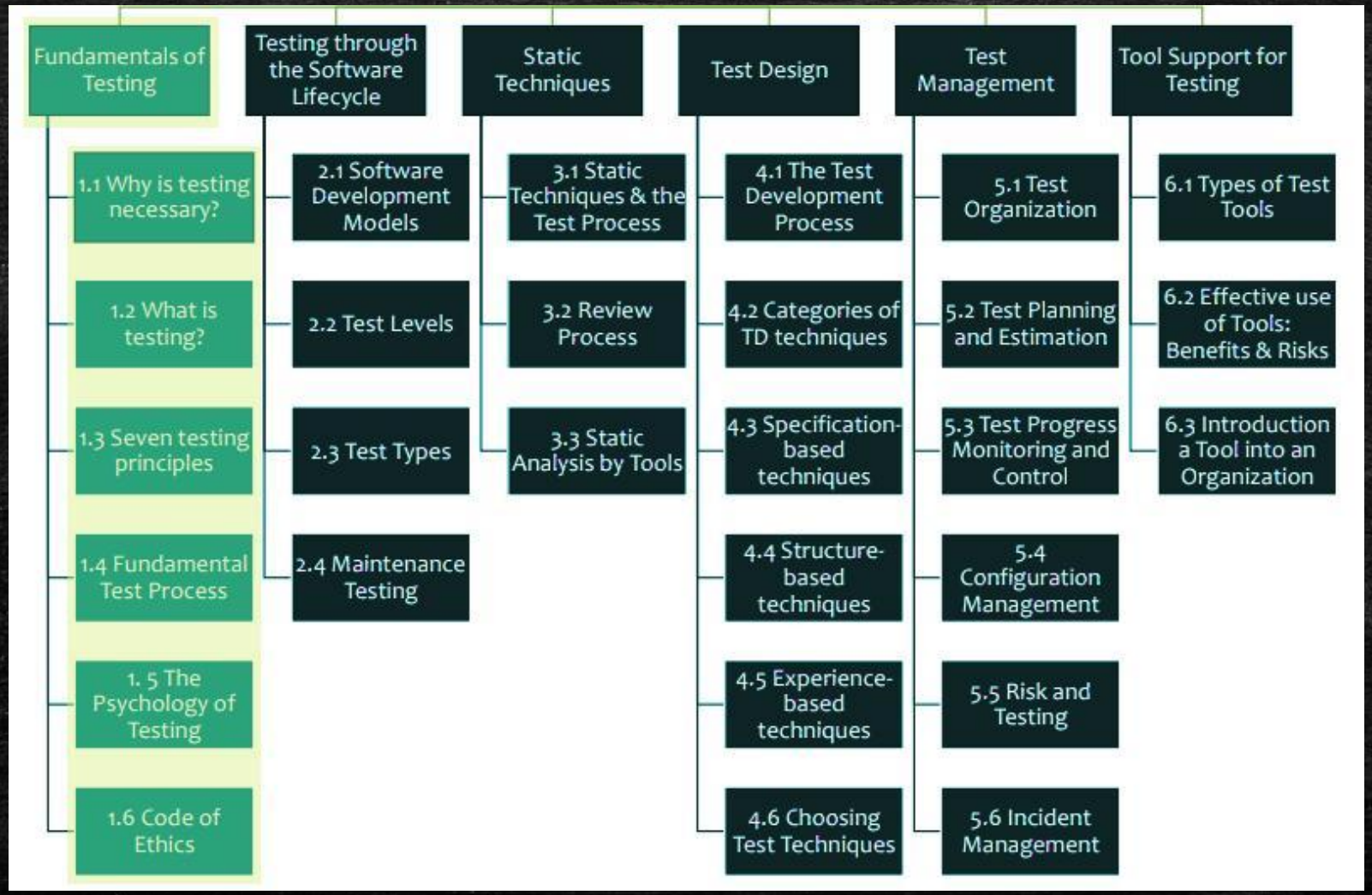
Тестирование через весь жизненный цикл софта

By CODIFY

Основы тестирования



Модели разработки ПО

Коробочный программный продукт
Итеративно - инкрементальная модель разработки ПО
Валидация
Верификация
V - модель

Зависимость тестирования

Адаптируемся!!!

Waterfall - модель

Одна из самых старых, подразумевает последовательное прохождение стадий, каждая из которых должна завершиться полностью до начала следующей.

Легко управлять проектом. Благодаря её жесткости, разработка проходит быстро, стоимость и срок заранее определены.

Каскадная модель будет давать отличный результат только в проектах с четко и заранее определенными требованиями

Тестирование начинается только после того, как разработка завершена или почти завершена.

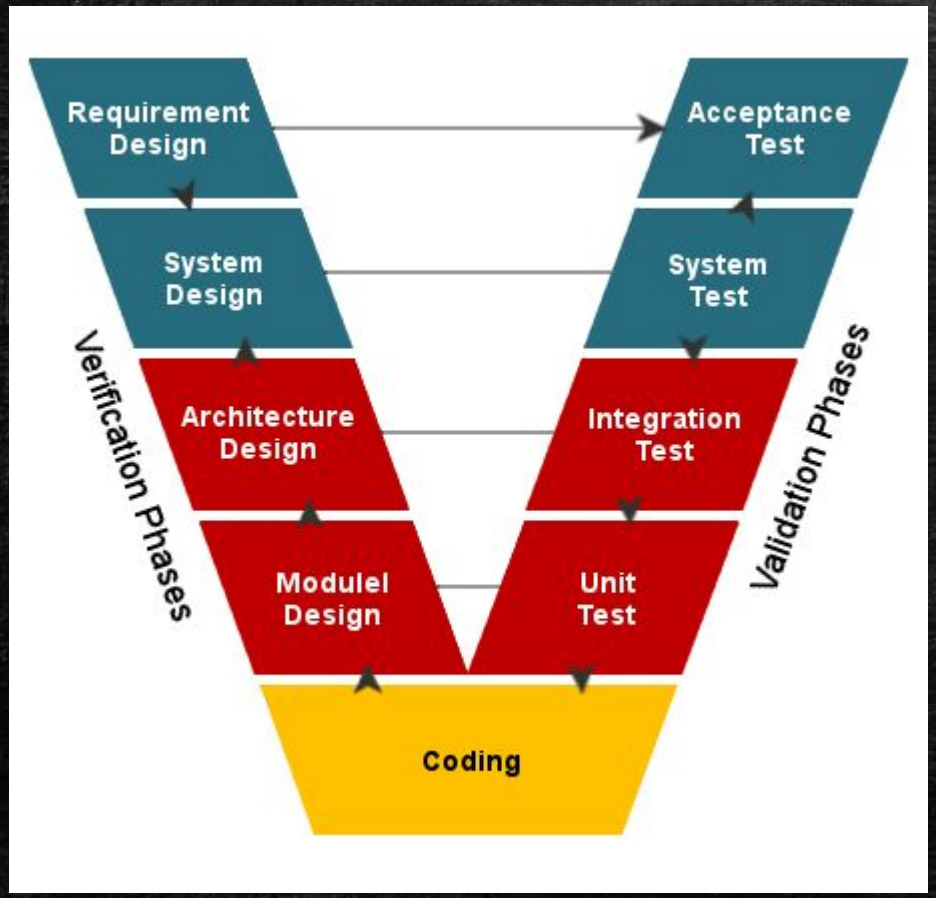
Waterfall - модель

Стоимость внесения изменений высока, так как для ее инициализации приходится ждать завершения всего проекта.

Когда использовать каскадную методологию?

- Только когда требования известны, понятны и зафиксированы. Противоречивых требований не имеется.
- Нет проблем с доступностью программистов нужной квалификации.
- В относительно небольших либо краткосрочных проектах.

Модели разработки ПО



Унаследовала структуру «шаг за шагом» от каскадной модели.

V-образная модель применима к системам, которым особенно важно бесперебойное функционирование:

Например, прикладные программы в клиниках для наблюдения за пациентами, интегрированное ПО для механизмов управления аварийными подушками безопасности в транспортных средствах

V- модель

Особенностью модели

- Направлена на тщательную проверку и тестирование продукта, находящегося уже на первоначальных стадиях проектирования.
- Стадия тестирования проводится одновременно с соответствующей стадией разработки, например, во время кодирования пишутся модульные тесты.

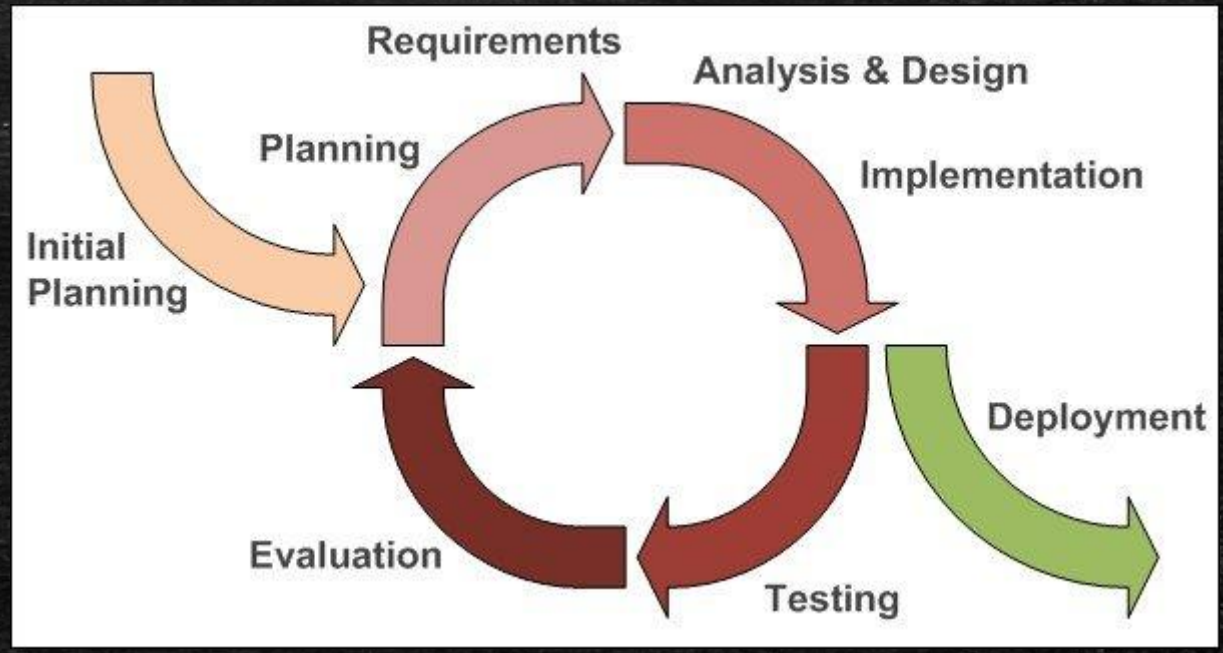
Когда использовать V-модель?

- Если требуется тщательное тестирование продукта, то V-модель оправдает заложенную в себя идею: validation and verification.
- Для малых и средних проектов
- Требования четко определены и фиксированы.
- У инженеров и тестировщиков есть необходимая квалификация

Итеративно-инкрементные модели

Например:
RUP
RAD
Agile

Задачи тестирования:
Оперативность
Регрессия



Итеративно-инкрементные модели

- Цикл разделен на более мелкие, легко создаваемые модули.
- Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования.
- Предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых «инкрементов».
- Процесс продолжается до тех пор, пока не будет создана полная система.

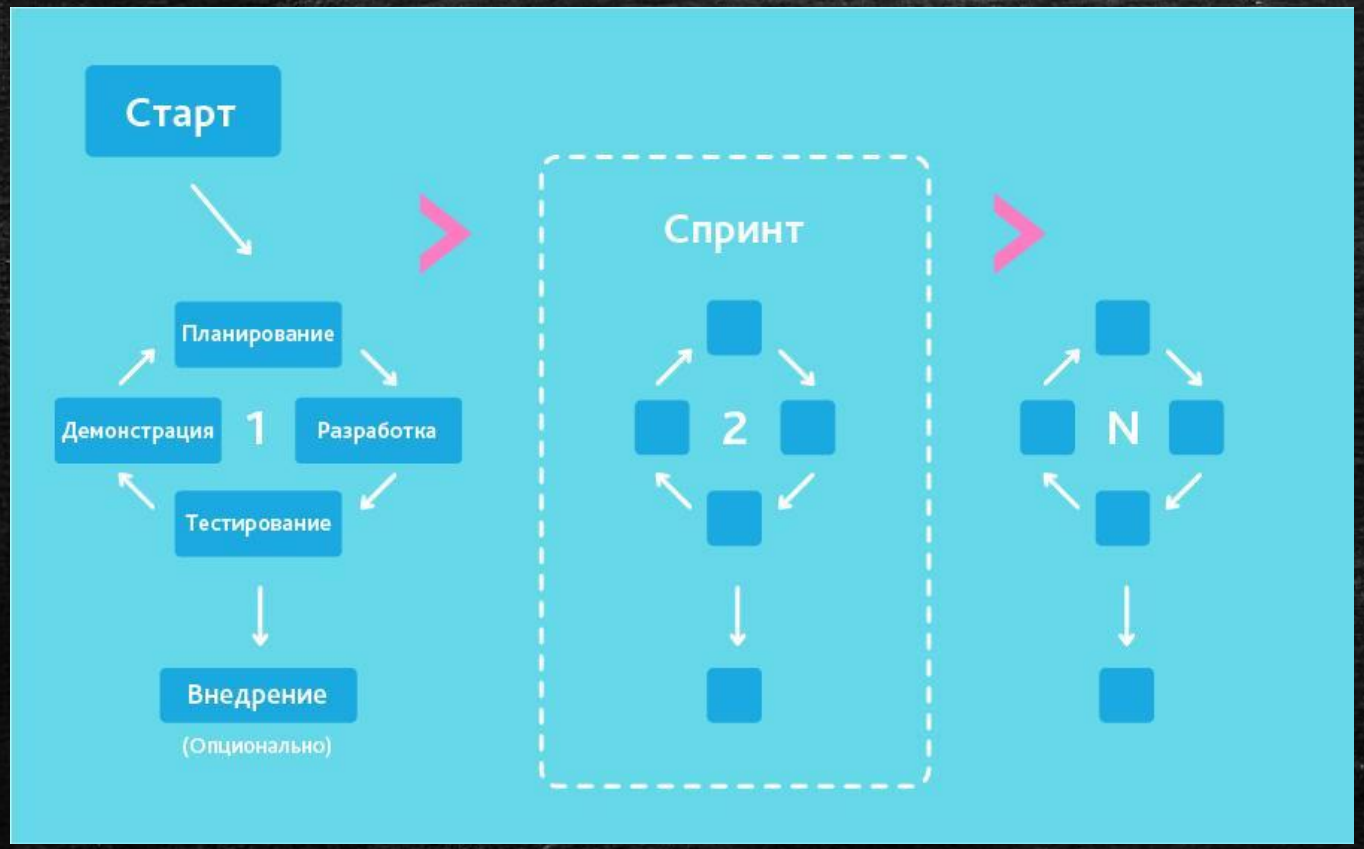
Итеративно-инкрементные модели

Инкрементные модели используются там, где отдельные запросы на изменение ясны, могут быть легко формализованы и реализованы

Когда использовать инкрементную модель?

- Когда основные требования к системе четко определены и понятны.
- В то же время некоторые детали могут дорабатываться с течением времени.
- Требуется ранний вывод продукта на рынок.
- Есть несколько рисковых фич или целей.

Agile - философия

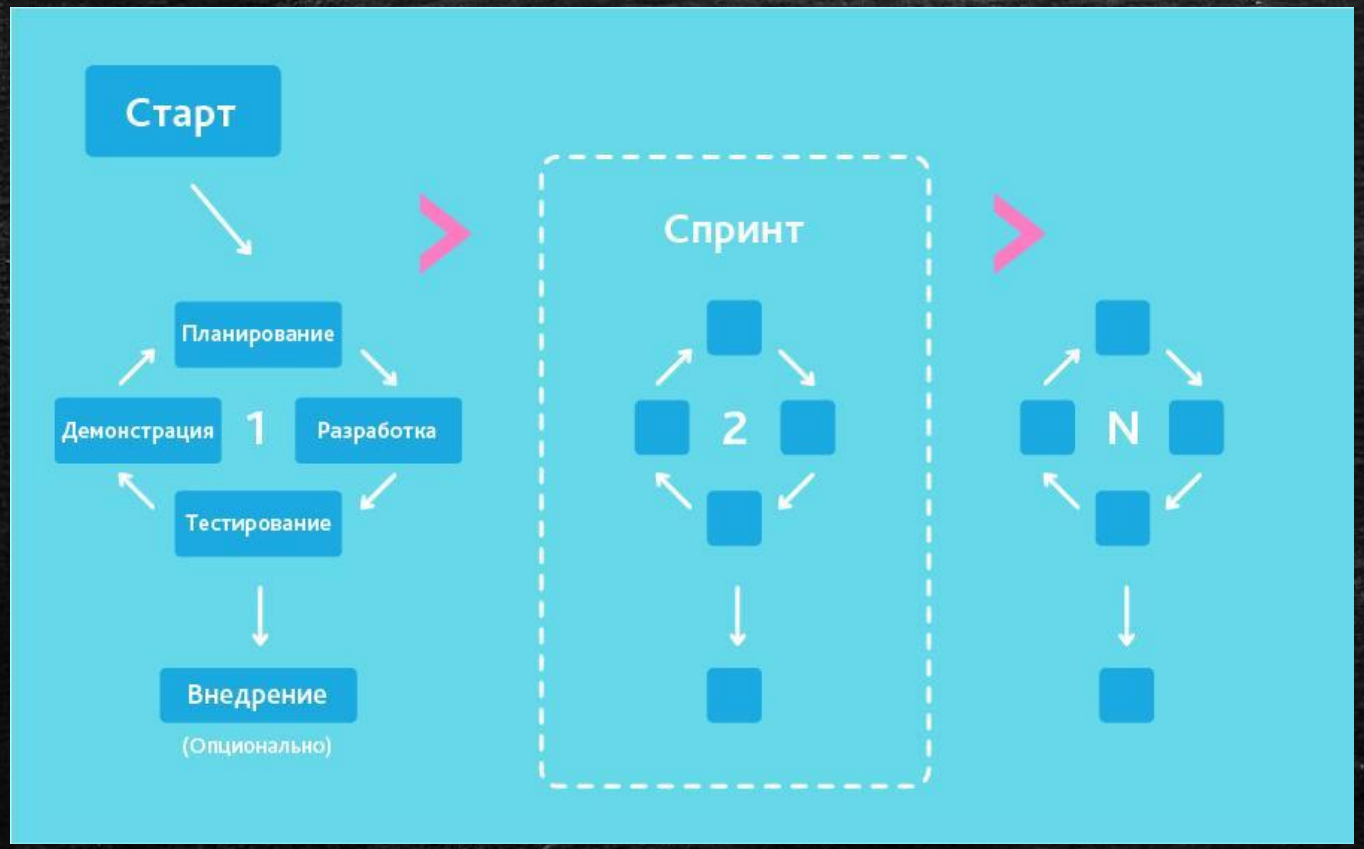


В «гибкой» методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет.

Из-за отсутствия конкретных формулировок результатов сложно оценить трудозатраты и стоимость, требуемые на разработку.

В основе такого типа — непродолжительные ежедневные встречи — «Scrum call/meeting» и регулярно повторяющиеся собрания

Agile - философия

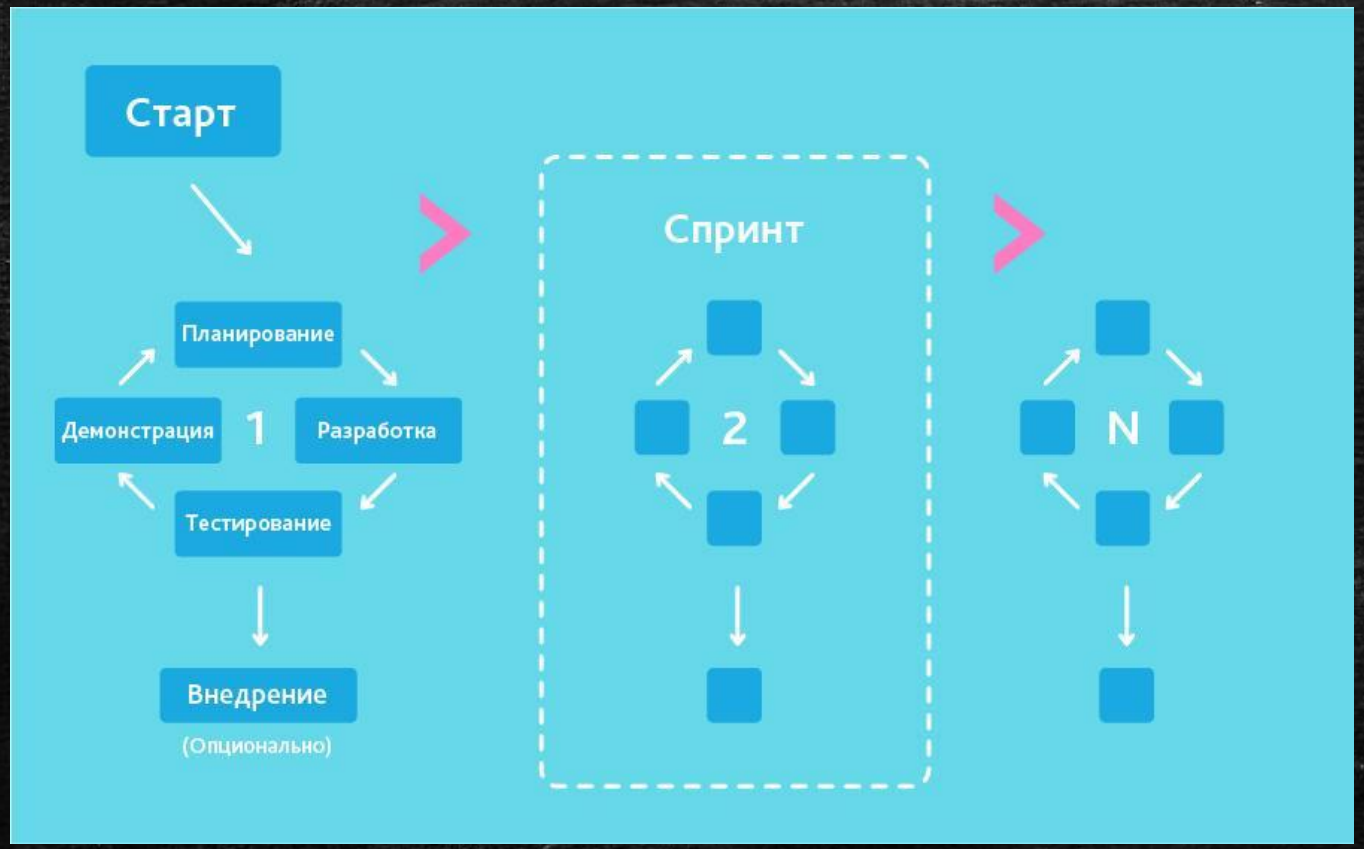


На ежедневных совещаниях участники команды обсуждают:

1. отчёт о проделанной работе с момента последнего Scrum’a;
2. список задач, которые сотрудник должен выполнить до следующего собрания;
3. затруднения, возникшие в ходе работы.

Методология подходит для больших или нацеленных на длительный жизненный цикл проектов, постоянно адаптируемых к условиям рынка

Agile - философия



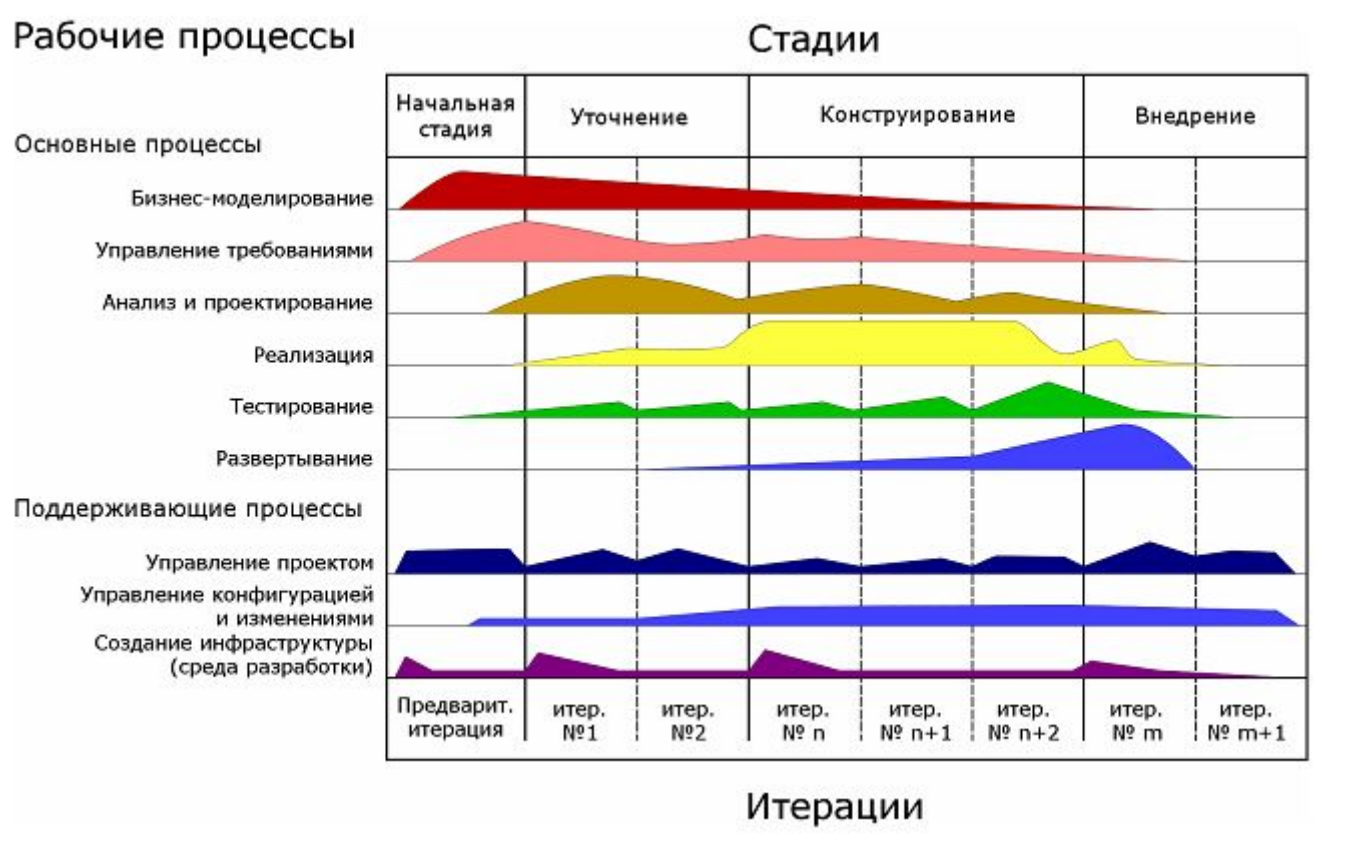
Когда использовать Agile?

- Когда потребности пользователей постоянно меняются в динамическом бизнесе.
- Изменения на Agile реализуются за меньшую цену из-за частых инкрементов.
- В отличие от модели водопада, в гибкой модели для старта проекта достаточно лишь небольшого планирования.

Оперативность в Agile

Agile -> Регрессия -> Автоматизация

RUP - рациональный унифицированный процесс

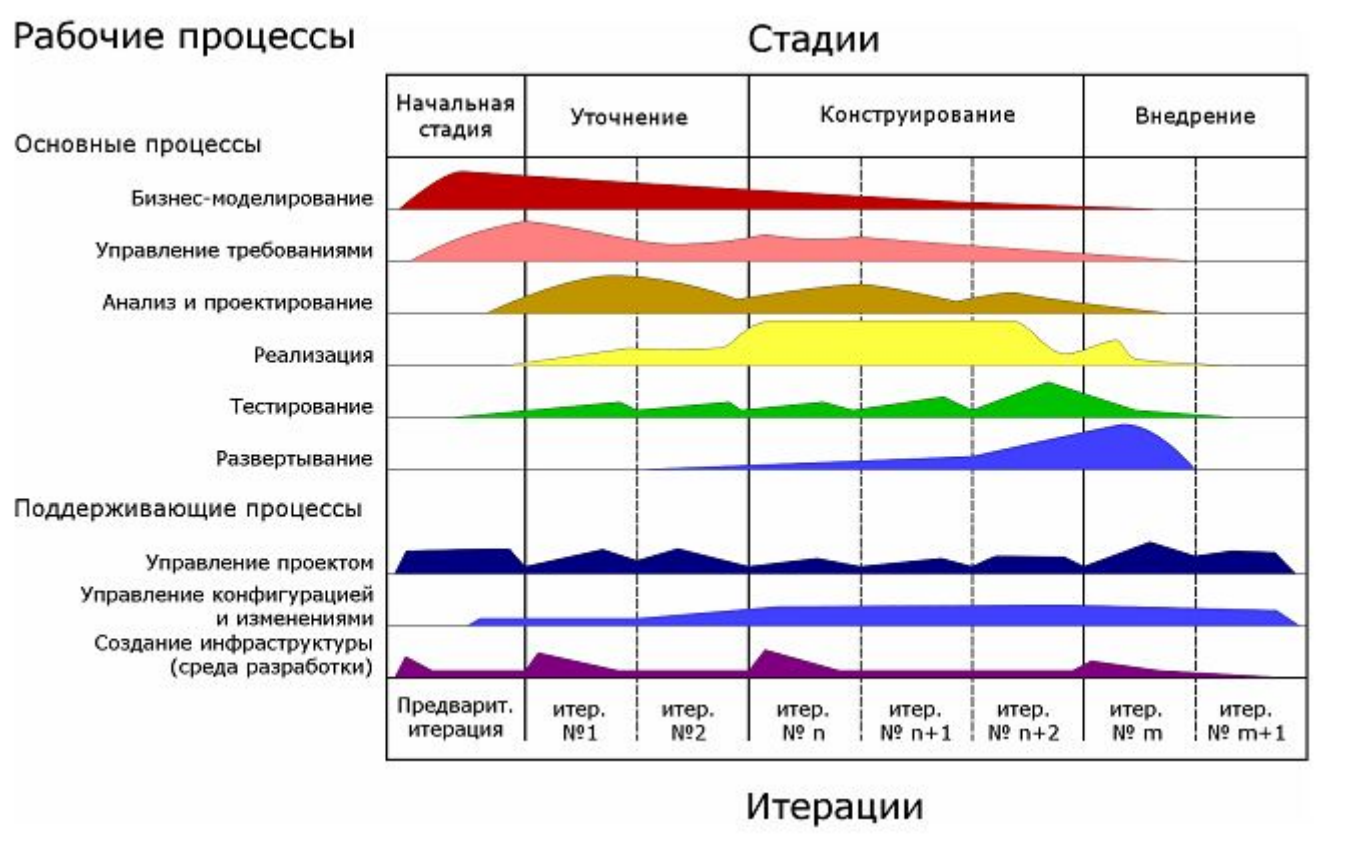


Разделение проекта на несколько мелких проектов, которые выполняются последовательно, и каждая итерация разработки четко определена целями, которые должны быть достигнуты в конце итерации

RUP достаточно хорошо формализован, и наибольшее внимание уделяется начальным стадиям разработки проекта — анализу и моделированию.

Методология направлена на снижение коммерческих рисков (risk mitigating) посредством обнаружения ошибок на ранних стадиях разработки.

RUP - рациональный унифицированный процесс

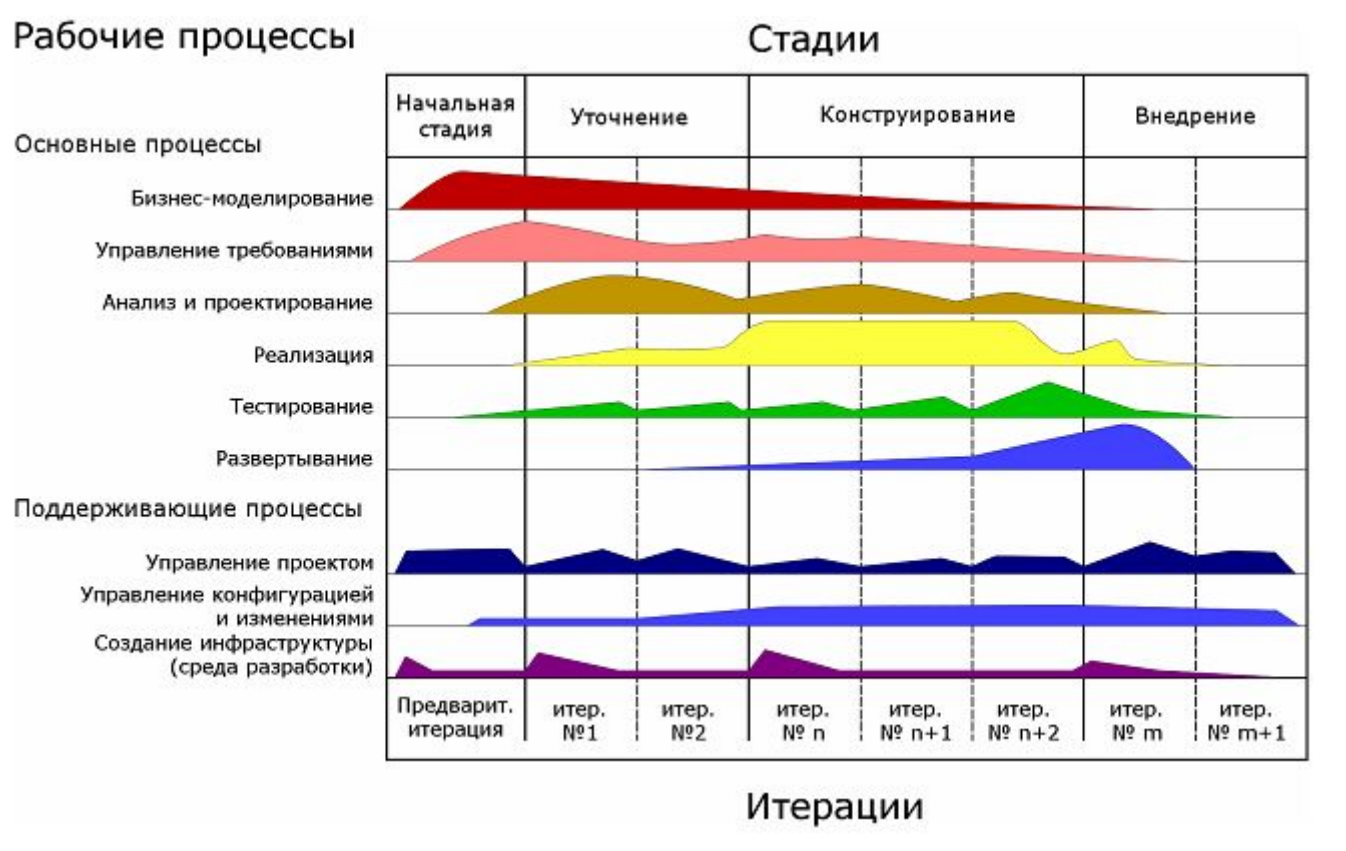


Процесс делится на четыре основные фазы во времени (milestones):

- Inception — понимание, что мы создаем. Фаза сбора информации и анализа требований, определение образа проекта в целом;

- Elaboration — понимание, как мы это создаем. Фаза анализа требований и проектирования системы, планирование необходимых действий и ресурсов, спецификация функций и особенностей дизайна;

RUP - рациональный унифицированный процесс



• Construction — создание бета-версии продукта. Основная фаза разработки и кодирования, построение продукта как восходящей последовательности итераций (версий кода);

• Transition — создание конечной версии продукта. Фаза внедрения продукта, поставка продукта конкретному пользователю

Стоит специально отметить, что regression testing должно содержать все актуальные тесты от предыдущей итерации. Ведущая роль - архитектор

Коробочный продукт

Коробочный (тиражный) программный продукт - коммерческое готовое программное обеспечение. ПО разработанное для широкого рынка и поставляемое большинству в одинаковой конфигурации

Коробочный продукт

Заказное ПО

Заказное (кастомное)ПО – ПО, разработанное специально для группы пользователей, заказчика

Коробочное + кастомное ПО

Верификация

Верификация - подтверждение исследованием и через объективные доказательства, того , что указанные требования были выполнены

Валидация

Валидация - подтверждение исследованием и через предоставление объективных доказательств, что требования для указанного, предполагаемого использования или применения были выполнены

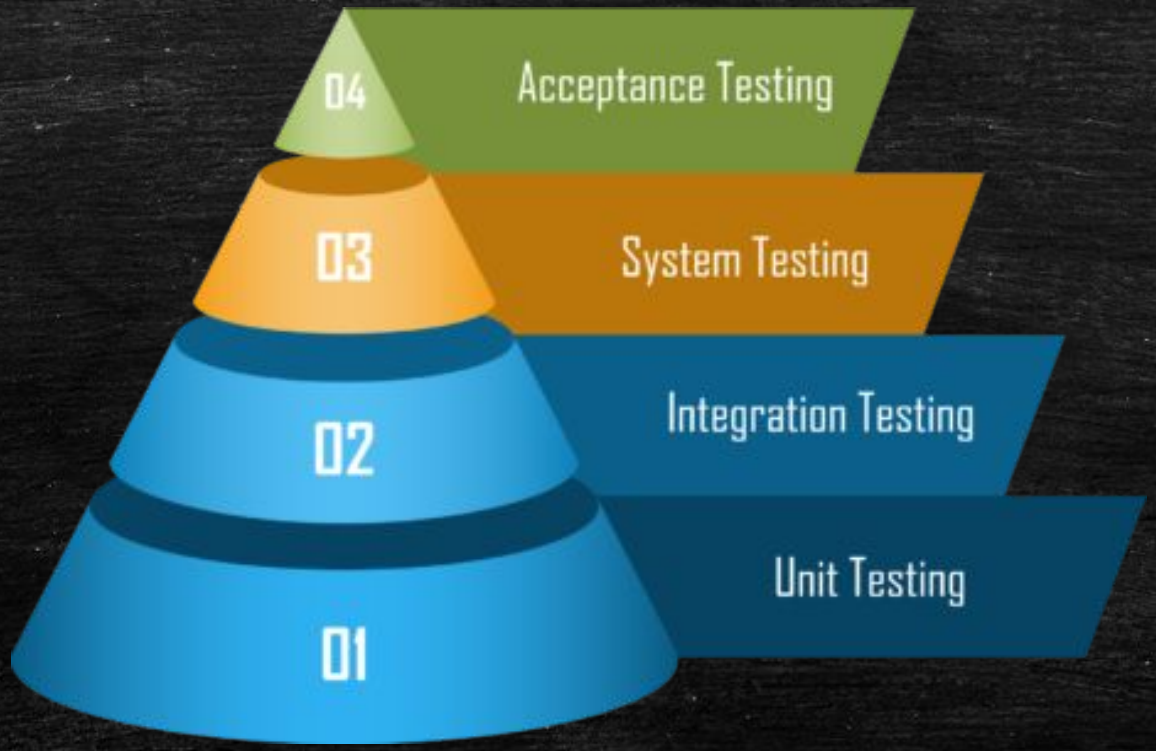
Верификация vs Валидация

- Верификация включает в себя проверку документов, дизайна, кода, а Валидация, включает в себя тестирование и валидация самого продукта.
- Верификация - без выполнения кода. Валидация - выполнение кода.
- Верификация использует методы, как ревью, прогоны, инспекции, а Валидация black box тестирование, white box тестирование и нефункциональное тестирование.
- Верификация проверяет подтверждает ли разрабатываемое ПО спецификации, а Валидация проверяет соответствует ли ПО требованиям и ожиданиям
- Верификация находит баги на ранних этапах процесса разработки. Валидация находит баги, которые верификация не смогла поймать.
- Верификация нацелена на архитектуру ПО, дизайн, БД, а Валидация на конечный продукт
- С начала делается верификация, а потом уже Валидация

Критерии качества тестирования

- Каждому процессу разработки, соответствует свой процесс тестирования
- Каждый уровень тестирования имеет свои цели тестирования
- Анализ и дизайн тестов для какого-либо уровня тестирования должны начинаться одновременно с соответствующей деятельностью разработчиков
- Тестировщики должны быть вовлечены в процесс просмотра и рецензирования документов, как только становятся доступными их предварительные версии

Уровни тестирования

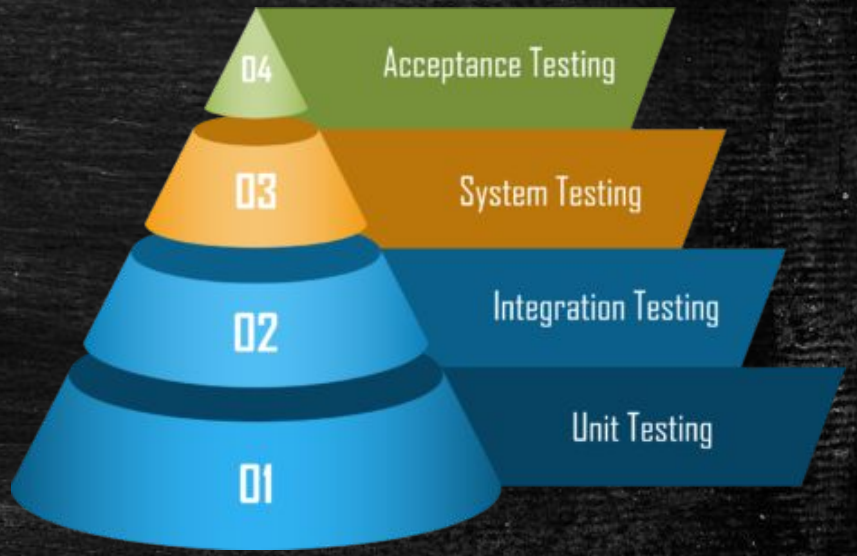


Уровни тестирования

- Альфа тестирование
- Бета тестирование
- Компонентное тестирование
- Драйвер
- Тестирование в условиях эксплуатации
- Интеграция
- Интеграционное тестирование
- Тестирование надежности
- Заглушка
- Системное тестирование
- Тестовое окружение
- Уровень тестирования
- Разработка управляемая тестированием
- Пользовательское приемочное тестирование

Уровень тестирования

Уровень тестирования - управляемая и организованная группа тестовых активностей . Уровень тестирования связан с обязанностями в проекте. Примеры уровней тестирования: компонентное(unit) тестирование, интеграционное тестирование, системное тестирование и приемочное тестирование



Адаптируемся !!!

Уровни тестирования

Цели ?
Базис?
Объекты?
Дефекты?
Формат результата?

Уровни тестирования

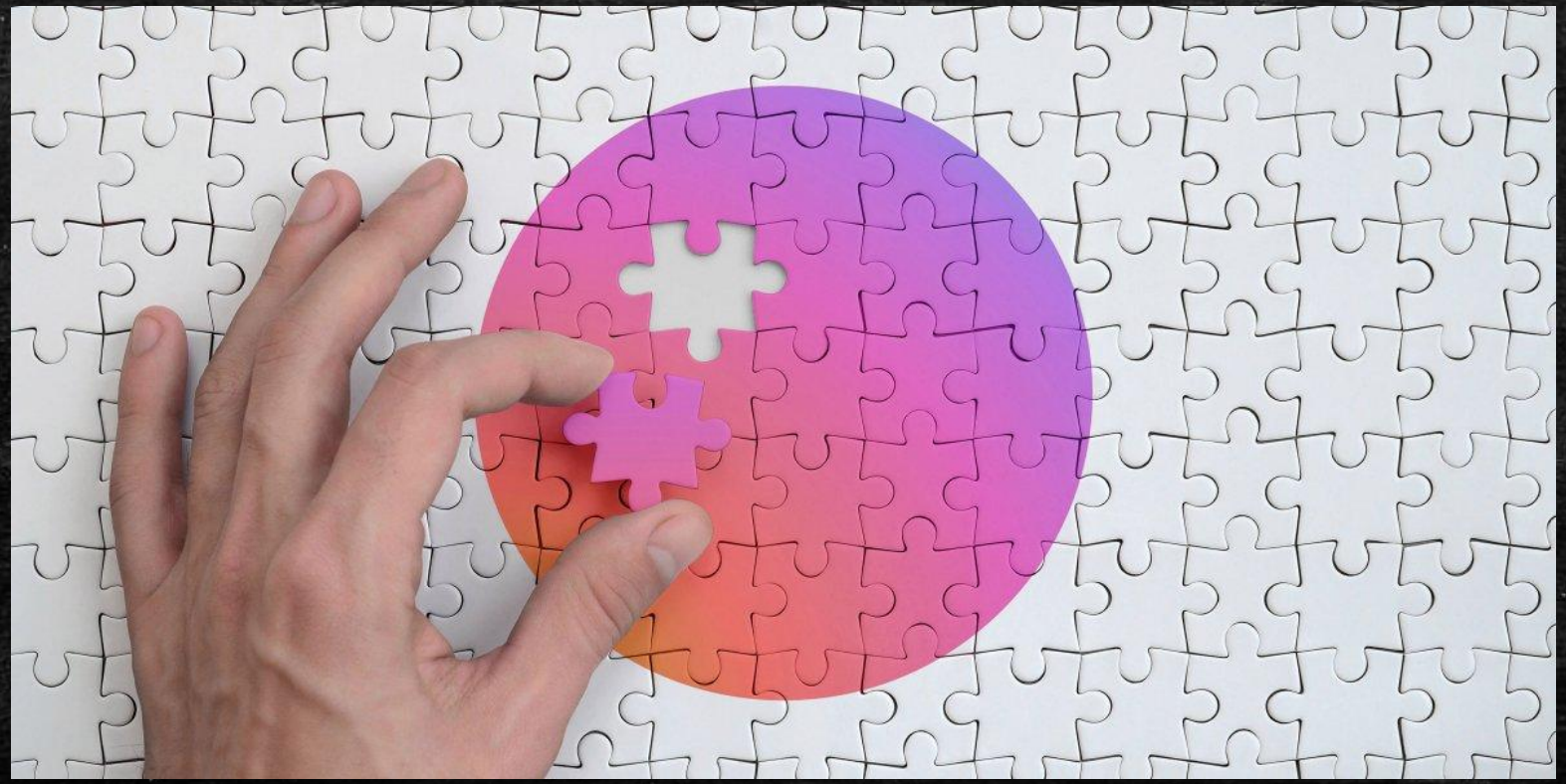
Компонентное тестирование -> дефекты -> спецификация-> код

Приемочное тестирование -> функционирует и выполняет требования заказчика клиента -> юзер мануал-> система

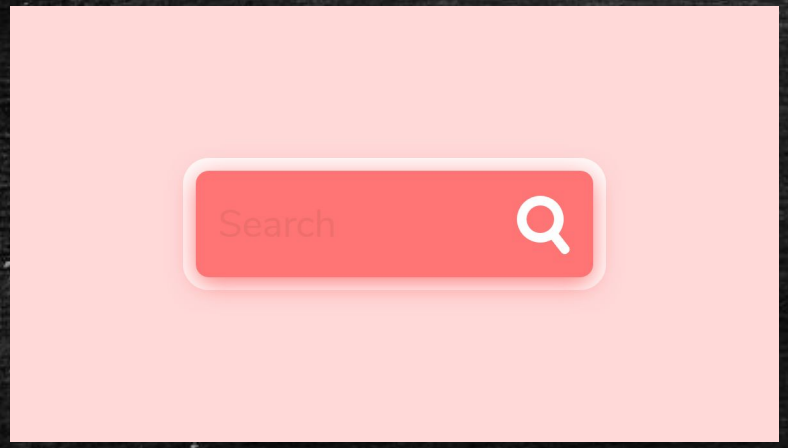
Компонент, интеграция, система

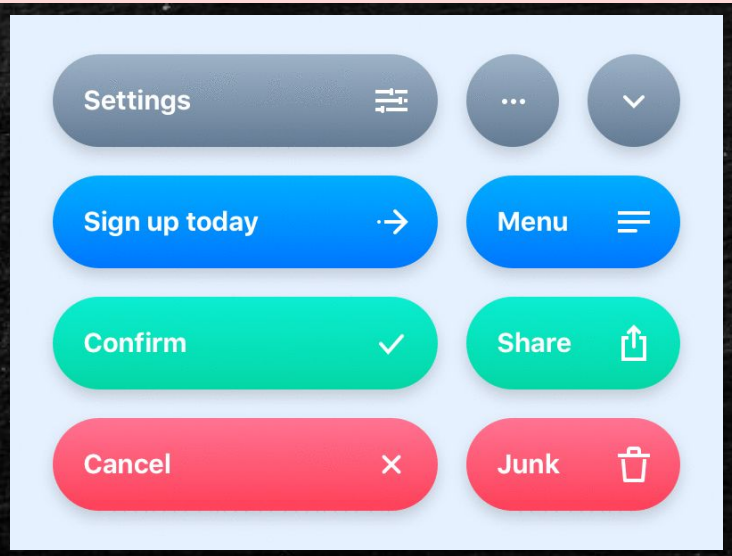
Компонент, интеграция, система

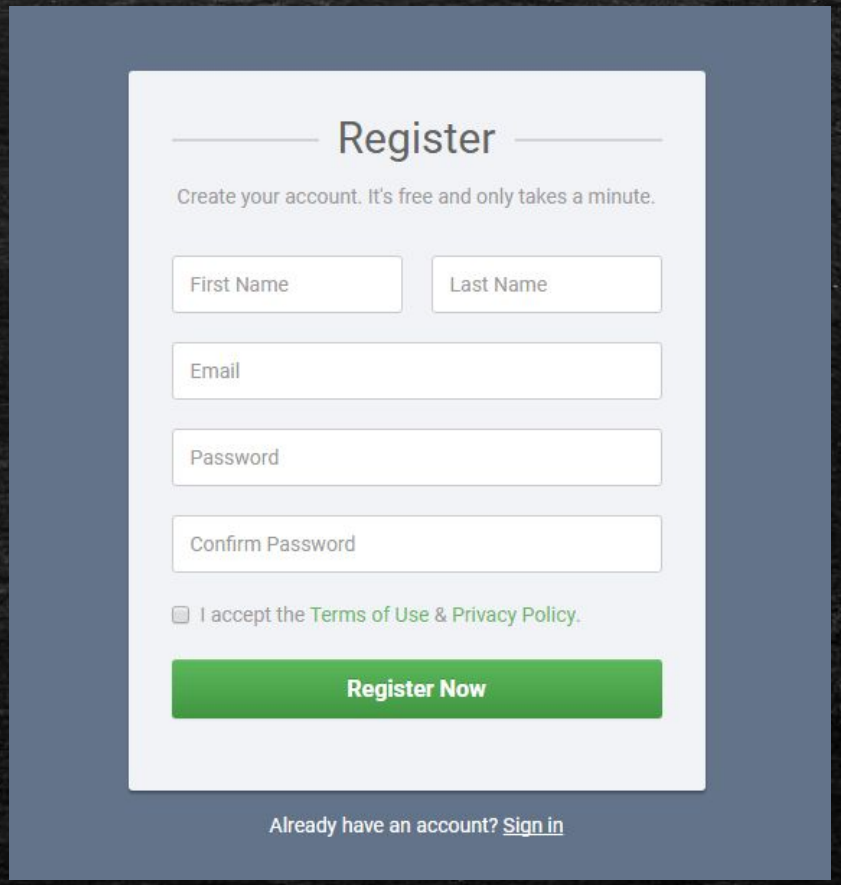
Компонент , модуль, программа- наименьший элемент ПО, который может быть протестирован отдельно



Компонент, интеграция, система

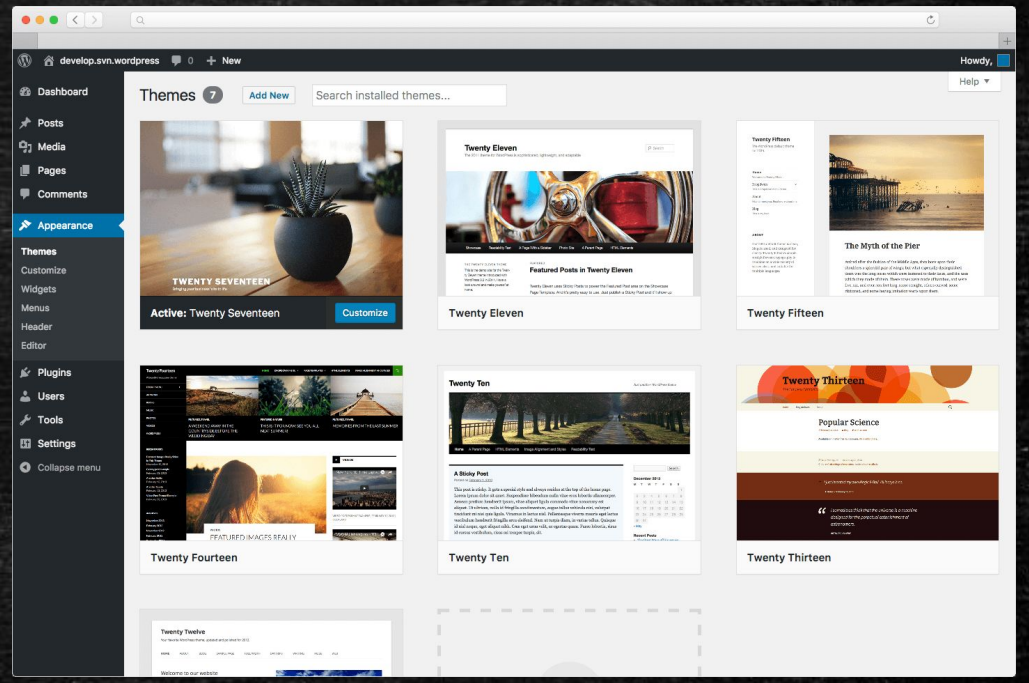






Компонент, интеграция, система

Система - совокупность компонентов, объединенная для выполнения определенной функции или набора функций



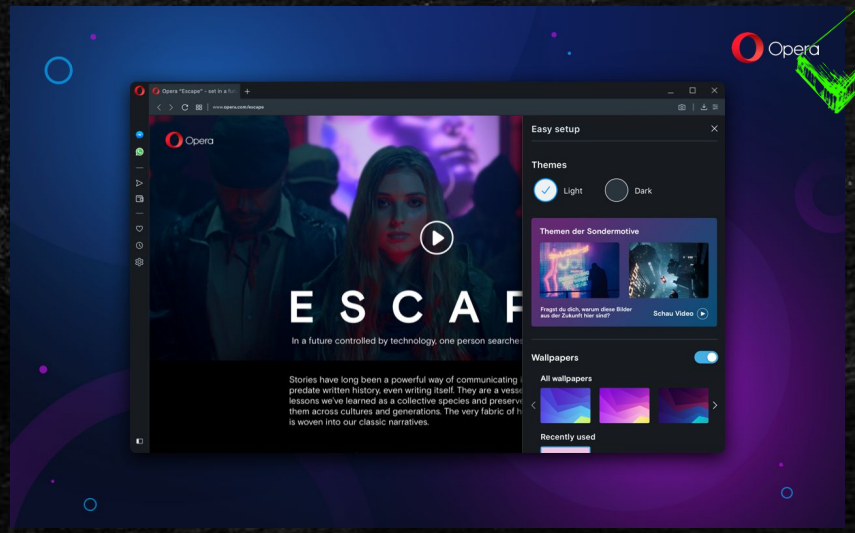
Компонент, интеграция, система

Интеграция - процесс объединения компонентов или систем в большую структуру

Уровни тестирования

Компонентное (unit) тестирование - тестирование отдельных компонентов ПО

Компонентное тестирование

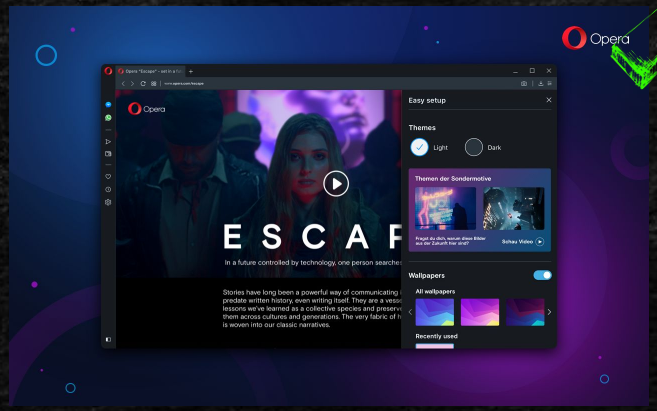


Драйвер

Драйвер - компонент ПО или тестовый инструмент, который заменяет компонент обеспечивающий управление или вызов компонента или системы

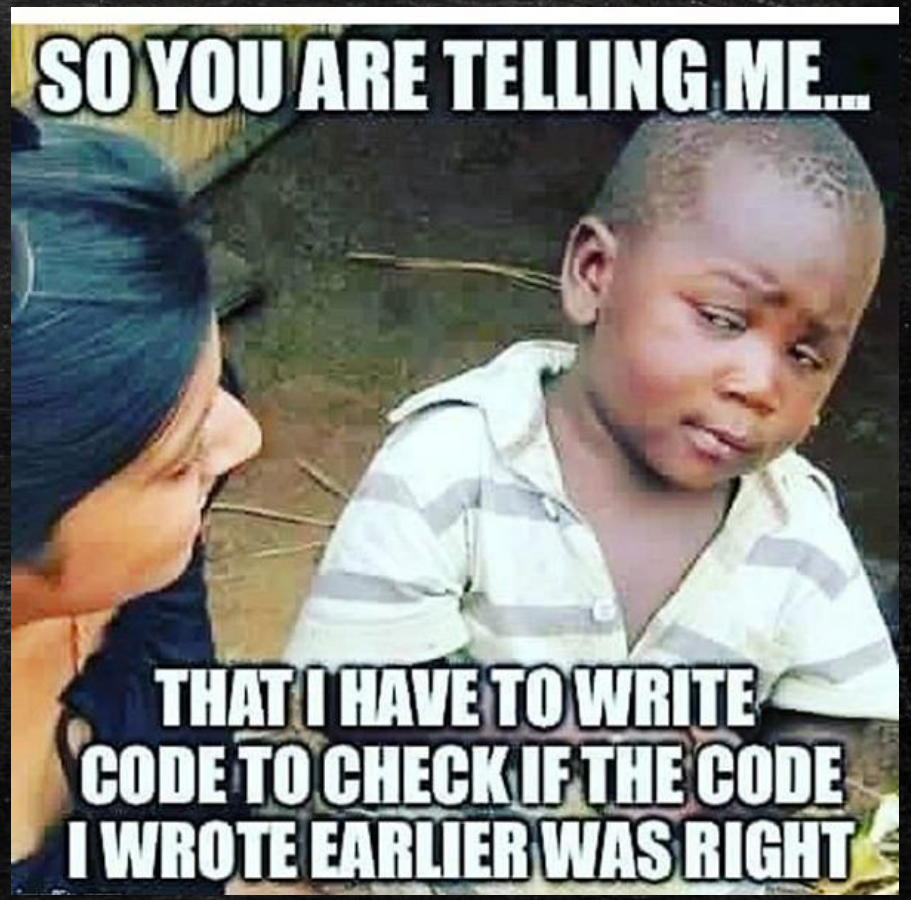
Заглушка

Заглушка - специализированная или минимальная реализация компонента, использующаяся для подмены компонента, от которого зависит разработка или тестирование другого компонента или системы



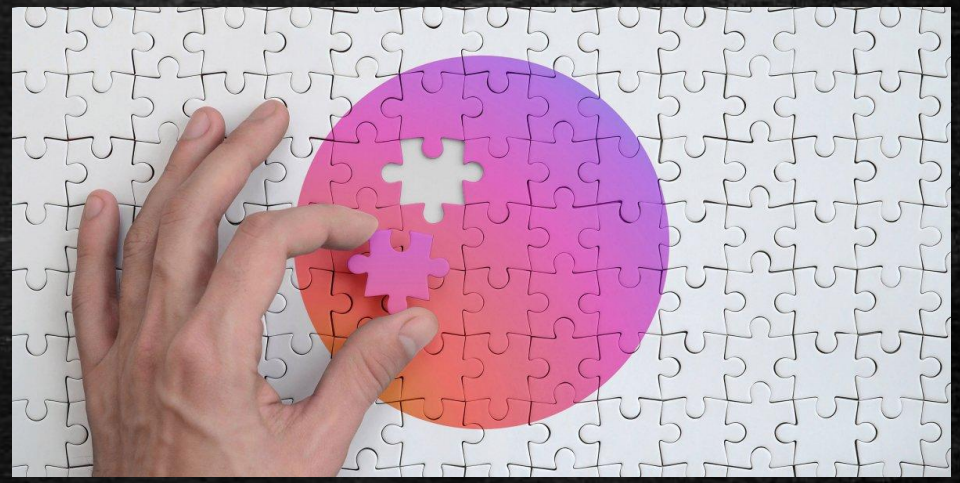
Компонентное тестирование

TDD (Test driven development) Разработка управляемая тестирование - способ разработки ПО, где тест кейсы разрабатываются и автоматизируются до того как будет написан сам код, который будет прогонять данные тест кейсы



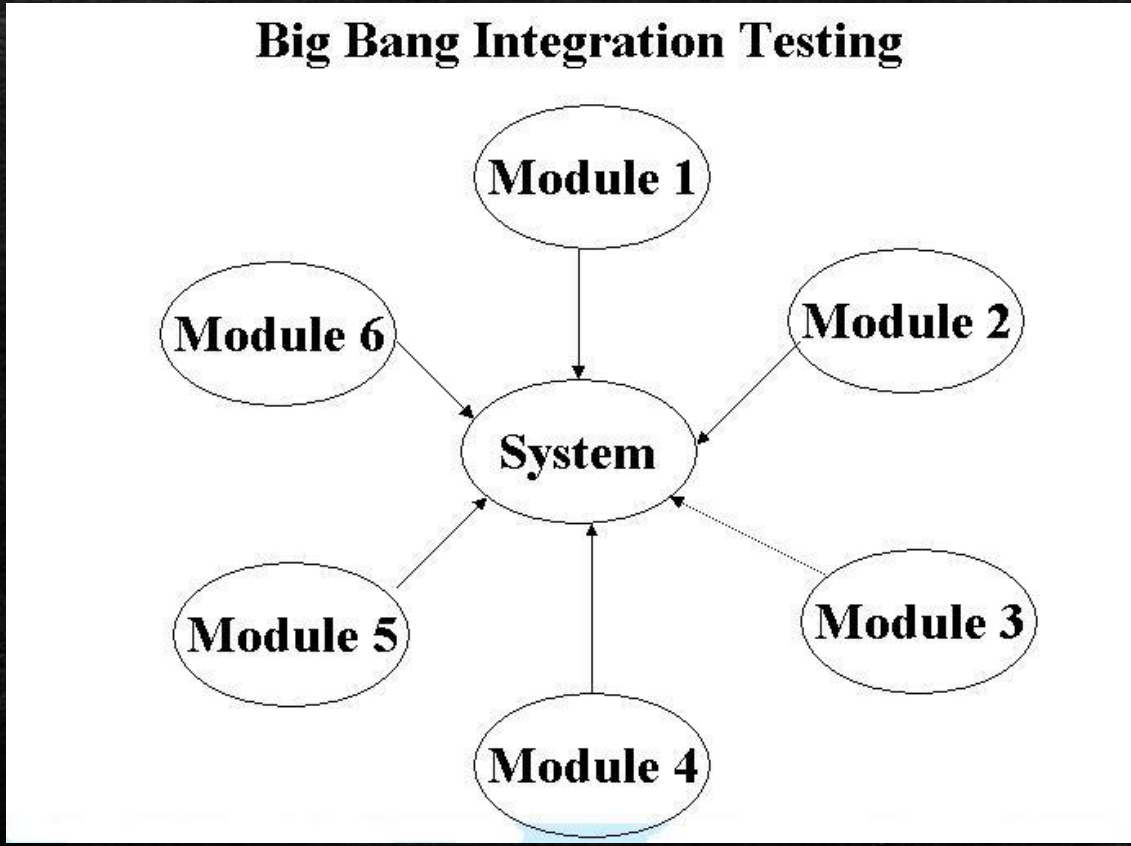
Интеграционное тестирование

Интеграционное тестирование - тестирование выполняемое для обнаружение дефектов в интерфейсах и во взаимодействии между объединенными компонентами или системами



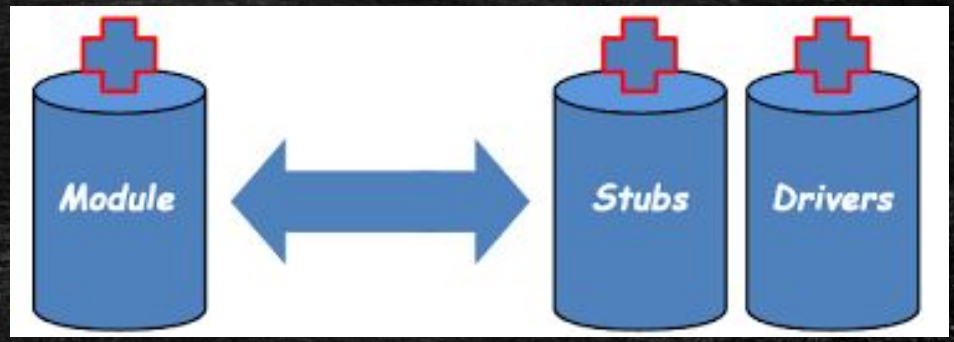
Стратегии интеграционного тестирования

Стратегия “Большого взрыва”



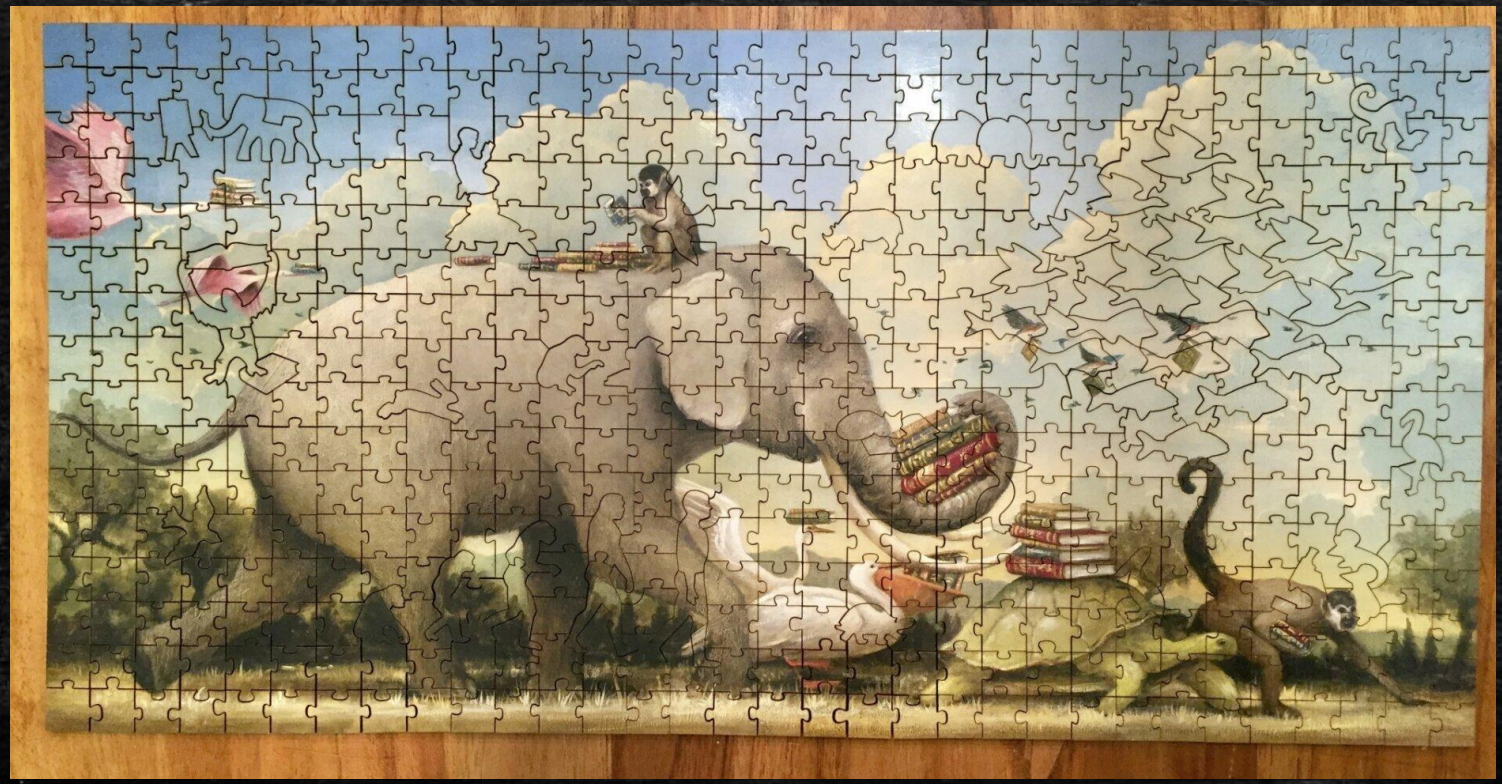
Стратегии интеграционного тестирования

Интеграция по нарастающей и убывающей

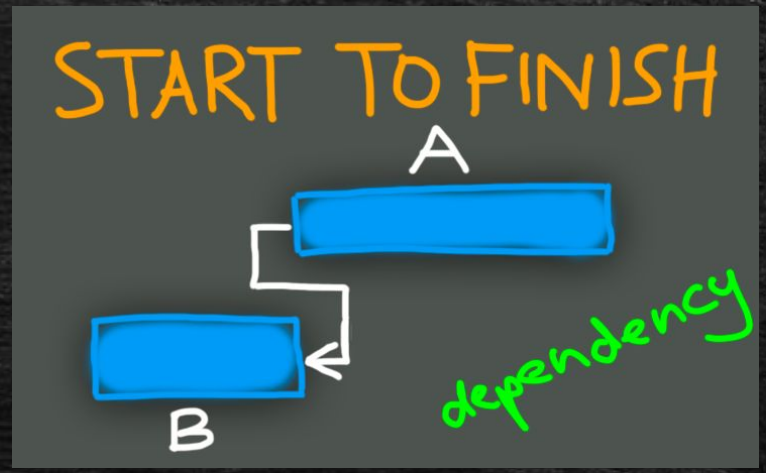


Системное тестирование

Системное тестирование - процесс тестирования системы в целом, с целью проверки того, что она соответствует установленным требованиям



Системное тестирование



Базис - > функциональные, нефункциональные (глобальные) требования -> тестовое окружение (эксплуатационное, !антивирус, фаервол, версия базы данных и т.д.)

Приемочное тестирование

Приемочное тестирование – формальное тестирование, по отношению к потребностям , требованиям и бизнес процессам конечного пользователя, проводимое для определения соответствия ПО, критериям приемки, а также дать возможность заказчикам или иным лицам определить, принимать ПО или нет

Приемочное тестирование

Критерии приемки: скорость ПО, как оно выглядит, пользователь проверяет пользовательские сценарии

Тестирование может проводится заказчиком, конечным пользователем, а иногда, если согласовано, то самой фирмой и ее тестировщиками.

Приемочное тестирование

- Пользовательское
- Эксплуатационное
- Контрактное и правовое
- Альфа
- Бета

Приемочное тестирование

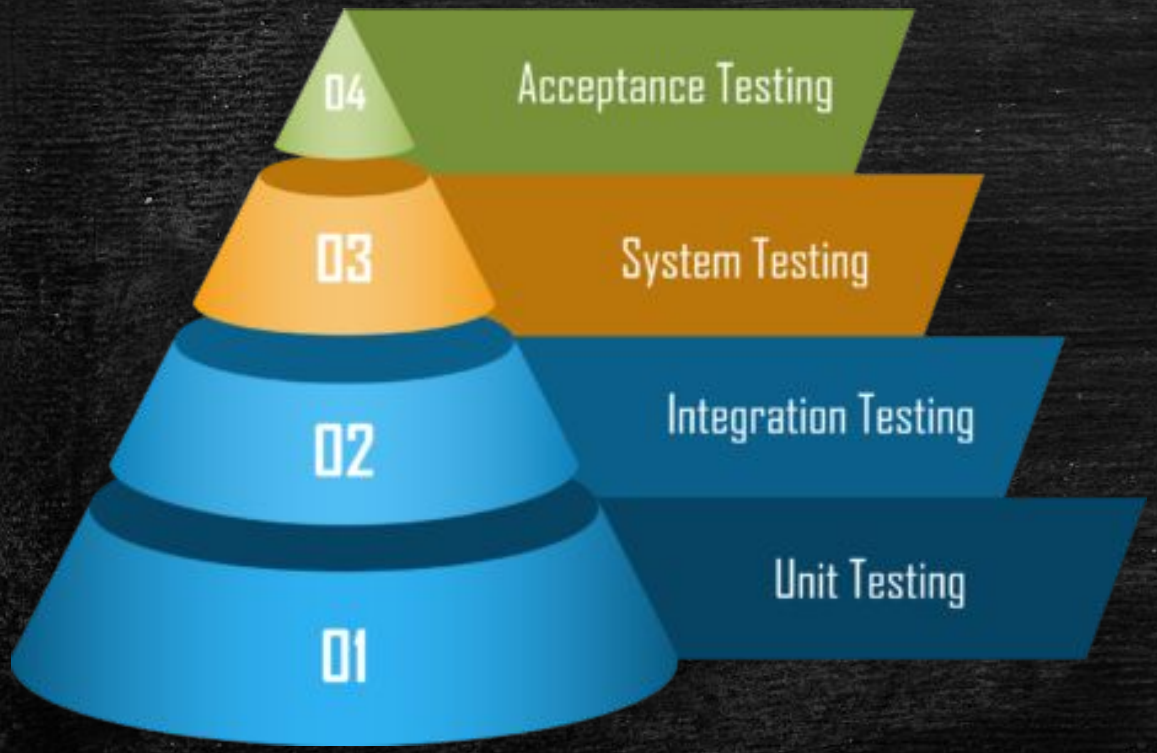
Альфа тестирование - моделируемое, действительное эксплуатационное тестирование потенциальными пользователями или независимой командой тестирования вне разрабатывающей организации. Часто применяется к коробочному ПО

Приемочное тестирование

Бета тестирование - эксплуатационное тестирование потенциальными пользователями или независимой командой тестирования вне разрабатывающей организации с целью определения действительно ли ПО удовлетворяет требованиям клиента

Тестирование надежности - тестирование устойчивости ПО

Зачем делить на уровни?



1. Чтобы видеть на каких уровнях есть тестирования. Если нет уровня, то вы теряете в объеме, глубине тестирования. То есть на этом уровне скорее всего есть дефекты
2. Хорош бы идти по уровням и предоставлять полноценную обратную связь разработчикам
3. Экономия ресурсов, если идем снизу вверх, иногда можем идти наоборот сверху вниз