Повторение урока№5

- Что такое требование?
- Как тестировать требования?
- Какими должны быть требования?

Тест дизайн.

Тест дизайн – процесс проектирования тестов.

Зачем нам нужен тест дизайн?

1. Придумать тесты, которые обнаружат наиболее серьезные ошибки продукта. Да, мы можем придумывать тесты, которые находят несерьезные ошибки, но тогда тестирование будет неэффективным.

2. Минимизировать количество тестов, необходимых для нахождения большинства серьезных ошибок. Мы может придумать столько тестов, сколько не в состоянии будем выполнить. Поэтому перед разработчиками тестов всегда стоит задача – сохранить эффективность тестов (то есть их способность обнаруживать серьезные ошибки) без увеличения их числа.

* Навыки тест дизайнера

1. Умение разделять систему на составляющие (делать декомпозицию). То есть, нужно уметь видеть систему как целое, но и уметь раскладывать ее на составные части. Это очень полезный навык для проведения функционального тестирования, где проверяется каждая фича продукта.

2. Умение собирать и анализировать требования к продукту. Если нет формально описанных требований – нужно уметь их собирать у разработчиков, у аналитиков, у пользователей.

3. Умение расставлять приоритеты. Тест дизайнер должен уметь отличать более важное от менее важного, и расставлять приоритеты тестирования.

4. Умение формулировать свои мысли (письменно и устно). Это умение важно для тестировщика в принципе. Тест дизайнеру оно здорово поможет при создании тест кейсов.

5. Знание техник тест дизайна. Основные техники мы коротко обсудим в этом сообщении.

6. Умение применять их на практике.

Техники тест дизайна

Эквивалентное Разделение (Equivalence Partitioning)

Анализ Граничных Значений (Boundary Value Analysis)

Предугадывание ошибки (Error Guessing)

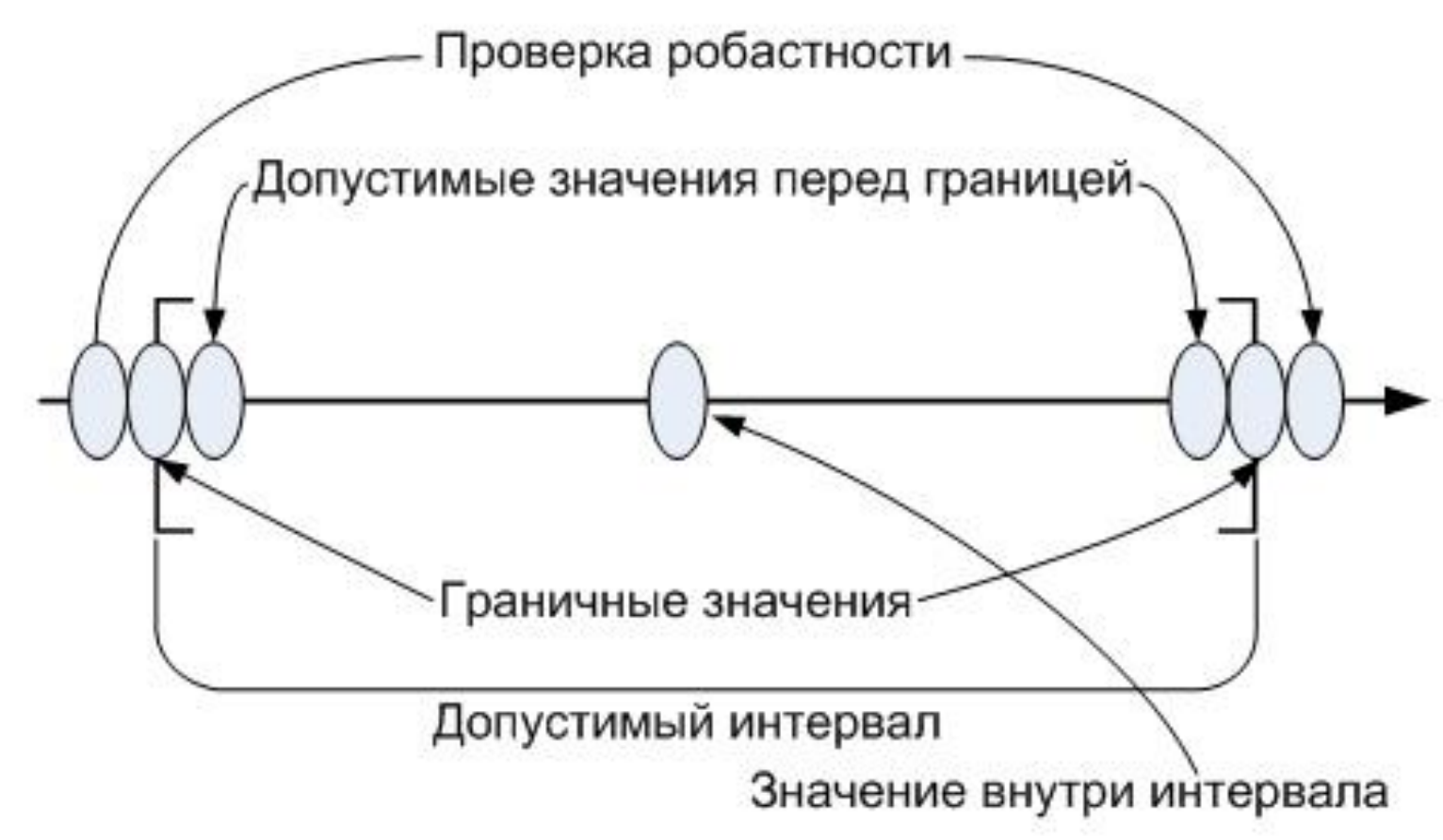
Причина / Следствие (Cause/Effect)

Исчерпывающее тестирование

Попарное тестирование (Pairwise testing)

* Эквивалентное Разделение (Equivalence Partitioning)

* Анализ Граничных Значений (Boundary Value Analysis)



Предугадывание ошибки (Error Guessing)

ПРЕДУГАДЫВАНИЕ ОШИБКИ (ERROR GUESSING - EG).

Это когда тест аналитик использует свои знания системы и способность к интерпретации спецификации на предмет того, чтобы "предугадать" при каких входных условиях система может выдать ошибку. Например, спецификация говорит: "пользователь должен ввести код". Тест аналитик, будет думать: "Что, если я не введу код?", "Что, если я введу неправильный код? ", и так далее. Это и есть предугадывание ошибки.

* Причина / Следствие (Cause/Effect)

Это, как правило, ввод комбинаций условий (причин), для получения ответа от системы (Следствие).

Например, вы проверяете возможность добавлять клиента, используя определенную форму.

Для этого вам необходимо будет ввести несколько полей, таких как "Имя", "Адрес", "Номер Телефона" а затем, нажать кнопку "Добавить" - эта "Причина".

После нажатия кнопки "Добавить", система добавляет клиента в базу данных и показывает его номер на экране - это "Следствие".

Исчерпывающее тестирование

ИСЧЕРПЫВАЮЩЕЕ ТЕСТИРОВАНИЕ (EXHAUSTIVE TESTING - ET)

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.

Попарное тестирование (Pairwise testing)

Допустим, какое-то значений (налог) для человека рассчитывается на основании его пола, возраста и наличия детей - получаем три входных параметра, для каждого из которых для тестов выбираем каким-то образом значения. Например: пол - мужской или женский; возраст - до 25, от 25 до 60, более 60; наличие детей - да или нет. Для проверки правильности расчётов можно, конечно, перебрать все комбинации значений всех параметров:

Попарное тестирование (Pairwise testing)

№	пол	возраст	дети
1	мужчина	до 25	детей нет
2	женщина	до 25	детей нет
3	мужчина	25-60	детей нет
4	женщина	25-60	детей нет
5	мужчина	старше 60	детей нет
6	женщина	старше 60	детей нет
7	мужчина	до 25	дети есть
8	женщина	до 25	дети есть
9	мужчина	25-60	дети есть
10	женщина	25-60	дети есть
11	мужчина	старше 60	дети есть
12	женщина	старше 60	дети есть

Попарное тестирование (Pairwise testing)

А можно решить, что нам не нужны сочетания значений всех параметров со всеми, а мы хотим только убедиться, что мы проверим все уникальные пары значений параметров. Т.е., например, с точки зрения параметров пола и возраста мы хотим убедиться, что мы точно проверим мужчину до 25, мужчину между 25 и 60, мужчину после 60, а также женщину до 25, женщину между 25 и 60, ну и женщину после 60. И точно так же для всех остальных пар параметров. И таким образом, мы можем получить гораздо меньше наборов значений (в них есть все пары значений, правда некоторые дважды):

* Попарное тестирование (Pairwise testing)

№	пол	возраст	дети
1	мужчина	до 25	детей нет
2	женщина	до 25	дети есть
3	мужчина	25-60	дети есть
4	женщина	25-60	детей нет
5	мужчина	старше 60	детей нет
6	женщина	старше 60	дети есть

Такой подход примерно и составляет суть техники pairwise testing - мы не проверяем все сочетания всех значений, но проверяем все пары значений.

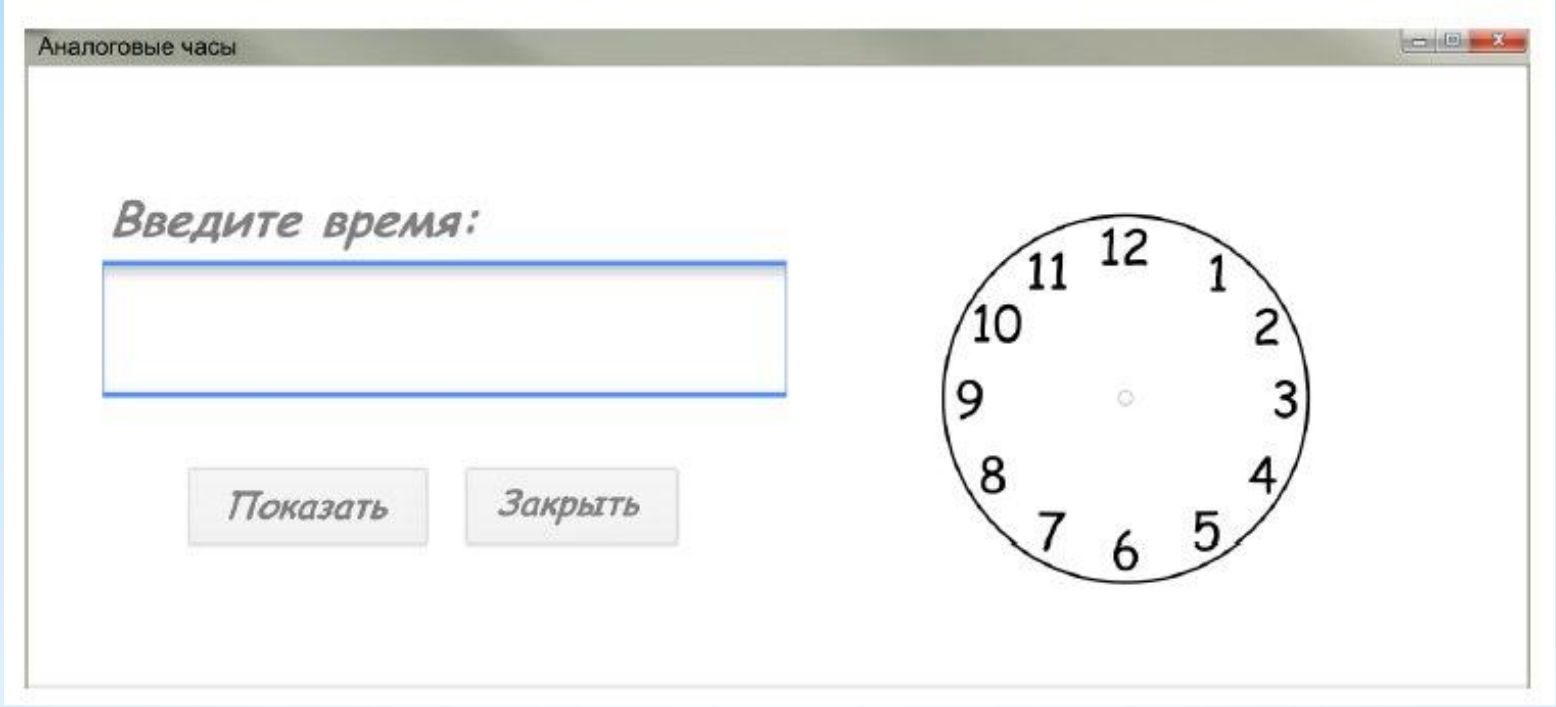
* Практическое задание:

Придумать сценарии тестирования на основе основных техник тестирования:

Эквивалентное Разделение (Equivalence Partitioning)

Анализ Граничных Значений (Boundary Value Analysis)

Причина / Следствие (Cause/Effect)



Дана программа "Аналоговые часы" для обучения детей определять время.
Программа открывается в оконном режиме, состоит из:
* Поля ввода "Введите время";
* Поля вывода в виде циферблата аналоговых часов;
* Кнопка "Показать"
* Кнопка "Закрыть"
Принцип работы программы заключается во введении времени в 24-часовом формате и нажатии на кнопку "Показать".

Нужно:
Протестировать программу (имитировать работу пользователя) на работоспособность минимальным(!) количеством проверок.