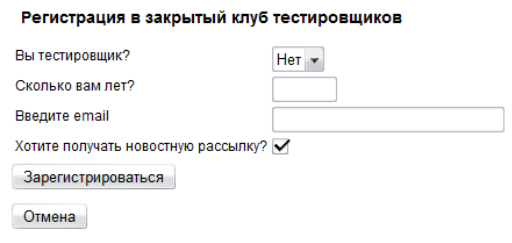
Лекция.Проектирование тестов. Исследование продукта.

# Знакомимся с продуктом

Сначала необходимо исследовать продукт. Подходы есть для этого разные: сначала изучаем продукт, т.е. смотрим на него со стороны пользователя, после изучаем документацию, либо сначала изучаем документацию, после изучаем продукт. Чаще используется второй вариант.

Познакомимся с тестовым продуктом. Есть некое тестовое приложение «Клуб тестировщиков». Оно позволяет произвести регистрацию в закрытый клуб тестировщиков. Ниже отображается скриншот формы регистрации.



На скриншоте представлены такие поля:

«Вы тестировщик?» - выпадающий список с возможностью выбрать ответ Да или Нет

«Сколько вам лет?» - поле ввода для возраста

«Введите email» - поле для ввода эл.адреса

«Хотите получать новостную рассылку?» - установлена галочка, еще называют их флажком, или чек-боксом (от англ.check box)

Кнопки:

«Зарегистрироваться» - при заполнение корректными данными пользователь регистрируется в клубе; если был установлен чек-бокс, то ему будет отправляться рассылка

«Отмена» - регистрация не происходит

**Регистрация происходит по следующим правилам**:

• Пользователь регистрируется, если

– Совершеннолетний

– Тестировщик

– Корректный е-мейл

• Получает письмо, если

– Успешно зарегистрирован в клубе

– Поставил галочку «хочу получать»

# Процесс тестирования и проектирования тестов

Модели тестирования бывают разные. В каждой команде с учетом целей все процессы разные. Глобально моделей тестирования две: Скриптовое и исследовательское тестирование.

В модели **скриптового тестирования** есть несколько ролей, которые участвуют в подготовке и тестировании:

* Тест-аналитик создаёт тестовый набор. Он определяет ЧТО тестировать: выявляет сущности, выбирает значения, определяет подход, комбинирует проверки.
* Тест-дизайнер. Определяет КАК тестировать: оптимизирует шаги проверок, документирует тест-кейсы, чек-листы, определяет способ проверок.
* Тестировщик по ним проводит тестирование и регистрирует найденные несоответствия

***Тесовый набор -****это набор тест кейсов, которые объединены тем, что относятся к одному тестируемому модулю, функциональности, приоритету или одному типу тестирования. Каждый тестовый набор состоит из более чем одного тест кейса и зачастую выполняется всей «пачкой» в процессе тестирования.*

Тестировщик и тест-дизайнер, и тест-аналитик могут быть совмещены в одном лице, так и в разных.

Такая модель используется, когда продукт сложный, есть необходимость документировать тесты.

**Плюсы такой модели**: высокая планируемость и предсказуемость, качественная отчетность.

**Минусы**: чек-листы\тест-кейсы могут со временем стать неактуальными, присутствует «эффект пестицида», общий процесс негибкий, работа монотонная, может наскучить.

**Исследовательское тестирование** – тестировщик заранее не знает, как будет тестировать, какие тесты необходимо выполнить, тестирует «налету». Тестировщик изучает\исследует продукт уже в процессе: тестирует, исследует, узнает новое, генерирует идеи, тестирует, исследует и т.д.

**Плюсы:** общий процесс работы гибкий, адаптивный к изменениям, задачи интересные

**Минусы:** сложности в отчетности, невозможно планировать, нет гарантий, общий результат зависит от квалификации тестировщика

**Хаотическое тестирование** – тестировщик просто нажимает кнопочки, нажимает кнопочки…А зачем? Нет никакого проектирования тестов, нет анализа. Цель тестирования не понятна.

Проектирование тестов необходимо выполнять всегда, даже если мы их не планируем документировать. Здесь важно на начальных этапах, когда еще у тестировщика мало опыта, всегда документировать тесты.

Во время проектирования мы определяем ЧТО будет тестировать, т.е. проводим тест-анализ, в результате создаем наборы сценариев, и КАК мы будем тестировать продукт (тест-дизайн) – описываем тест-кейсы\чек-листы.

При проектировании тестов мы должны ответить на ряд вопросов:

• Как тестировать? Важно понять с какими значениями будем проверять, с какой последовательностью.

• Сколько тестов мы готовы выделить на тестирование? Всегда необходимо искать компромисс между качеством тестирования и скоростью тестирования. Например, при проверке поля «сколько вам лет?» мы не можем себе позволить проверять каждую цифру возраста, в реальной жизни никогда не хватает времени на тесты, поэтому важно находить компромисс.

• Как протестировать БЫСТРО?

• Как протестировать ПОЛНОЦЕННО?

Как происходит сам процесс проектирования тестов:



При исследовании продукта необходимо понять, что хочет пользователь и какие у него цели, намерения, как должен работать тестируемый продукт. Можно описать это общими словами, а после структурировать в виде таблицы, майнд-карт или иначе.При этом необходимо выделить ключевые функции, которые предоставляет тестируемый продукт, определить какие входные и выходные параметры возможны (например, для поля «Вы тестировщик?» есть возможные значения только ДА и НЕТ), какие последовательности действий необходимо выполнить.

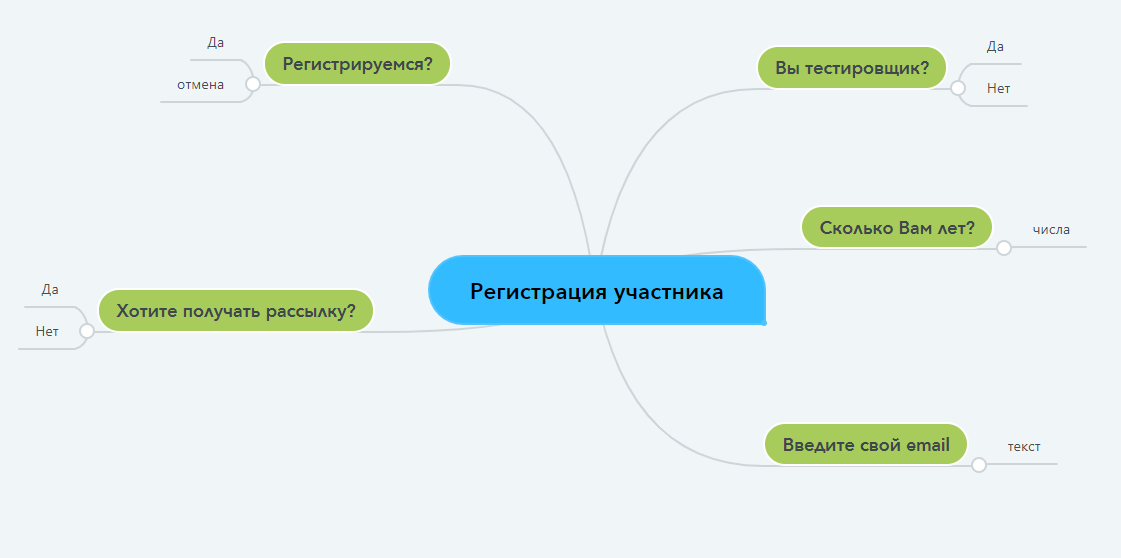
После проверяем ЧТО мы можем протестировать, с какими значениями, параметрами и комбинациями.

После выбираем, что будет протестировано, .т.е определяем, что из этого большого объема проверок мы действительно сможем проверить, выбираем тесты, которые будем проходить и нет.

# Исследование возможных значений. Классы и границы

Для начала нам нужна карта, чтобы определить, что будем тестировать.

(данная карта реализована с помощью <https://www.mindmeister.com/>)



После составления основного списка параметров, разберем каждый из них.

## Классы эквивалентности и граничные условия

1)



Что будем проверять?

Как сделать набор не избыточным?

Как сделать набор достаточным?

У нас есть условия, при которых происходит регистрация: Пользователь регистрируется, если он Совершеннолетний

***Вопрос: Какие проверки по Вашему мнению необходимо сделать?***

***Ответ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Рассмотрим пример:

На дороге стоит мост с ограничения по высоте 4,5м.

Необходимо протестировать мост.

Вопрос: Какие грузовики смогут пройти под ним, а какие нет?

Делим все грузовики на **классы**:

• Все грузовики меньше 4,5 м по высоте – ЭКВИВАЛЕНТНЫ

• Все грузовики выше 4,5 м по высоте – ЭКВИВАЛЕНТНЫ

При этом у нас есть **граничные условия** (границы):

• Если 4,5 м пройдёт, то и 4,49 пройдёт? (Да)

• Если 4,51 не пройдёт – то и 5,50 не пройдёт? (Да)

• А если 3 пройдёт – то о чём это говорит? (Ни о чем)

Делаем вывод, что двух значений достаточно, чтобы проверить оба класса эквивалентности!

Этот метод тестирования основан на **классах эквивалентности и граничных условиях**.

**Класс эквивалентности** — набор данных, обрабатываемый одинаковым образом и приводящий к одинаковому результату.

**Граничное условие** — значение, находящееся на границе классов эквивалентности.

**Эквивалентное** разбиение применимо на всех уровнях тестирования, может быть использовано с целью покрытия входных и выходных данных. Оно может применяться при ручном вводе данных, при передаче данных через интерфейсы в систему или при проверке параметров интерфейсов в интеграционном тестировании.

Примерный **алгоритм использования** **техники классы эквивалентности** следующий:

1. Определить классы эквивалентности. Это главный шаг техники. От него во многом зависит эффективность её применения.

2. Выбрать одного представителя от каждого класса. На этом шаге из каждого эквивалентного набора тестов выбирают один тест.

3. Выполнить тесты. На этом шаге выполняют тесты от каждого класса эквивалентности.

Вернемся к примеру с возрастом на форме регистрации.

Поле «Сколько Вам лет?» :

• Какие значения эквивалентны?

• Какие граничные условия?

0………1-ый класс.…………...18 лет ………2-ой класс…………...неизвестный максимум

Чтобы не гадать какое максимальное значение будет, мы можем узнать это информацию у разработчика или у аналитика. Ни в коем случае не додумываем самостоятельно – это источник пропущенных ошибок!

Например, 255 - сказал разработчик и подтвердил аналитик. Т.е. 255 –это максимальное граничное условие, 256 – определим в след класс, т.к больше максимального значения.

Примерный **алгоритм использования техники анализа граничных условий** состоит из четырёх основных шагов:

1. Необходимо выделить классы эквивалентности. От правильности разбиения на классы эквивалентности зависит эффективность тестов граничных значений.

2. Далее нужно определить граничные значения этих классов.

3. Затем следует понять, к какому классу будет относиться каждая граница.

4. Наконец, для каждой границы нужно провести тесты по проверке значения до границы, на границе и сразу после границы.

Вернемся к примеру. Достаточно ли нам этих проверок? Нет.

Таким образом проверили, граничные условия, но что на счет значений внутри класса? Необходимо добавить проверку – проверить значение внутри класса. Какое значение стоит выбрать? **Неэффективная техника:** среднее значение между 18 и 255. **Эффективная техника:** выбрать часто используемо значение между 18 и 255, например, 25.

Картина полна? Нет. Картина никогда полной не будет.

Мы можем разбивать тесты на классы эквивалентности по разным принципам. От этого эффективность тестирования может выиграть. Например, если тестируем поле ввода, которое принимает максимум пять символов, то можно выбрать разные принципы разбиения на классы эквивалентности:

* по количеству символов;
* по типу символов (цифры, буквы, спецсимволы).

Т.е в нашем примере с возрастом возможно добавить проверку символов. Т.е. тестов может быть намного больше.

**Наша задача - отобрать самые критичные и важные тесты**. В рамках второй лекции говорили о последовательности выполнения тестирования: всегда идем от простого к сложному, от позитивных тестов к негативным:

* Простой позитивный тест
* Простой негативный тест
* Сложный позитивный тест
* Сложный негативный тест

Например, если кто-то введет кракозябры и зарегистрируется - это страшно, но не очень. Гораздо страшнее, когда не сможет зарегистрироваться 18-летний тестировщик.

Итого для поля «Сколько Вам лет?» получили 7 проверок.



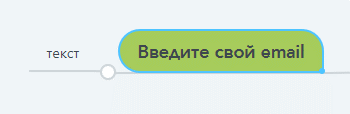
В итоге получили такой список проверок для формы «Регистрация участника»:

* Позитивные и негативные проверки
* Классы и границы
* Добавим значения внутри класса

**Наша задача - минимальным количеством тестов добиться максимального покрытия!**

## E-mail: исследуем стандарты, совмещаем проверки

Каким может быть EMAIL? Как мы можем проверить это поле?

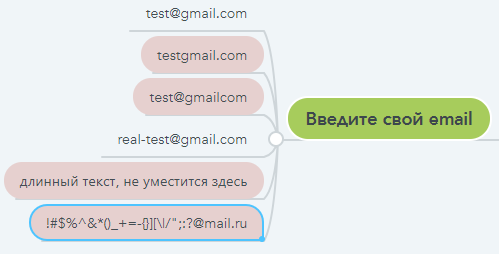


Здесь тоже есть классы эквивалентности:

* Общая длина – количество символов всего email
* Длина локальной части email
* Длина доменной части email
* Допустимые и недопустимые символы
* Обязательные символы

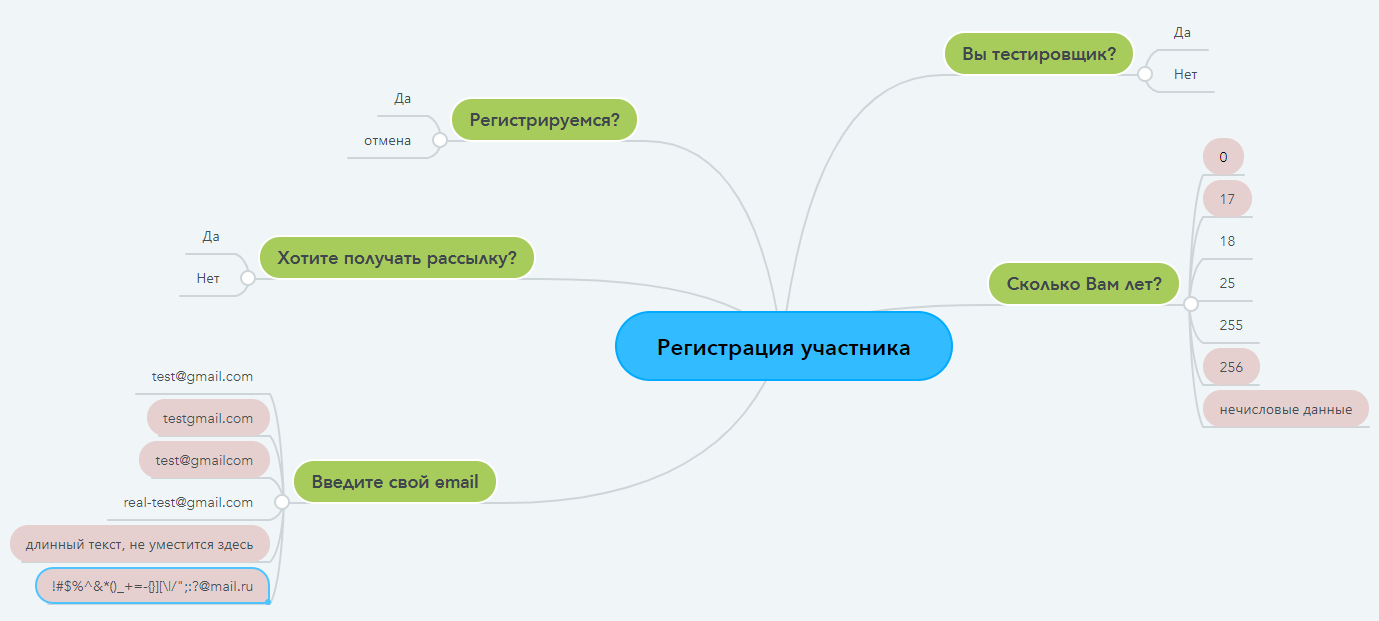
Для некоторых полей есть свои стандарты. Например, есть стандарты, каким может быть email. В соответствии со стандартами мы должны выбирать проверки в том числе. (Какие стандарты есть для поля email в данной лекции не описывается. Нужно погуглить=) ). Додумки неприемлемы!

Какие проверки получились:



Это не все проверки, какие можно выполнить! =)

Получаем такую карту:



После исследования формы что необходимо выполнять дальше?

На карте зеленым выделены параметры, которые отображены на форме. Но это не все параметры, которые влияют на работу нашего продукта.

Какие же еще параметры есть, которые влияют на работу продукта, при этом в интерфейсе их нет, они не очевидны?

Дальше по этой карте возможно создать тестовый набор. В тестовый набор будут входить проверки:

• Проверяем по отдельности все сущности

• Комбинируем проверки на основе функционала

• Тестируем отдельно бизнес-логику

• Добавляем нефункциональные тесты!

Проверяем сущности:

• Как сделать?

• Как проверить?

• В каком порядке?

• Где нужны нефункциональные тесты?

Мы можем выполнять все проверки по отдельности (например, отдельно провести тест по полю Тестировщика Да или Нет, отдельно по полю EMAIL, и т.д.), но это не совсем эффективно, поэтому смело можем объединить **положительные** проверки. Например, выполнить одновременно проверку по полю «Вы тестировщик?» с условием выбора Да и по полю «сколько Вам лет?» с выбором значения 18.

Негативные проверки как правило не объединяются.

Ответив на эти вопросы, мы можем сформировать набор тестов.

## Бизнес-логика

Так же при проектировании тестов нужно основываться и на бизнес-логику.

• Что влияет на поведение продукта?

• Как всё это проверить?

Например:

|  |  |  |  |
| --- | --- | --- | --- |
| **Е-мейл** | **Возраст от 18?** | **Пользователь – тестировщик?** | **Получить рассылку?** |
| Валидный - регистрируем | Да – регистрируем | Да – регистрируем | Да – подписываем (только если регистр.) |
| Невалидный – говорим «введите корректный е-мейл» | Нет – говорим «Тебе пока рано!» | Нет – говорим «ты слабое звено» | Нет – не подписываем |

Важно смотреть как правильно комбинируются проверки. Помимо приведенных в таблице примеров, есть еще случаи, когда email корректный, остальные поля некорректны, и т.д.

Есть разные техники, с помощью которых комбинируются проверки. Одна из самых популярных – таблица решений. Пример таблицы решений рассмотрим на другой лекции.

Итого:

• Тест-анализ – это процесс определения ЧТО будет тестироваться

• Тест-дизайн – определение КАК это будет тестироваться

• Проектирование тестов – процесс, которые есть всегда ВНЕ зависимости от документирования, тест-кейсов, чек-листов

• Проектирование позволяет оптимизировать тесты, экономить время, повышать тестовое покрытие