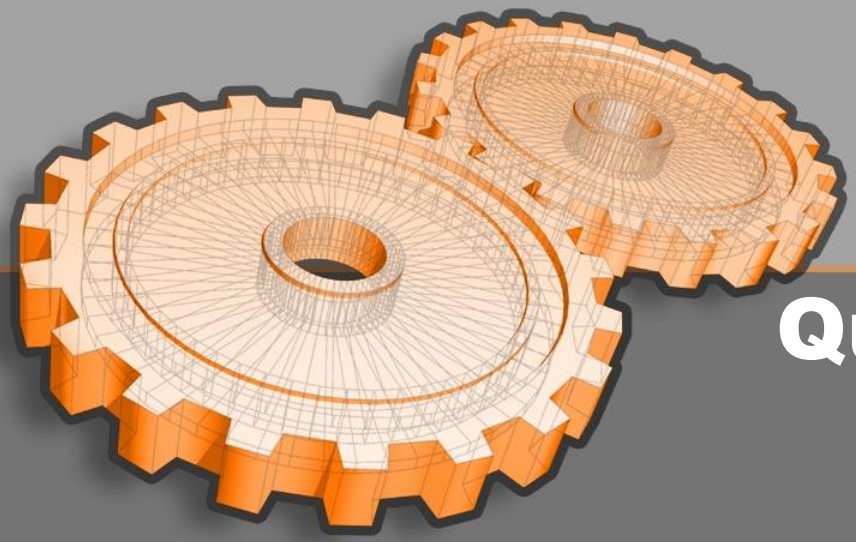
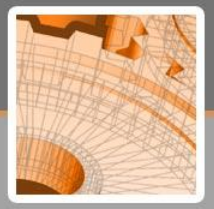
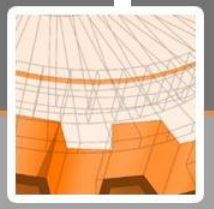


Курс лекций

Quality Assurance in software development

Лекция 7

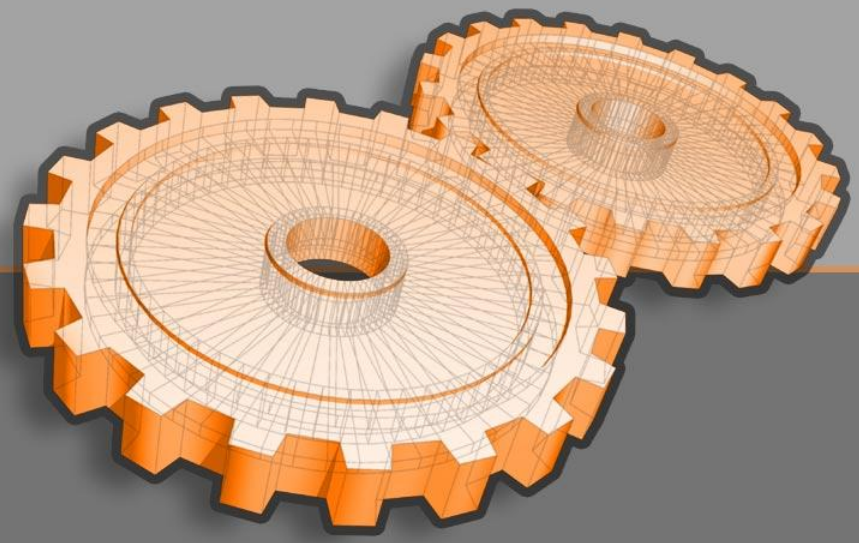
Тест дизайн

Лектор: Ильиных Дмитрий

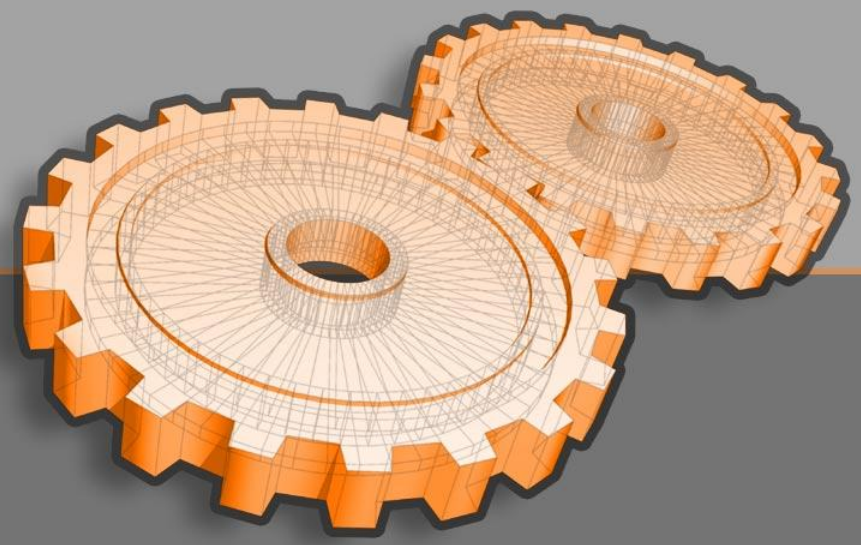
Саратов, 2015

Тема лекции

Тест дизайн

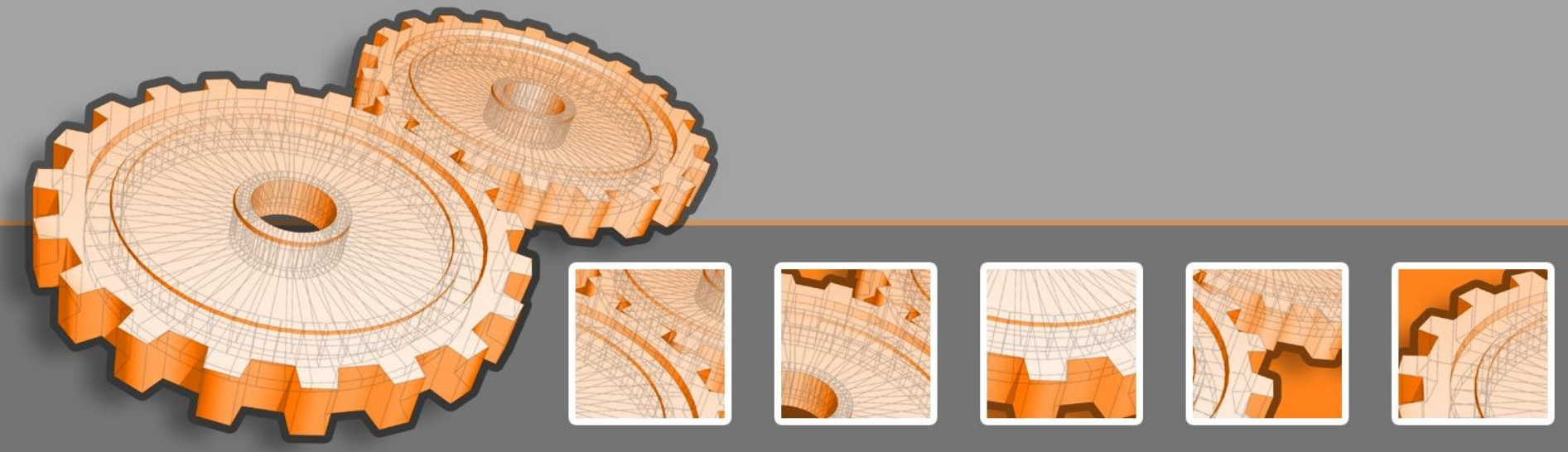


Основные вопросы лекции

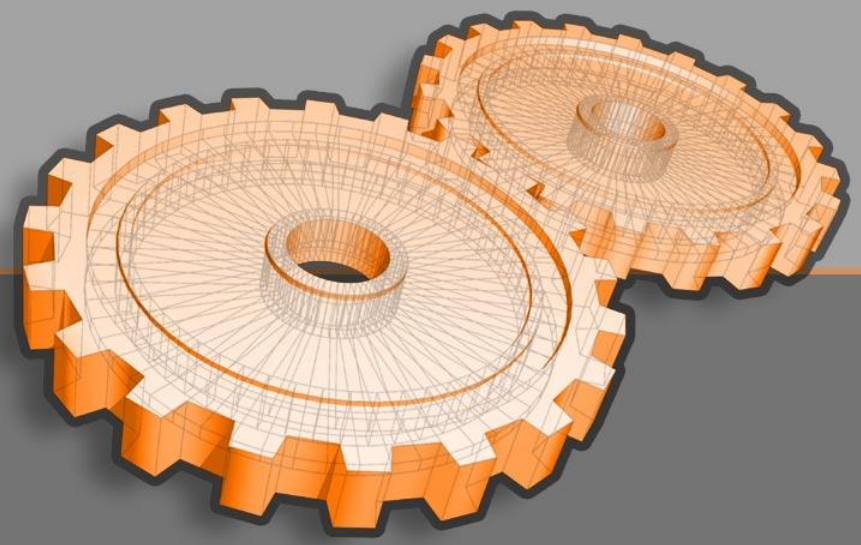


Основные вопросы лекции

- Определение понятия “Тест дизайн”
- Техники тест дизайна
- Тестовое покрытие
- Анализ и приоретизация тестов



Ввдение понятия “Тест дизайн”

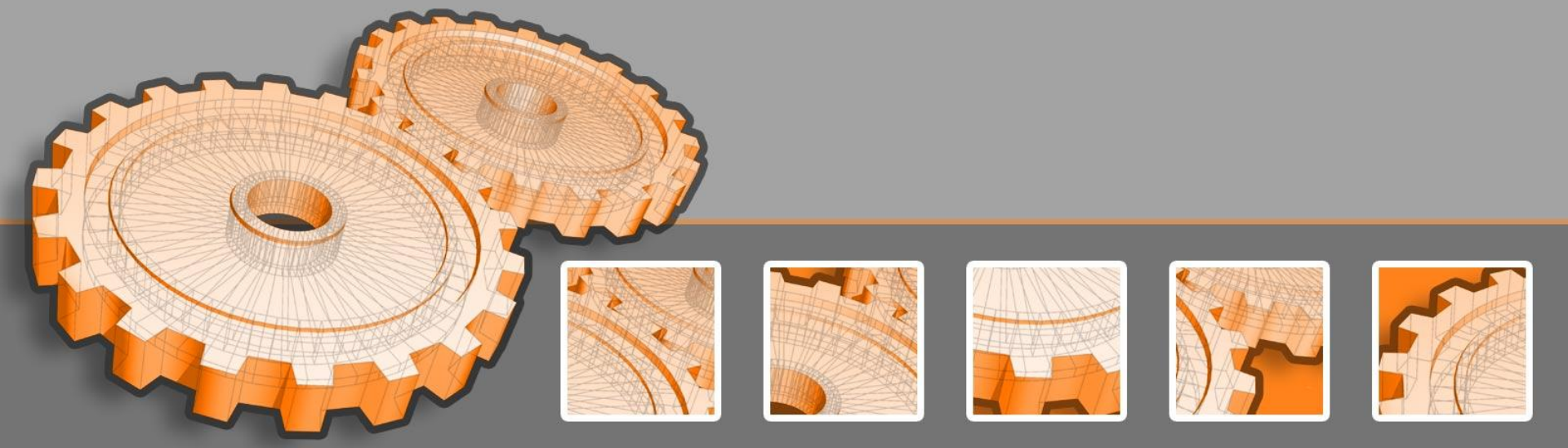


Постулат тестирования

- “Software has bugs”

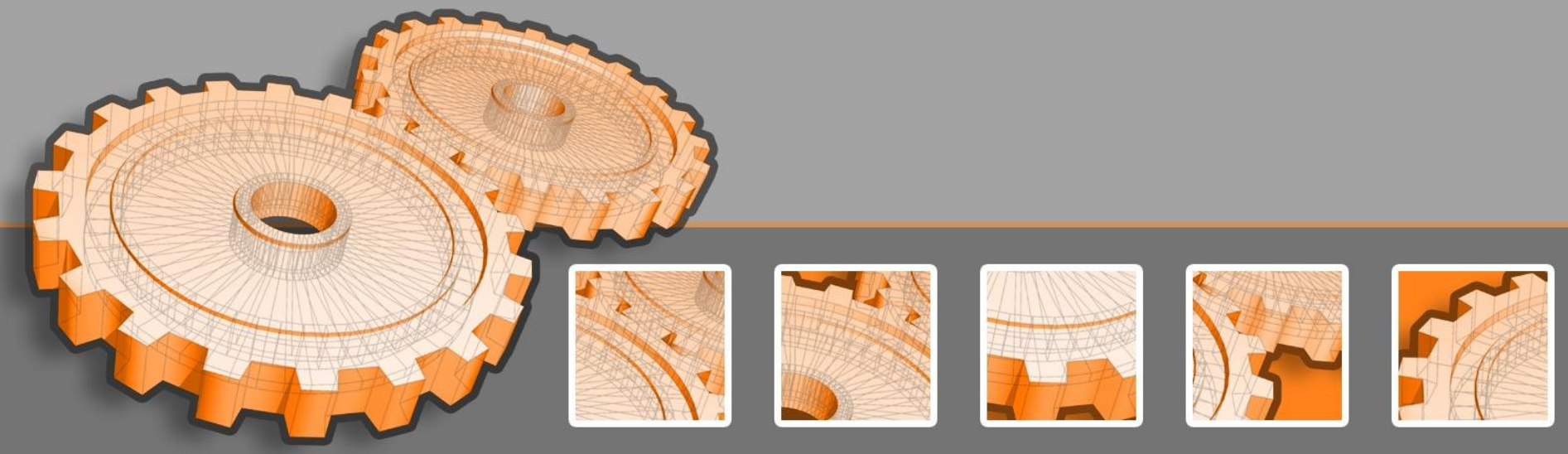
Определение тест дизайна

- **Тест дизайн** – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.



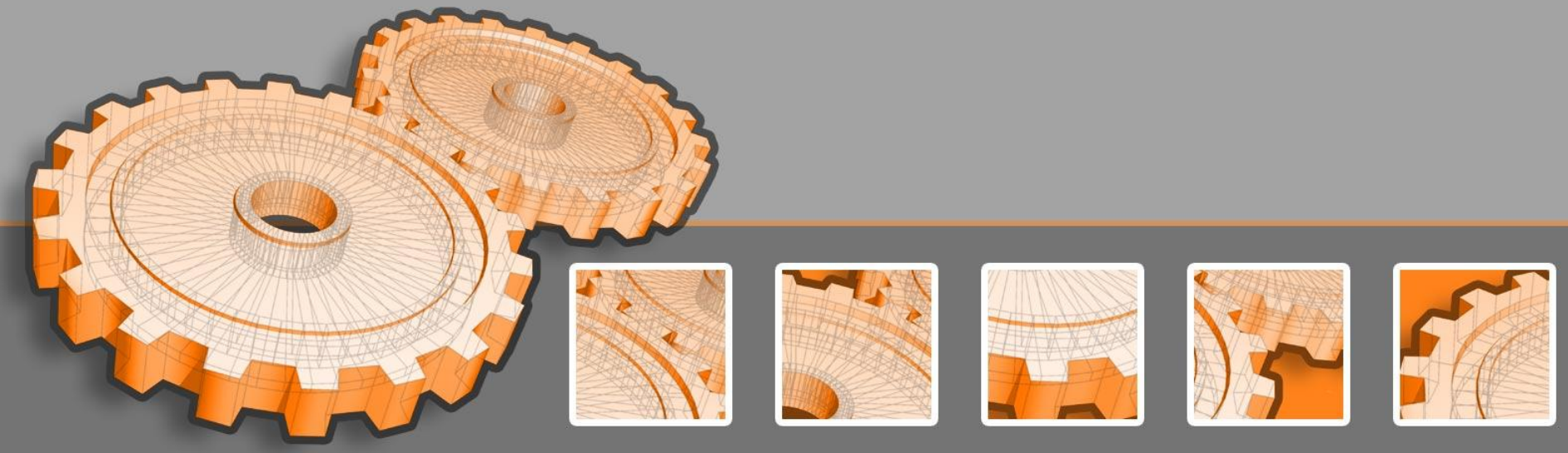
Введение понятия “Тест дизайн”

- В тест дизайне обычно выделяют:
  - Анализ объёма работ
  - Определение и описание тестовых случаев
  - Определение и структурирование тестовых процедур
  - Обзор и оценка тестового покрытия



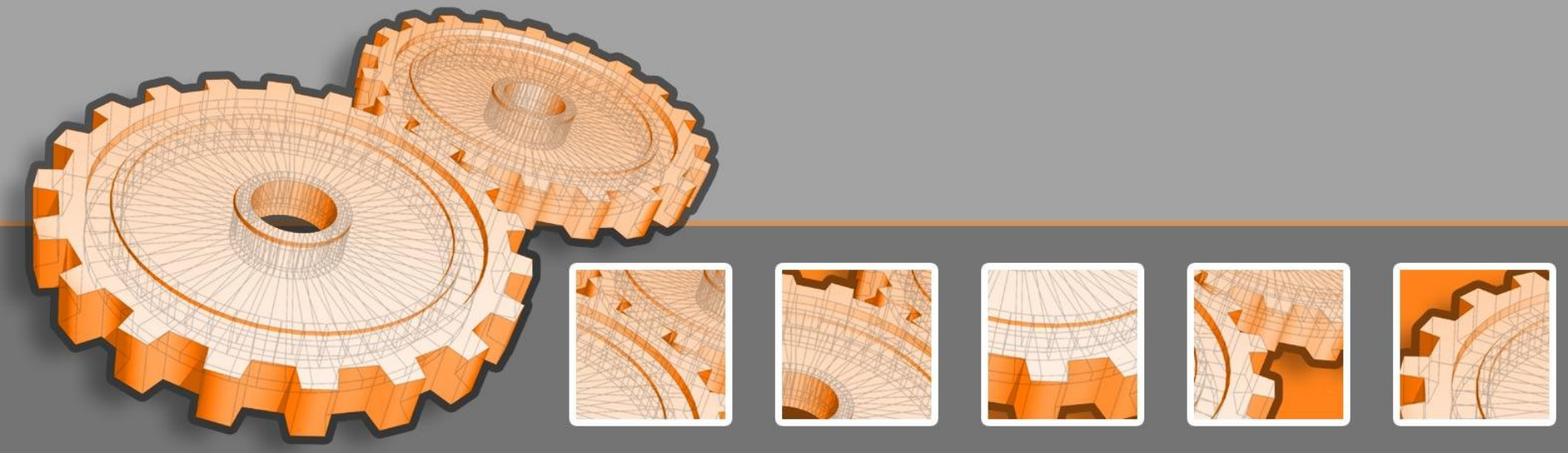
Определение тест кейса

- **Тест-кейс (Test Case)** - это артефакт, описывающий:
  - совокупность шагов для выполнения,
  - конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части,
  - описание ожидаемого результата, который должен быть проверяться на каждом отдельном шаге либо в разультате выполнения всей последовательности шагов.



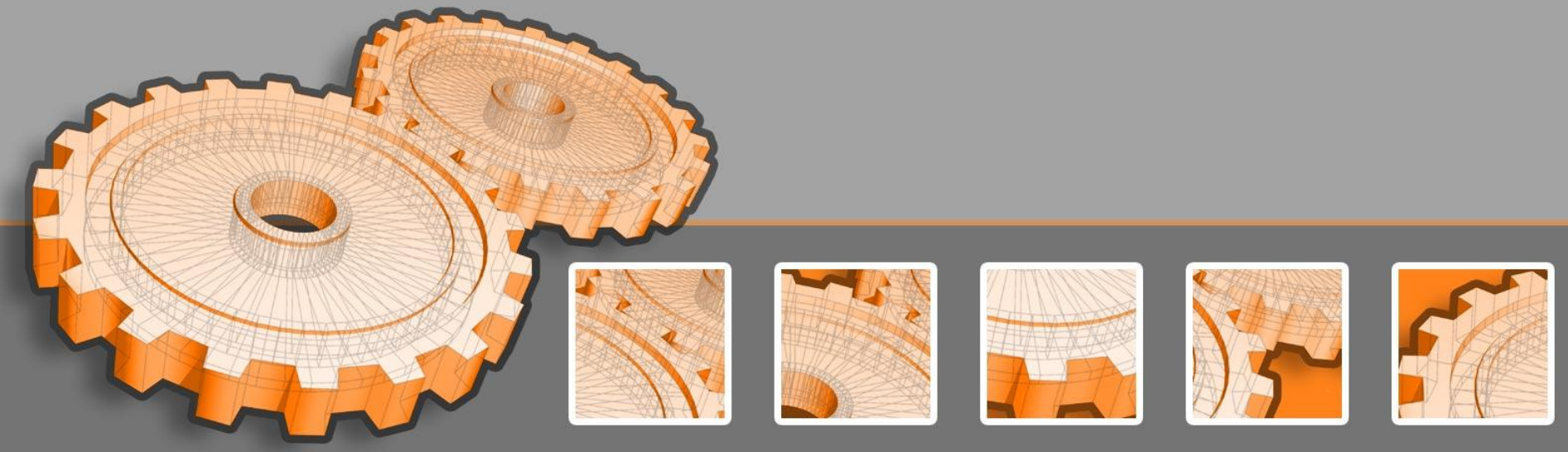
Определение тест кейса

- Тест-кейс содержательно состоит по крайней мере из ожидаемого результата, но, как правило, это комбинация:
  - идеи тест-кейса,
  - сценария,
  - ожидаемого результата.



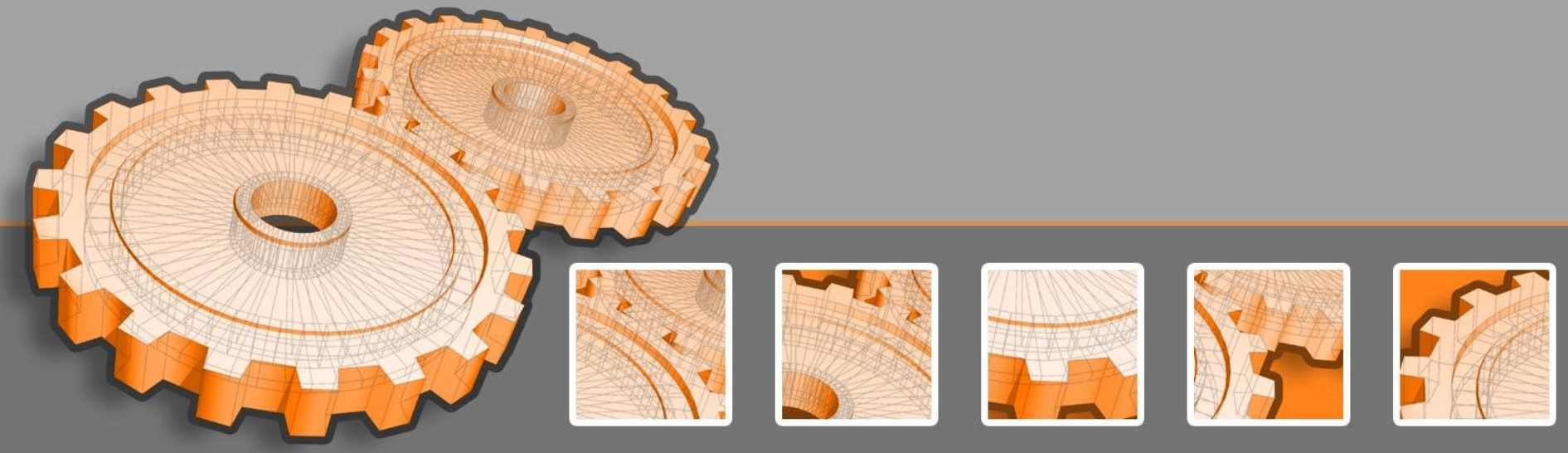
Виды тест кейсов

- Тест кейсы разделяются по ожидаемому результату на позитивные и негативные:
- **Позитивный** - использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию.
  - Пример: 2+2=4
  - Пример: После выполнения указанных операций в системе должен быть зарегистрирован новый пользователь.



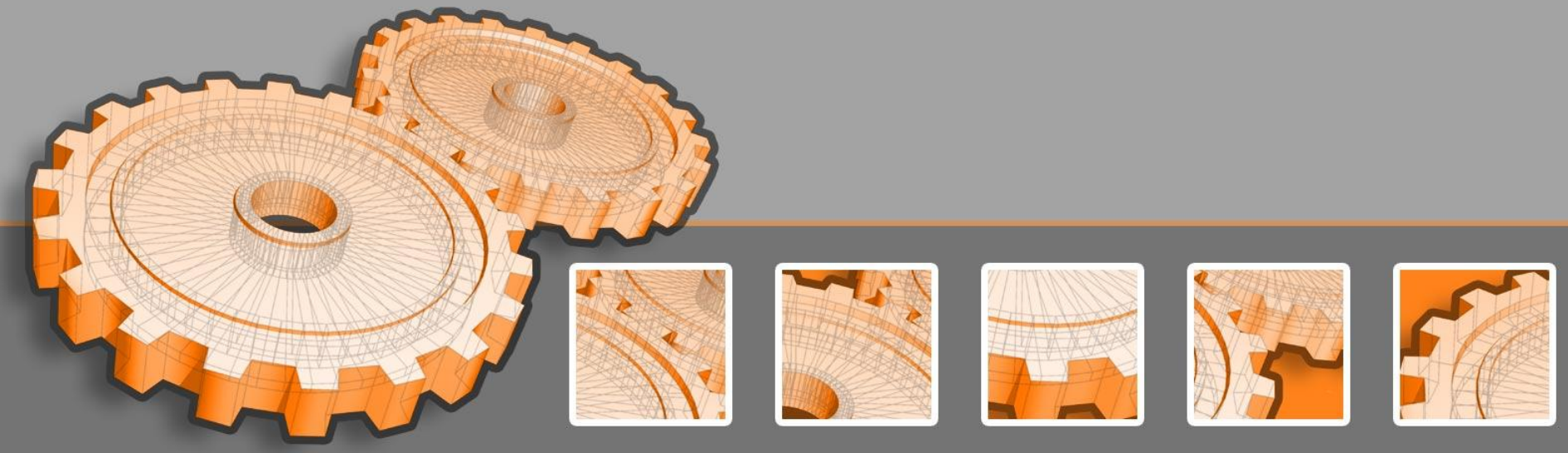
Виды тест кейсов

- Тест кейсы разделяются по ожидаемому результату на позитивные и негативные:
- **Негативный** - оперирует как корректными так и некорректными данными (минимум 1 некорректный параметр) и ставит целью проверку исключительных ситуаций (срабатывание валидаторов), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора.
  - Пример: MAX INT + 1 = ошибка переполнения.
  - Пример: ping 192.168.1.257

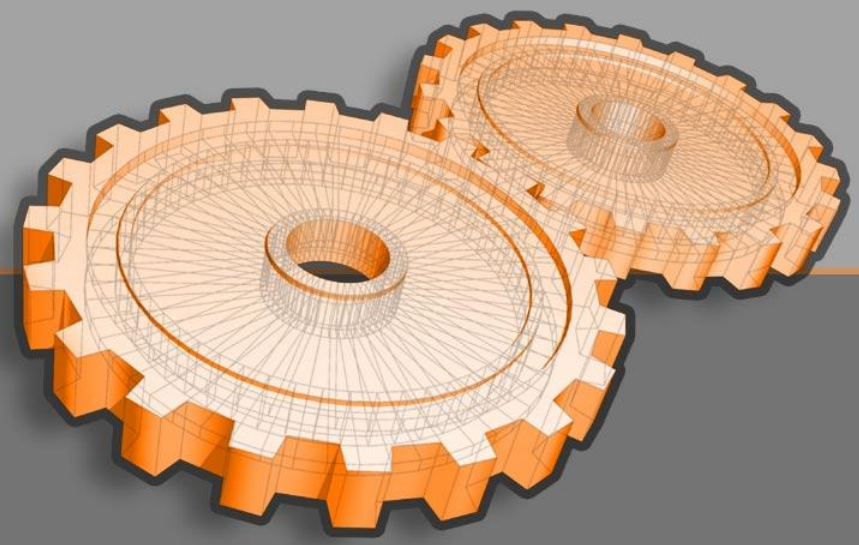


Критерии хорошего тест кейса

- Существуют следующие формальные признаки для оценки качества тест кейсов:
  - Существует обоснованная вероятность выявления тестом ошибки
  - Тест должен быть наилучшим в своей категории
  - Набор тестов не должен быть избыточным
    - ???
  - Он не должен быть слишком простым или слишком сложным
    - ???

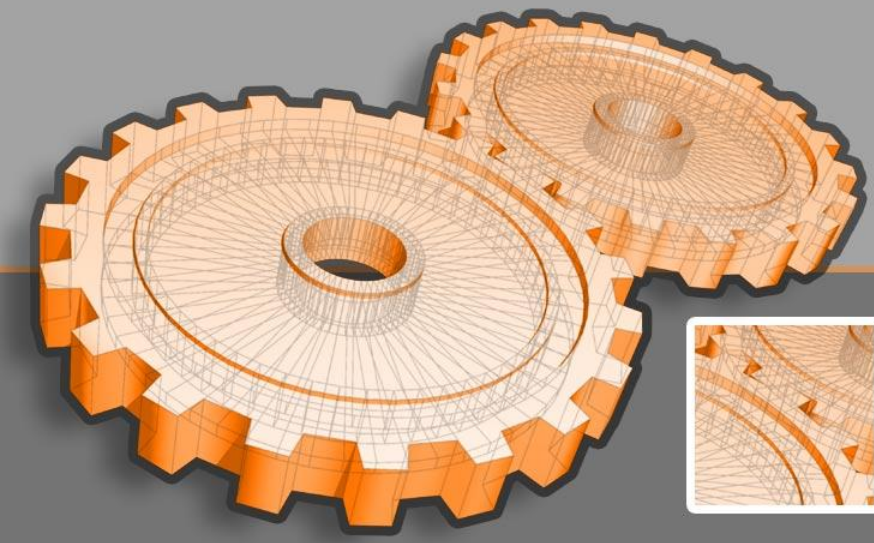


Основные техники тест дизайна



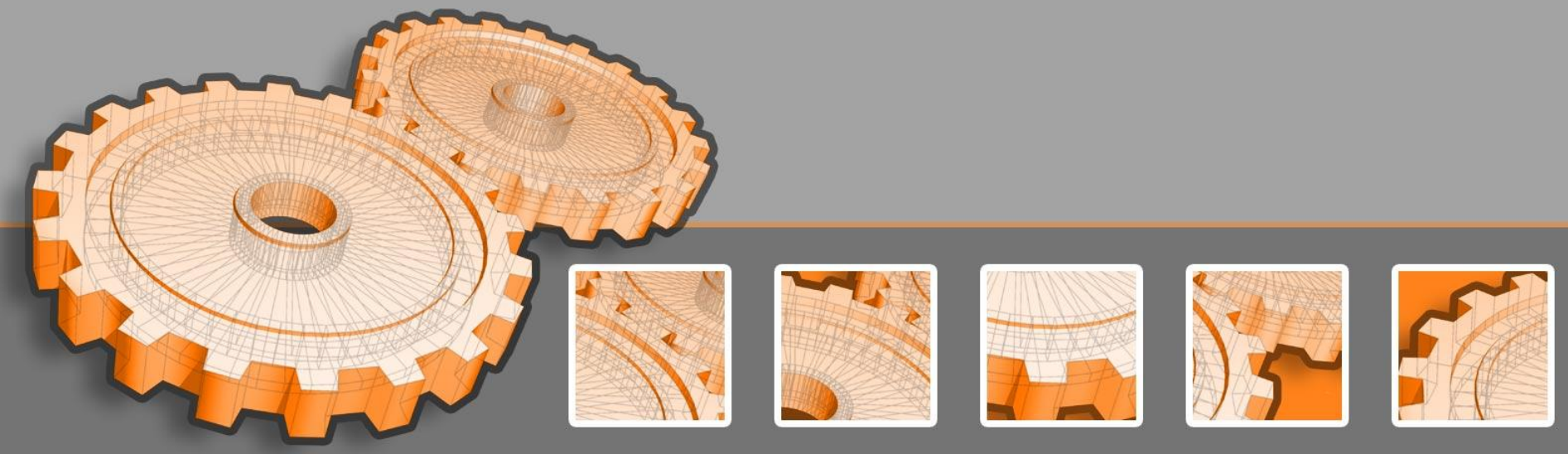
Техники тест дизайна

- Существует несколько основных техник тест дизайна:
  - Эквивалентное Разделение (Equivalence Partitioning - EP).
  - Анализ Граничных Значений (Boundary Value Analysis - BVA).
  - Причина/Следствие (Cause/Effect - CE).
  - Предугадывание ошибки (Error Guessing - EG).
  - Исчерпывающее тестирование (Exhaustive Testing - ET).



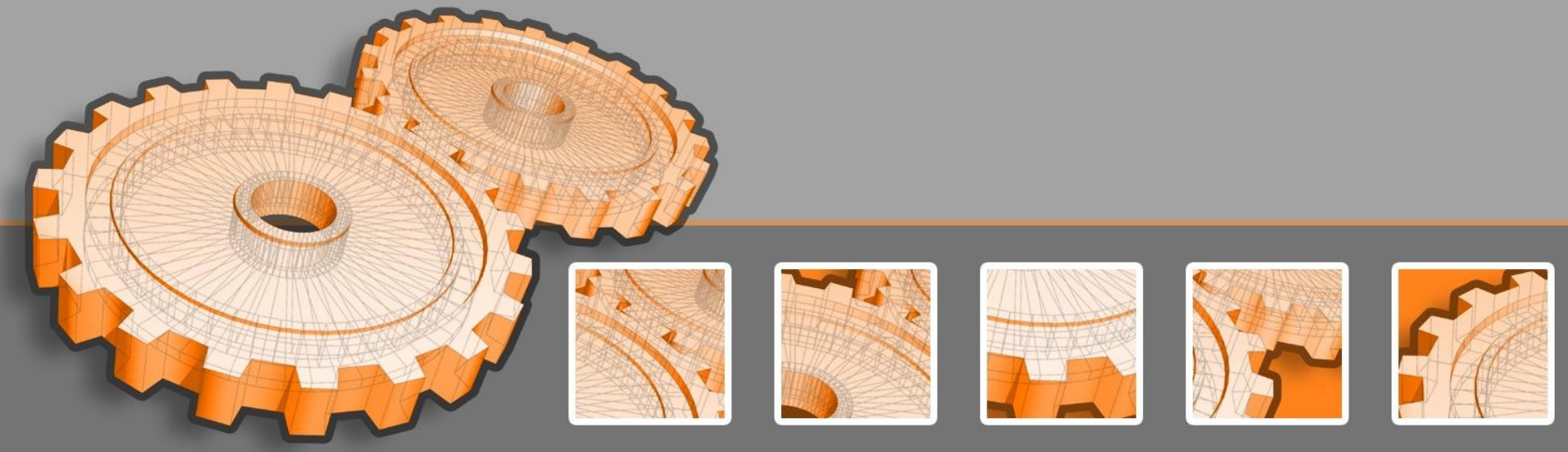
Эквивалентное разделение

- **Эквивалентное Разделение (Equivalence Partitioning - EP).** Рассмотрим данную технику на примере:
  - **Пример:** у вас есть диапазон допустимых значений от 1 до 10, вы должны выбрать одно верное значение внутри интервала, скажем, 5, и одно неверное значение вне интервала - 0.



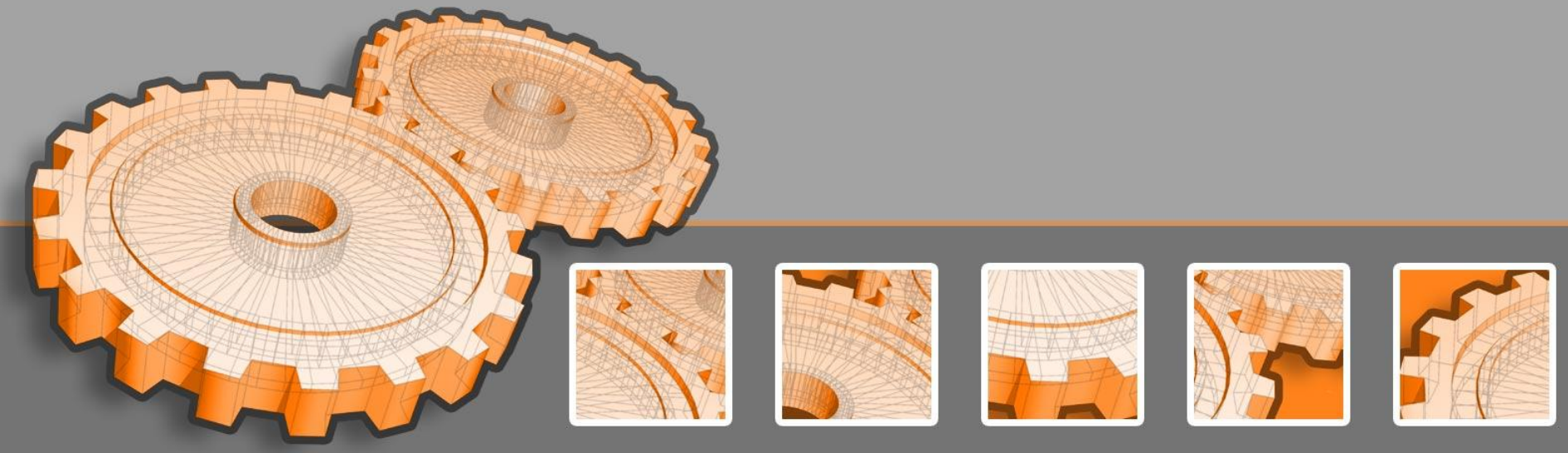
Анализ Граничных Значений

- **Анализ Граничных Значений (Boundary Value Analysis - BVA).** Для данной практики так же рассмотрим пример
  - **Пример:** если для вышеприведенного примера, в качестве значений для позитивного тестирования выберем минимальную и максимальную границы (1 и 10), и значения больше и меньше границ (0 и 11). Анализ Граничных значений может быть применен к полям, записям, файлам, или к любого рода сущностям имеющим ограничения.



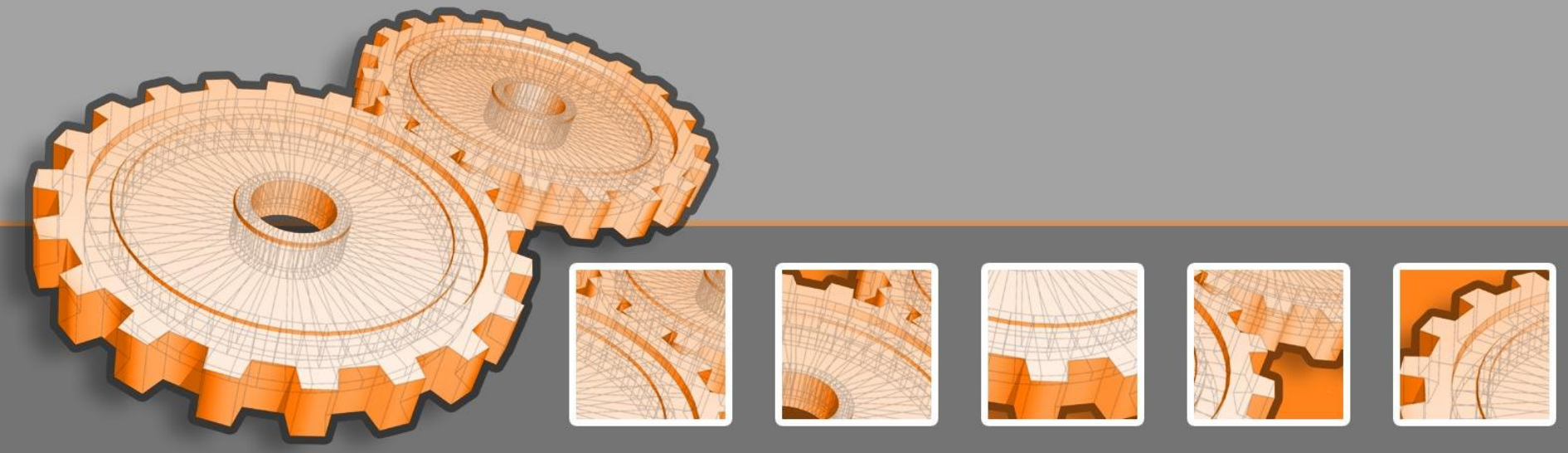
Причина/Следствие

- **Причина/Следствие (Cause/Effect - CE).** Это, как правило, ввод комбинаций условий (причин), для получения ответа от системы (Следствие). Рассмотрим следующий пример:
  - Вы проверяете возможность добавлять клиента, используя определенную экранную форму. Для этого вам необходимо будет ввести несколько полей, таких как "Имя", "Адрес", "Номер Телефона" а затем, нажать кнопку "Добавить" - это "Причина".
  - После нажатия кнопки "Добавить", система добавляет клиента в базу данных и показывает его номер на экране - это "Следствие".



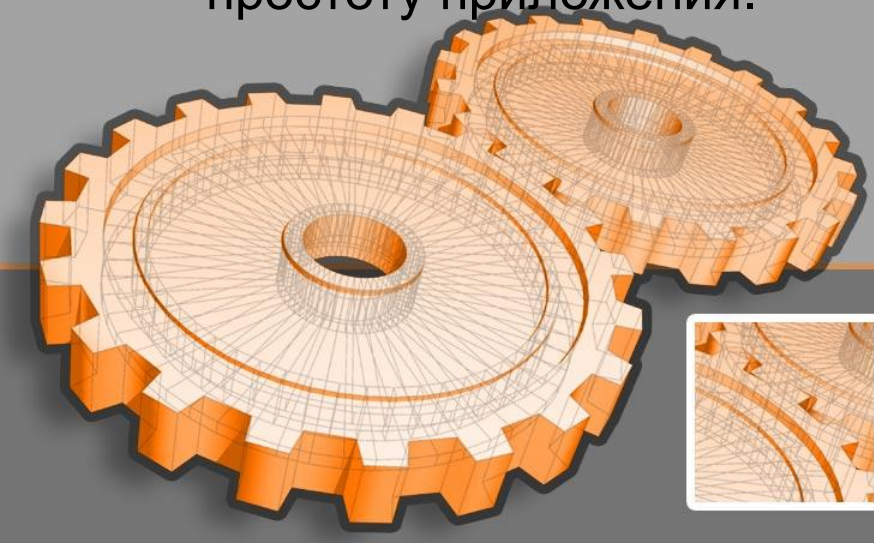
Предугадывание ошибки

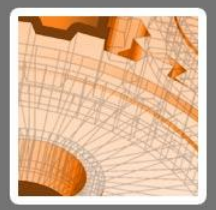
- **Предугадывание ошибки (Error Guessing - EG).** Аналитик использует свои знания системы и способность к интерпретации спецификации на предмет того, чтобы "предугадать" при каких входных условиях система может выдать ошибку.
  - **Пример:** спецификация говорит: "пользователь должен ввести номер счета". Тест аналитик, будет проверять:
    - "Что, если я не введу номер счета другого клиента?",
    - "Что, если я введу номер счета уже удаленного клиента? ", и так далее. Это и есть предугадывание ошибки.

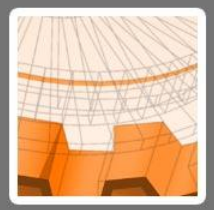


Предугадывание ошибки

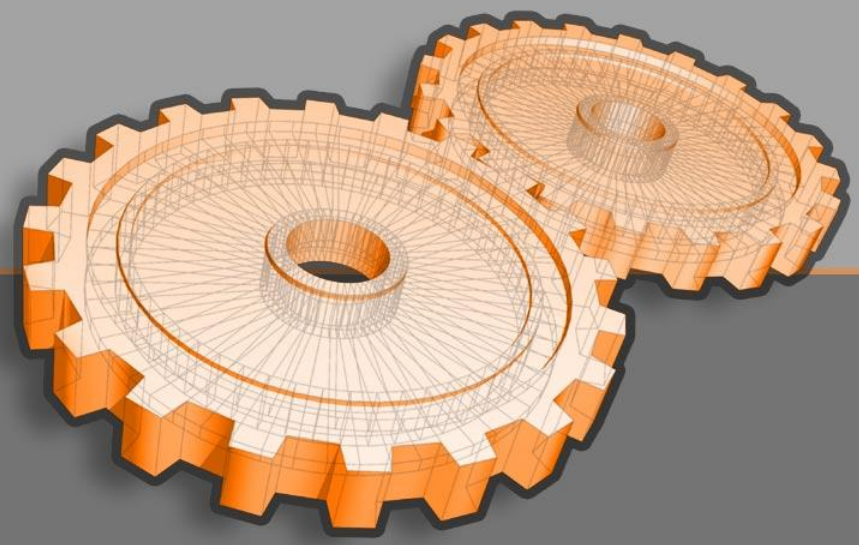
- **Исчерпывающее тестирование** (**Exhaustive Testing - ET**) - это крайний случай.
  - В пределах этой техники вы должны проверить все возможные комбинации входных значений, и в принципе, это должно найти все проблемы. На практике применение этого метода не представляется возможным, из-за огромного количества входных значений.
  - **Пример:** для рассматриваемого нами калькулятора – количество тестов, разрабатываемых для калькулятора может быть огромным, если проверять все возможности, не смотря на простоту приложения.







Тестовое покрытие

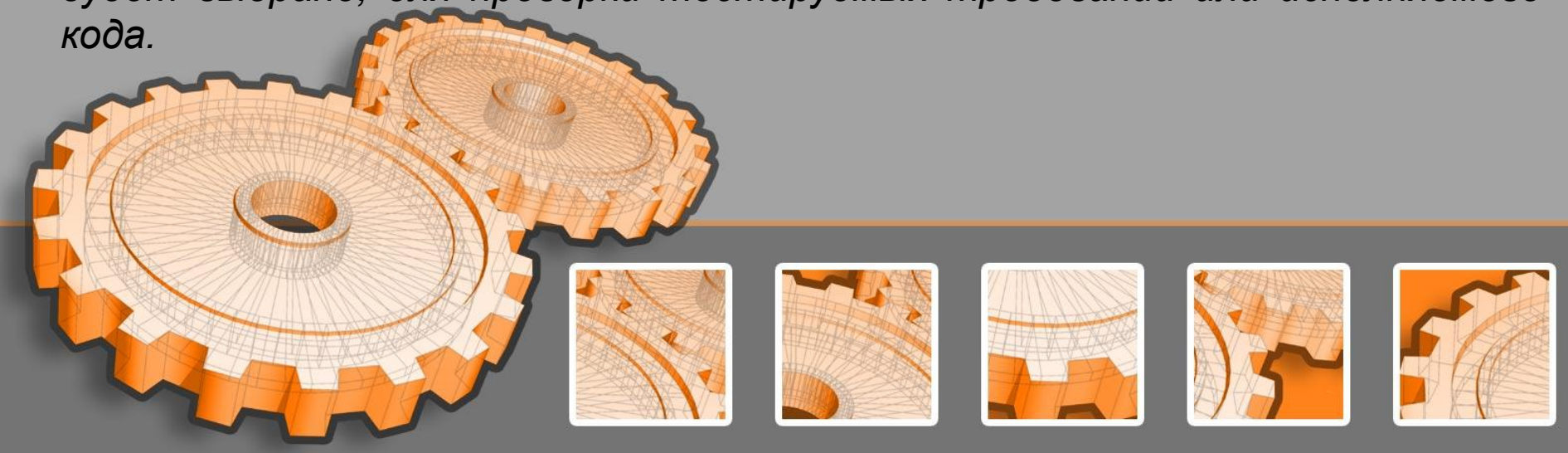


Тестовое покрытие. Определение

Что такое тестовое покрытие?

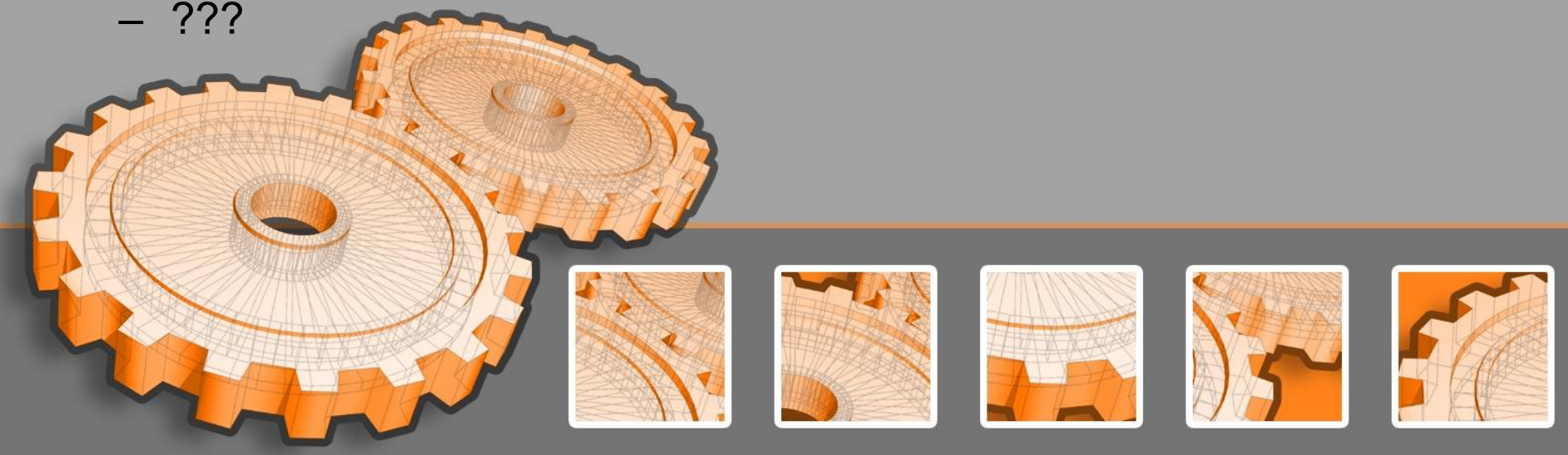
- Это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.
- Если рассматривать тестирование как "проверку соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов", то именно этот конечный набор тестов и будет определять тестовое покрытие:

Чем выше требуемый уровень тестового покрытия, тем больше тестов будет выбрано, для проверки тестируемых требований или исполняемого кода.



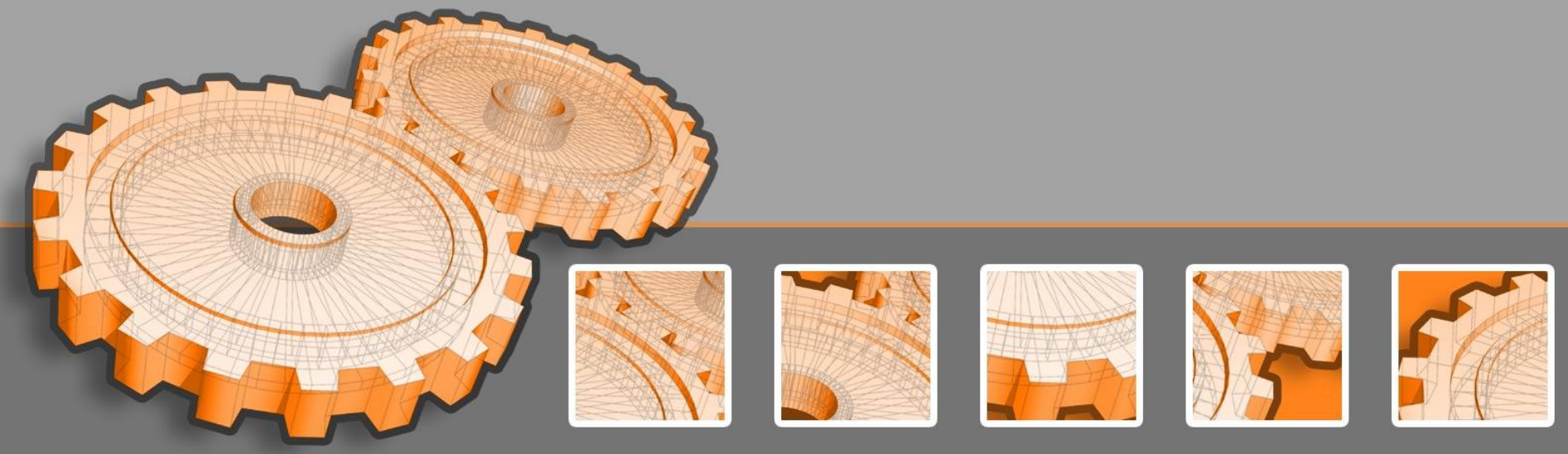
2 подхода к оценке тестового покрытия

- **Покрытие требований (Requirements Coverage)** - оценка покрытия тестами функциональных и нефункциональных требований к продукту путем построения матриц трассировки (traceability matrix).
  - ???
- **Покрытие кода (Code Coverage)** - оценка покрытия исполняемого кода тестами, путем отслеживания непроверенных в процессе тестирования частей программного обеспечения.
  - ???



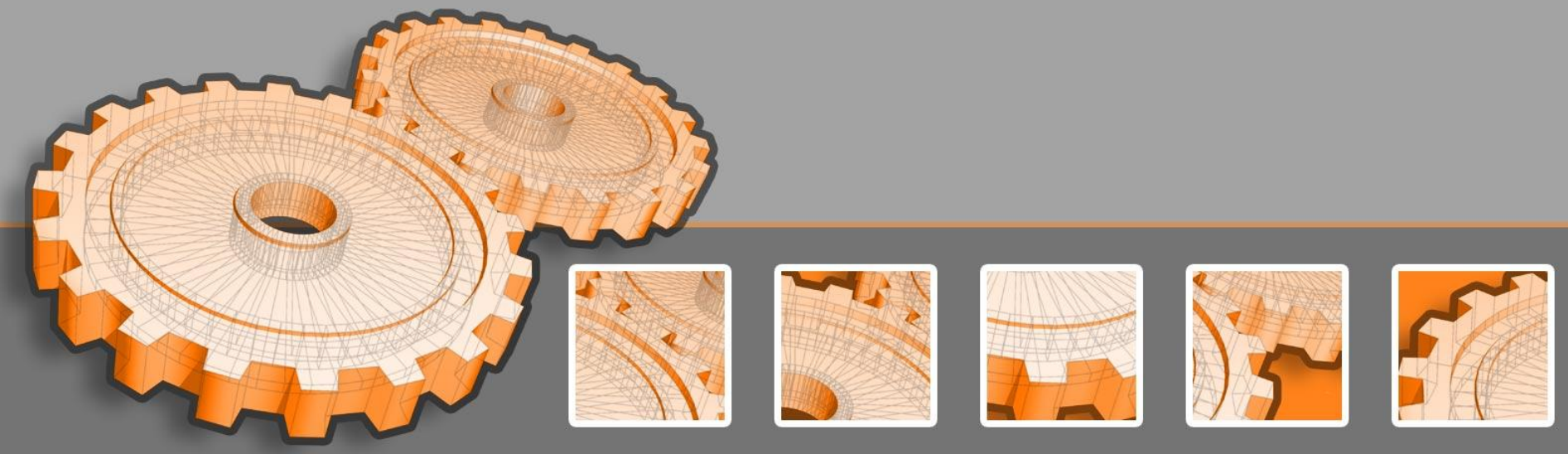
Различия

- Метод покрытия требований сосредоточен на проверке соответствия набора проводимых тестов по отношению к требованиям к продукту
- В тоже время анализ покрытия кода – нацелен на полноту проверки тестами, разработанной части продукта (исходного кода)



Ограничения

- Метод покрытия требований может оставить непроверенными некоторые участки кода, потому что не учитывает конечную реализацию.
- Метод оценки покрытия кода не выявит нереализованные требования, так как работает не с конечным продуктом, а с существующим исходным кодом.



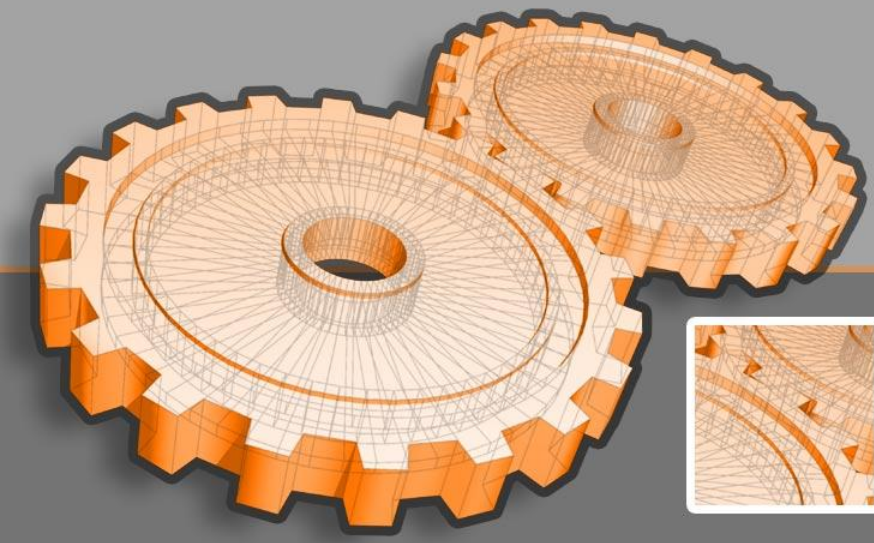
Покрытие требований

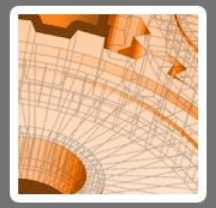
Расчет тестового покрытия относительно требований проводится по формуле:

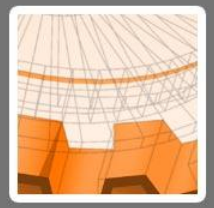
Tcov = (Lcov/Ltotal) * 100%

где:

Tcov - тестовое покрытие
Lcov - количество требований, проверяемых тест кейсами
Ltotal - общее количество требований







Покрытие кода

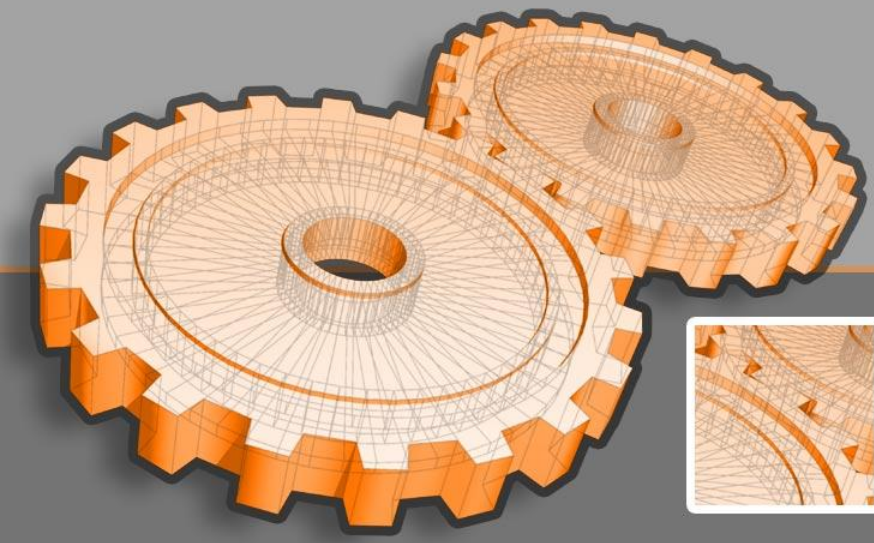
Расчет тестового покрытия относительно исполняемого кода программного обеспечения проводится по формуле:
Tcov = (Ltc/Lcode) * 100%

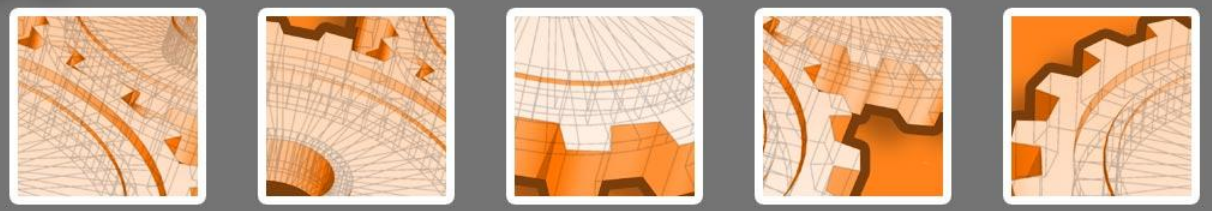
где:

Tcov - тестовое покрытие

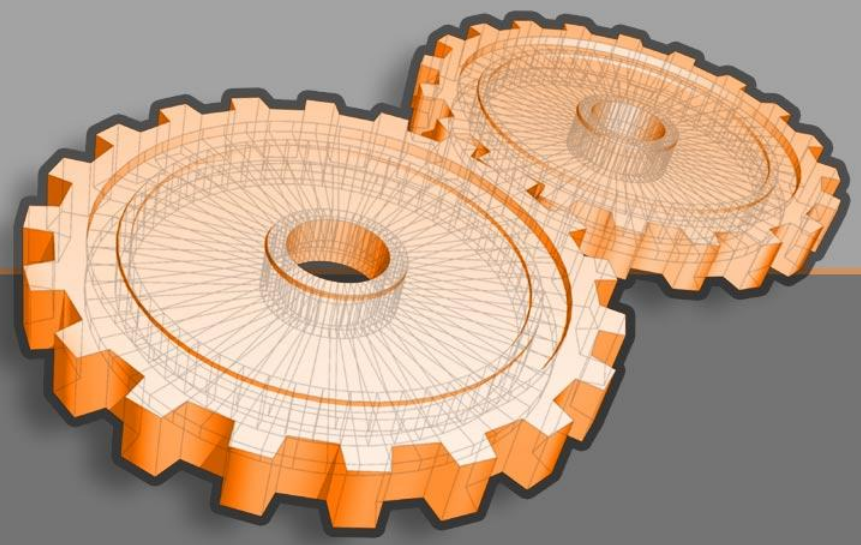
Ltc - кол-ва строк кода, покрытых тестами

Lcode - общее кол-во строк кода.





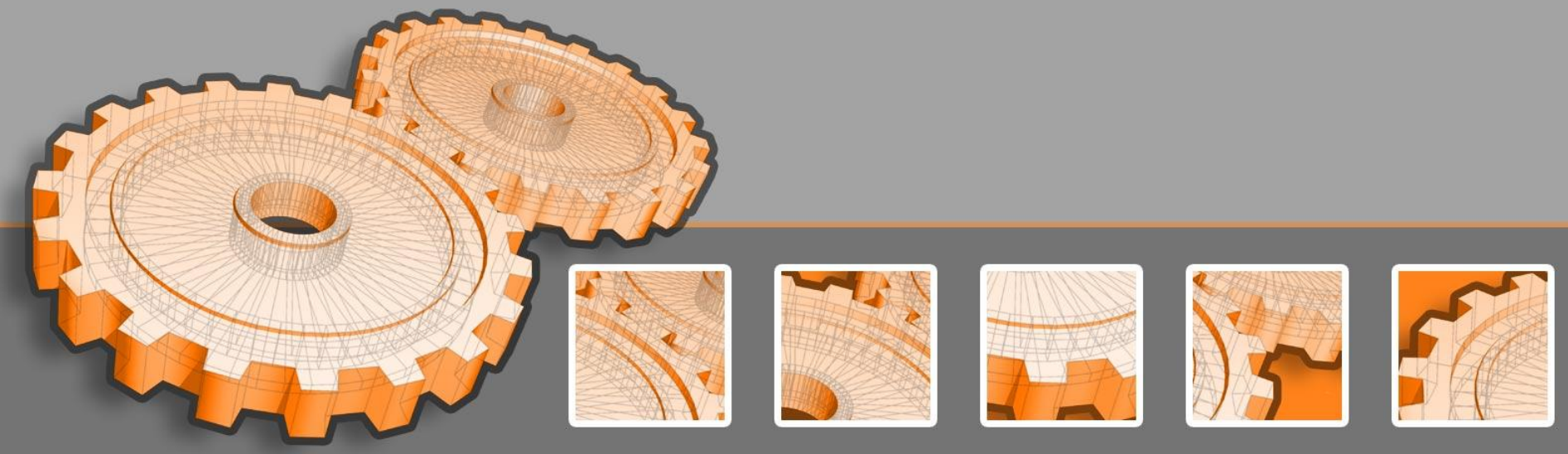
Анализ и приоретизация тестов



Построение процесса тестирования

Анализ результатов проведения тестирования и приоритезация – это одни из наиболее важных вопросов.

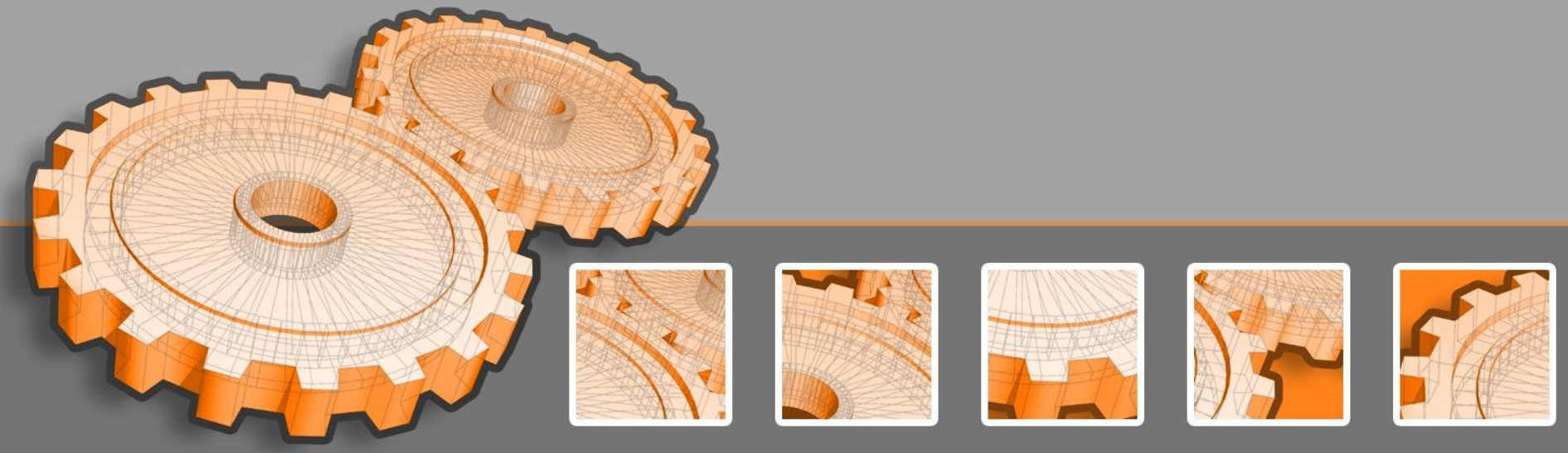
- **Анализ результатов** – позволит не допускать повторения ранее допущенных ошибок.
- **Приоретизация** – может позволить очень существенным образом сэкономить время и ресурсы.



Работаем над примером

Рассмотрим следующий пример. Входные данные:

- На этапе подготовки к сдаче проекта команда тестирования старается протестировать всё.
  - В результате в баг-трекере есть ошибка о том, что программа крашится при нажатии на самую редкоиспользуемую кнопочку 286 раз.
  - Ошибка про странный серый пиксель в нижнем правом углу программы тоже заведена. Команда трудилась в поте лица по ночам и выходным.



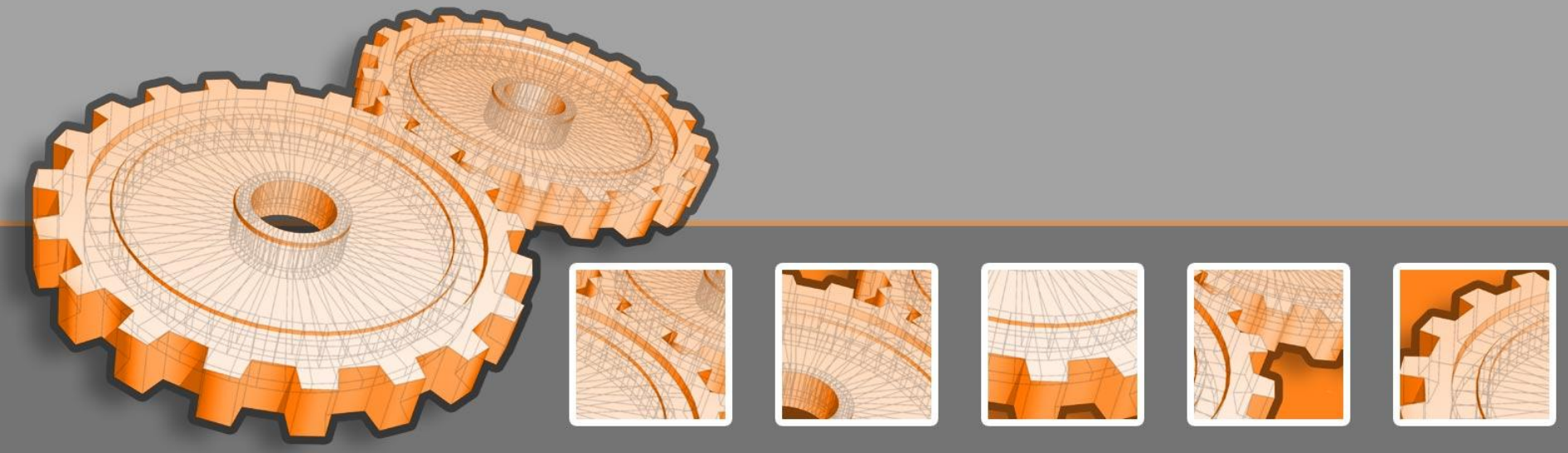
Работаем над примером

- Выходные данные:
  - Релиз.
  - Не работает основной функционал!!!
  - Почему такое возможно?

Что могло послужить причиной несоответствия ожидаемых результатов с реальными результатами?

Думаем и обсуждаем!!!

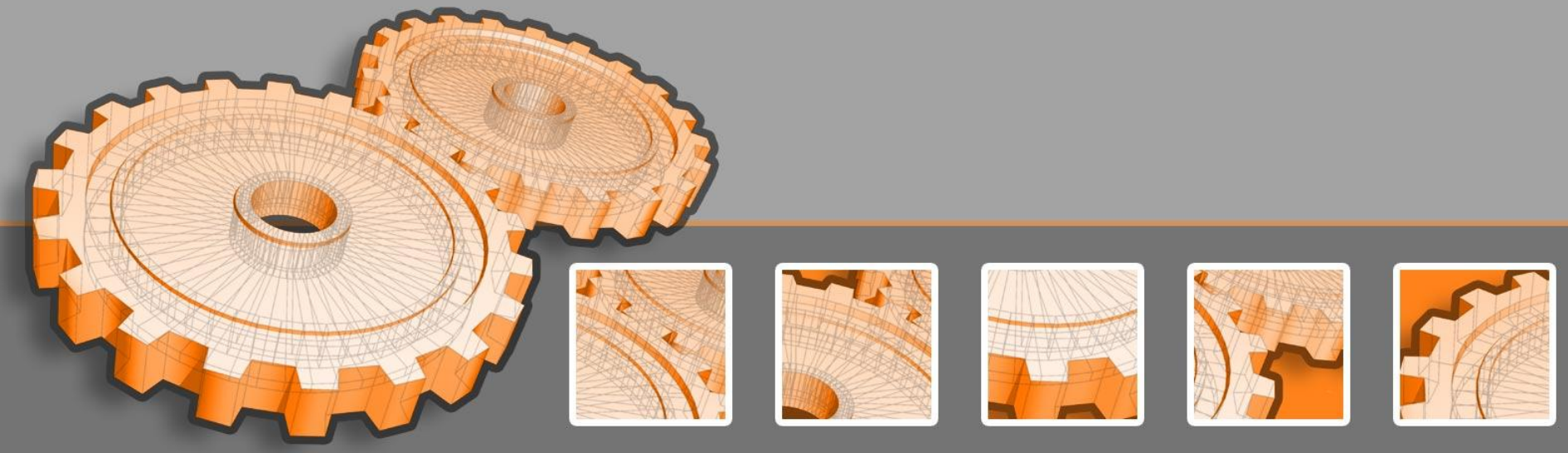
???



Процесс тестирования. Пример

Отсутствие какой-либо приоретизации:

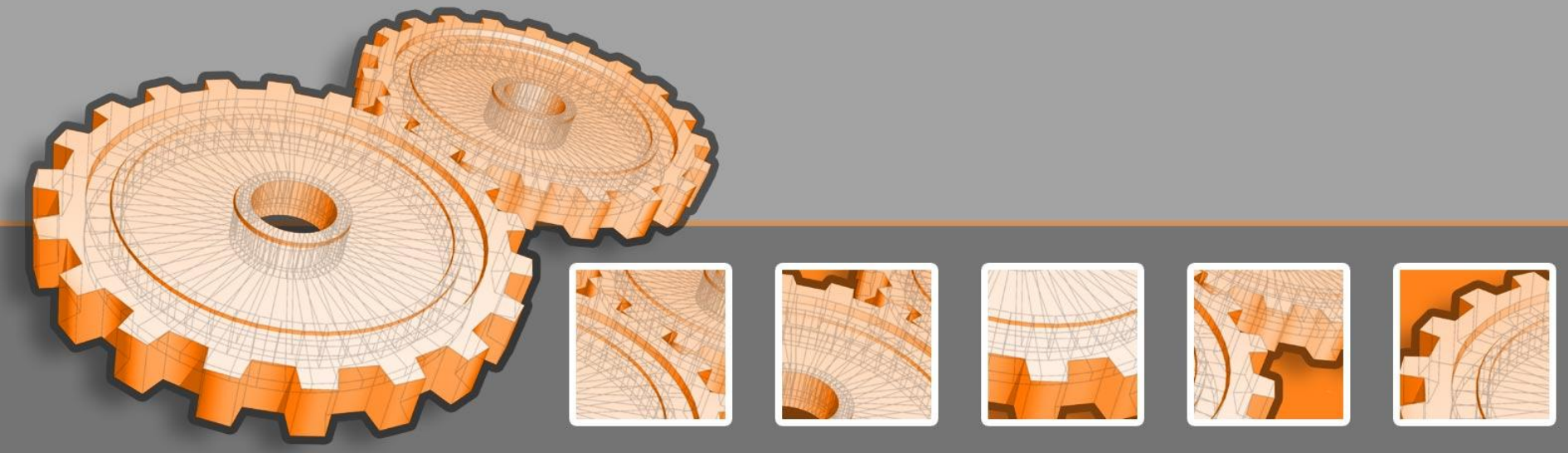
- Заведение всех подряд ошибок мешает разработке.
- Разработчики тратят своё время на исправление минорных ошибок и вносят новые, зачастую более серьёзные.
- Потраченное на мелочи время не дало возможности проверить более серьёзные пользовательские сценарии и найти более критичные дефекты.



Процесс тестирования. Пример

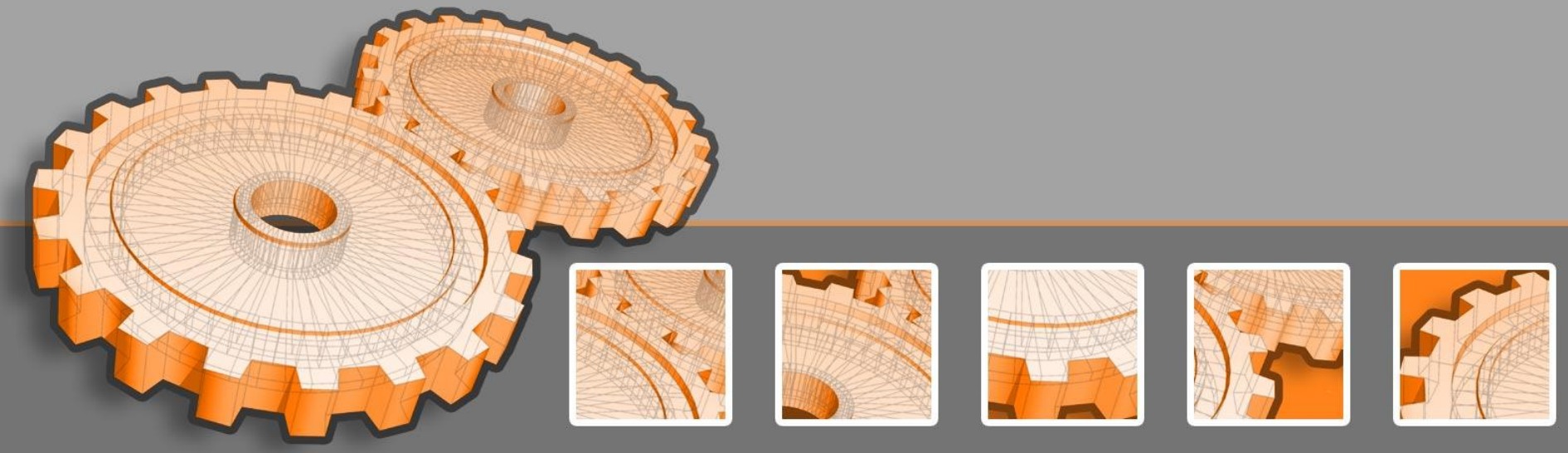
Отсутствие какой-либо приоретизации:

- Обратная связь по статусу сборки предоставлялась разработчикам с запозданием: вместо критичных дефектов непрерывно сыпались миноры.
- Проектный паттерн «дохлая рыба» сыграл своё дело: все участники команды прекрасно понимали, что протестировать всё нельзя, и это не могло не сказаться на качестве работы. А реалистичных целей им никто не поставил…

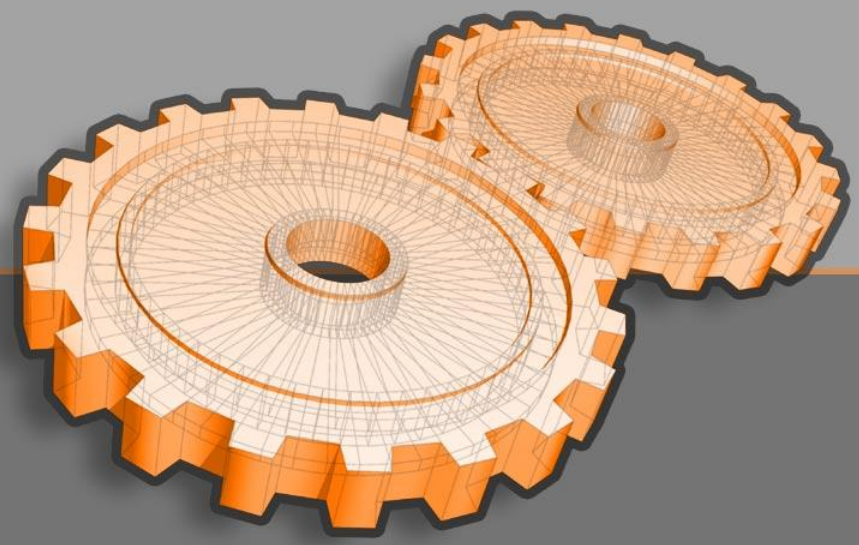


Процесс тестирования. Пример

- Что Вы можете предложить сделать по-другому?
  - ???
- Что Вы предлагаете поменять в первую очередь?
  - ???



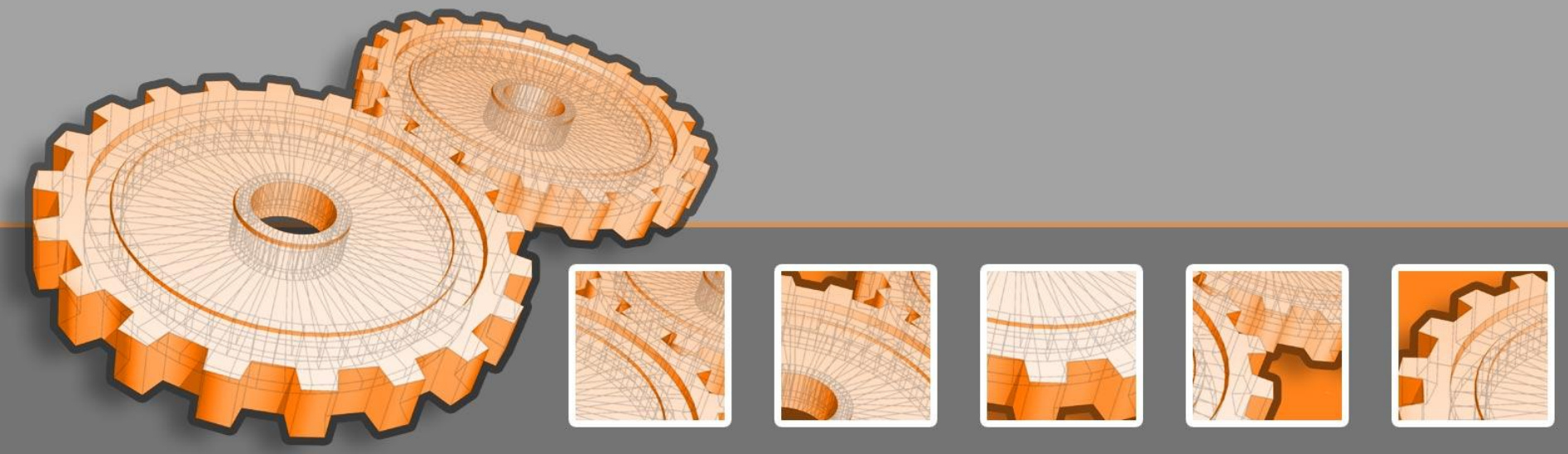
Заключение



Заключение

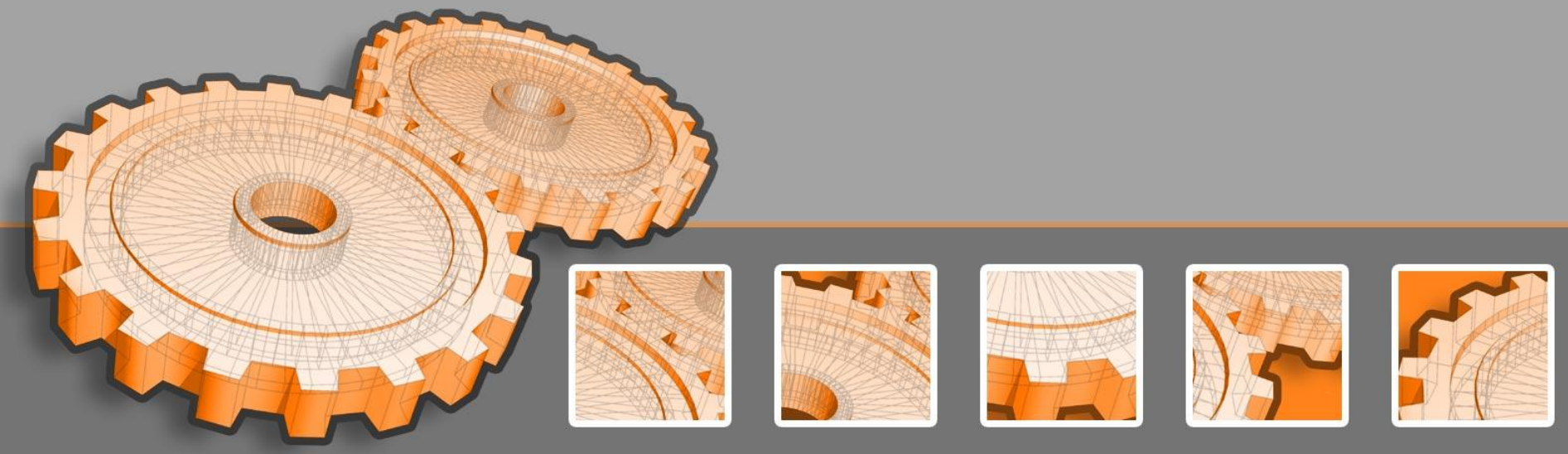
На лекции были рассмотрены следующие вопросы:

- Определение понятия “Тест дизайн”.
- Техники тест дизайна.
- Тестовое покрытие.
- Анализ и приоретизация тестов.



Заключение

Q & A



Контактные данные

E-Mail: ilinikhdu@gmail.com
Skype: ilinikhdmitry
Tel.: +7 917 312 48 73