.

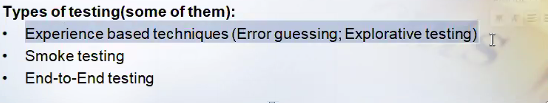
Beta проводится на стороне заказчика или ограниченным количеством реальных пользователей после альфа

**Operational acceptance testing** – нацелен на нефункциональную часть продукта, те нагрузочное тестирование, производительность, отказоустойчивость

**Contract testing** проверяет критерии принятия продукта, прописанные в контракте(пишет заказчик). В зависимости от проекта может существовать разное количества критерий

**regulation testing** проверяются соотв гос стандартам, стандартам безопасности(банковские системы, фармакологии )

**Типы тестирования**

****

**experience based techniques** основывается на опыте работы со схожими системами и сосредотачиваются на более ненадежных частях программы, не привязываясь к тестированию спецификации. Применяется в случае не детально описанных спецификации и отсутствие времени на прогон тест кейсов.

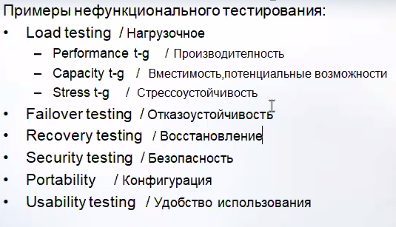
*Error guessing*- угадывание ошибок, выполняется после формального тестирования. Тестировщик находит дефекты в слабых местах, основываясь на своем опыте. У этой техники не гарантированная эффективность, т.к результат полностью зависит от опыта тестировщика.

*Explorative testing* – комбинирует опыт тестировщика и структурированный подход к тестированию. Может выполняться при отсутствии детальных спецификации или нехватки времени. Explorative testing выполняется без тест кейсов( тестировщик, который хорошо знает продукт находит ошибки.)

**Smoke testing –** выполняется с целью проверить, что критически важные функциональности в системе работают**,** проверяется стабильность системы в целом, чтобы дать зеленый свет(разрешить следующие действия) для проведения более тщательного тестирования. Перепроверка дефектов не является целью смоук . Дымовое тестирование может выполнятся перед функциональным и регрессионным. Может выполнятся вручную или автоматизировано.

**End-to-end testing** проверка продукта в реальных бизнес условиях и процессах, включает в себя взаимодействия с базами данных, с другими приложениями, системами, устройствами.

**Виды нефункционального тестирования**

****

**1.Нагрузочное-** проверка системы выдерживать разные типы нагрузок

Типы нагрузочного тестирования:

*Performance*- оценка производительность, проверяется поведение системы под нормальной нагрузкой.

Во время тестирования оцениваются следующ факторы:

1. количество поддерживаемых пользовательских сессий(поток)
2. количество свободных системных ресурсов
3. количество свободных аппаратных ресурсов

*Capacity –* тут проверяется сколько пользователей могут одновременно работать на протяжении длительной нагрузки системы

*Stress -*  проверка производительности и устойчивости системы при максимальной ее работе в условиях критической нехватки ее ресурсов. Целы – вывести систему из строя и определить условия при которых она не сможет далее функционировать

Такие типы нагрузочного тестирования часто выполняются для веб-систем и систем с открытым доступом и выполняются разными приложениями ( Jmeter, LoadRuner, HP Performance Center)

При нагрузочном тестировании учитываются Key Performance Indicators, то есть ключевые индикаторы производительности.

Индикаторы:

* Время отклика тестирования
* Максимальное количество поддерживаемых пользовательских сессий
* Максимальное количество транзакций, действий во временной интервал
* Количество используемых системных и аппаратных ресурсов

**2.Отказоустойчивость**- проверка поведения системы в случае сбоя серверов или других неблагоприятных факторов

**3.Recovery testing –** проверяется, что система восстанавливает свою функциональность и продолжает корректно работать после сбоев при этом обращается внимание на 2 фактора:

* Минимизация потерь данных в результате сбоя
* Минимизация времени между сбоем и продолжениям нормального функционирования

**4.Security testing** – проверяется, что информация в системе не теряется, не повреждается и ее не возможно подменить, а также невозможно получить несанкционированный доступ(важно проверять для систем, которые хранят или обрабатывают личные, коммерческие, государственные, любые тайные данные)

**5.Portability** **testing** - проверяется, что система будет работать при перемещении в другую среду( например с виндовс 8 в виндовс 10)

**6.Usability testing** – проверяется удобства использования пользовательского интерфейса