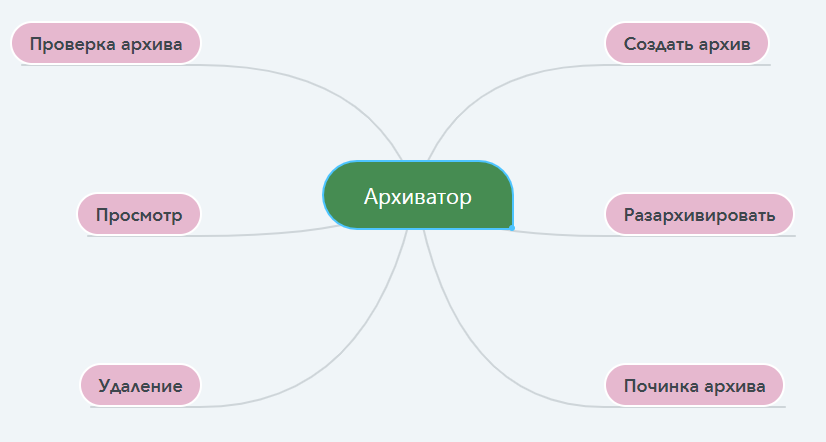
Лекция. Тест – анализ. Техники проектирования тестов.

## Анализ функционала на основе действий

Это базовая техника. Она используется для того, чтобы не забыть важные проверки, для комбинации проверок.

Сначала анализируются действия. У любого программного обеспечения есть какие-либо действия. Для того, чтобы рассмотреть этот подход сегодня рассмотрим приложение архиватор. Какие действия с помощью него можно выполнить:

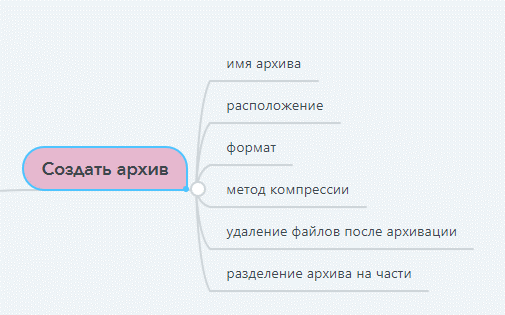


Анализируем продукт, анализируем документацию, чтобы понять, что продукт умеет делать. В результате анализа получаем список действий.

После определения списка действий необходимо определить список параметров действий.

## Анализ параметров

Параметры действия есть всегда. Параметры бывают очевидными и неочевидными, т.е. они явно не указаны.



Параметры всегда влияют на действия. Для любого действия параметры есть всегда, даже если мы его не выбираем. Если мы его не выбираем, значит параметр уже предустановлен по умолчанию.

Т.е. результат анализа – список параметров действий.

## Анализ значений параметров

После получения списка параметров, необходимо проанализировать значения параметров действий.

Например:

Имя архива – может быть коротким, длинным, с разными символами, в разной кодировке.

Расположение архива – локальной, сетевое, usb flash.

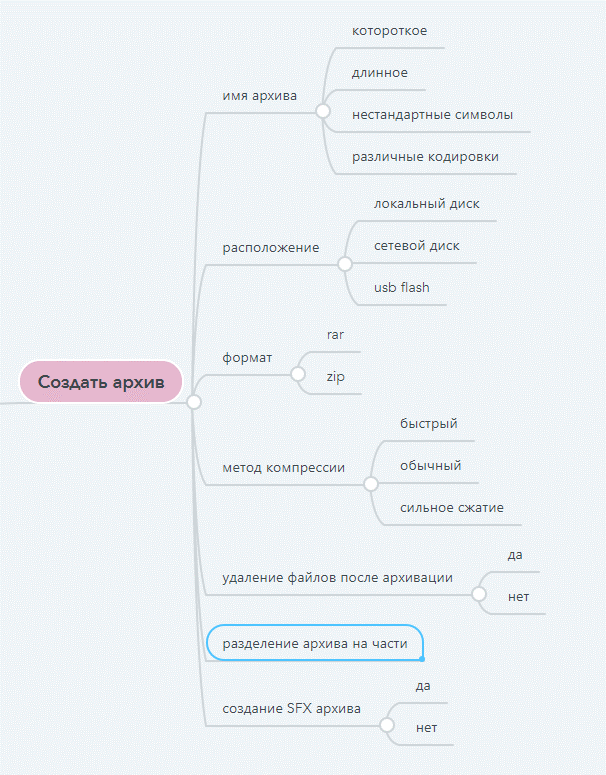
Формат архива - rar, zip  
и т.д.

Есть простые параметры, где есть всего два варианта значений – Да или Нет.

Есть более сложные, которые явно не видны, например, какой тип файла может быть помещен в архив, какого размера.

Именно со сложными параметрами есть большая вероятность обнаружить ошибку.

Ниже на рисунке представлены значения для параметров действия «Создать архив»



Результат анализа – список значений параметров действий.

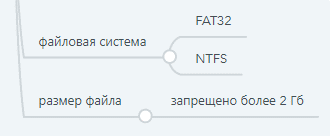
Некоторые параметры могут быть не указаны в требованиях, но это не помешает им вызвать ошибку.

## Анализ архитектуры

Анализируем архитектуру продукта, т.к. можем не знать какую-то часть продукта.

Например:

* в какой файловой системе должен работать Архиватор: fat32, ntfs;
* размер архивируемого файла – запрещено более 2Гб



За подобной информацией можно общаться с разработчиками, смотреть код, смотреть настройки, опыт и исследовательское тестирование.

Результат анализа архитектуры – список значений параметров действий.

## Создание тестового набор на основе результатов анализа

На основе анализа создаем тестовый набор, при этом используем техники:

• Позитивные, **исследовательские** и **негативные** проверки

• Учёт архитектуры продукта

• Оптимизация набора за счёт классов эквивалентности и границ значений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длина имени | Символы | Расположение | Файловые системы | Размер файла |
| 0 | Английские | Локально | NTFS | 0 |
| 1 | Русские | Сеть | FAT32 | 1 байт |
| 255 | ***Спец. символы*** | USB |  | 2 Гб |
| 256 |  | ***CD-RW*** |  | 2,01 Гб |

В результате получается список значений параметров действий, с которыми необходимо выполнить проверки.

В уме такой набор значений невозможно держать, есть риск пропустить ошибки. Тестирование в уме - всегда тестирование низкого качества!

Чтобы сделать эти тесты максимально эффективными переходим к комбинаторике.

## Комбинирование тестов

Есть паттерны проектирования тестов. Их много. Рассмотрим основные на примере нашего тестового набора.

### 1) Подход минимальных проверок

Каждая строка в таблице представляет собой один тест, т.е. когда проходит один тест, сразу проверяется 5 параметров.

Во второй строке - все параметры новые, т.е те, которых нет в первой строке. Далее так же.

Максимальное количество значений в столбце - 4, т.е. меньше четырех тестов сделать нельзя. В рамках действия мы не можем отказываться от каких-то параметров, дальше можно проверять на любой файловой системе или чередовать. Для имени можно вводить промежуточные значения, на количество тестов не повлияет. Получаем **максимальное покрытие - минимальным количеством тестов**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длина имени | Символы | Расположение | Файловые системы | Размер файла |
| 1 | Английские | Локально | NTFS | 1 байт |
| 255 | Русские | Сеть | FAT32 | 2 Гб |
| 1 | ***Спец. символы*** | USB | NTFS | 1 байт |
| 255 | Английские | ***CD-RW*** | FAT32 | 2 Гб |

В комбинаторике не участвуют негативные проверки. Если в 2 поставить 256, то будет ошибка (правильное поведение), но мы не проверим все остальные параметры. Все негативные тесты проводим отдельно от позитивных и отдельно друг от друга, чтобы исключить их влияние друг на друга.

**Итого:**

Минимум проверок – 4

Негативные выносятся отдельно

Количество тестов = максимальное количество значений у параметров + негативные тесты.

### 2) Метод перебора значений

С помощью данного метода проверяем все значения параметров в комбинации друг с другом, т.е. подразумеваем, что какая-то комбинация может привести к ошибке. Количество тестов = произведение параметров. Например, Длина имени – 2 параметра, Символы – 3 параметра, Расположение – 4 параметра, и т.д

Количество тестов = 2 \* 3 \* 4 \* 2 \* 2 = 96.

В результате получаем количество тестов - 96. Если считать, что изначальная таблица верна, не пропущен ни один параметр, то получаем 100%-е покрытие.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длина имени | Символы | Расположение | Файловые системы | Размер файла |
| 1 | Английские | Локально | NTFS | 1 байт |
| 255 | Английские | Локально | NTFS | 1 байт |
| 1 | Русские | Локально | NTFS | 1 байт |
| 255 | Русские | Локально | NTFS | 1 байт |
| ... | ... | ... | ... | ... |

Такой подход может подойти для каких-то ключевых фич/параметров, для автоматизации, т.к. очень трудоемкий (но даже если это 100500 миллионов комбинаций - это будет долго).

**Итого:**

Максимум тестов = 96

Максимальное покрытие

Количество тестов = умножение количества всех значений всех параметров.

### 3) Метод атомарных проверок

Допустим первый тест прошел успешно. Второй тест – не прошел, fail, но в таком случае непонятно, по какой причине тест упал: из-за длины имени 255 или из-за русских символов, или из-за расположения, и т.д. В этом случае **необходимо локализовать ошибку**!

Для того, чтобы локализовать ошибку в таком случае используется подход атомарных проверок. Принцип его заключается в том, что сначала необходимо провести тест с основными, самыми стабильными и часто используемыми простыми значениями. После в следующий тест отличается от предыдущего ТОЛЬКО одним полем. Если второй тест упал (fail), то становится очевидно в чем причина. Аналогично далее.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длина имени | Символы | Расположение | Файловые системы | Размер файла |
| 1 | Английские | Локально | NTFS | 1 байт |
| 255 | Английские | Локально | NTFS | 1 байт |
| 1 | Русские | Локально | NTFS | 1 байт |
| 1 | Английские | Сеть | NTFS | 1 байт |
| ... | ... | ... | ... | ... |

**Итого:**

Атомарные тесты = 9

Легко локализуемое покрытие

Количество тестов = сумма значений - сумма параметров.

Подходит для неопытных тестировщиков и для автоматизации.

### 4) Попарное тестирование (pairwise)

Зачастую ошибки вызываются не конкретным значением, а комбинацией значений. 97% ошибок кроется в комбинации не более чем двух параметров. Такие баги менее приоритетные, чем отдельно не работающие значения, но более приоритетные, чем негативные проверки. Метод попарного тестирования заключается в том, чтобы **за минимальное количество тестов перебрать максимальное количество пар.**

**В каждом тесте высчитывается количество пар.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Длина имени | Символы | Расположение | Файловые системы | Размер файла | Количество пар |
| 1 | Английские | Локально | NTFS | 1 байт | 10 |
| 255 | Русские | Локально | FAT32 | 2 Гб | 10 |
| 255 | Английские | Сеть | NTFS | 2 Гб | 8 |
| 1 | Русские | Сеть | FAT32 | 1 байт | 8 |
| ... | ... | ... | ... | ... | ... |

Существует множество утилит, которые помогают подготавливать наборы пар для тестирования. Одна из самых популярных и бесплатных – это AllPairs. На вход утилите подается таблица значений. В результате формируется таблица со значениями и количеством пар. В первом тесте 5 значений и 10 пар проверяется. В 3-м и 4-м тестах по 8 пар, т.к. в предыдущих тестах эти пары уже включены.

*Почему в 3-ей строке 8 пар?*

*Ответ: 1) Английские символы + файловая система NTFS – участвует в 1-м тесте; 2) Длина имени 255 + размер файла 2 Гб – участвует во 2-м тесте.*

**Итого:**

Перебор всех пар - 12

Сложно локализуемое покрытие

Количество тестов = перемножение двух максимальных наборов значений.

**5) Метод взаимосвязанных проверок**

Рассмотрим комбинацию всех техник, которые рассмотрели выше. Важно выбрать проверки, связанные между собой. Если параметры не связаны, то не имеет смысла делать полный перебор значений.

Итак:

Анализируем параметры. Выявляем какие комбинации могут вызвать проблемы.

Остальные позитивные тесты проверяются минимальными проверками.

Так же проверяем негативные тесты – методом атомарных проверок.

Например, файловая система и размер файла взаимно могут влиять, 2\*2 = 4 + тест по принципу минимальной проверки.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длина имени | Символы | Расположение | Файловые системы | Размер файла |
| 1 | Английские | Локально | NTFS | 1 байт |
| 1 | Английские | Локально | FAT32 | 2 Гб |
| 1 | Английские | Локально | NTFS | 2 Гб |
| 1 | Английские | Локально | FAT32 | 1 байт |
| 255 | Русские | Сеть | NTFS | 1 байт |

Дифференцированное количество проверок, учет взаимозависимостей при формировании набора, подходит при успешном освоении всех предыдущих техник.

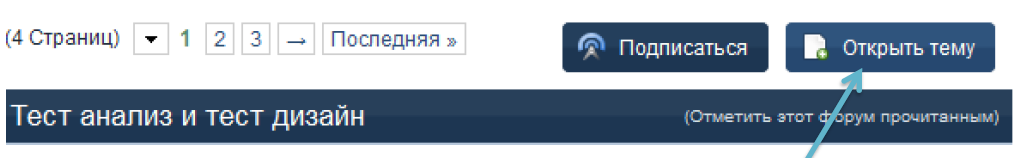
**Сравнение подходов:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Параметры** | **Минимальные проверки** | **Перебор** | **Атомарные** | **Pairwise** | **Взаимозависимости** |
| Количество тестов | 4 | 96 | 9 | 12 | ? |
| Глубина покрытия | ~70% | 100% | ~71% | ~97% | ? |
| Простота создания | Легко | Легко | Легко | Средне | Сложно |
| Локализация дефектов | Сложно | Легко | Легко | Сложно | Легко (+/-) |
| Область применения | Неприоритетный функционал, смоук-тесты | Критичный функционал, автоматизация | Функционал среднего приоритета, автоматизация | Высокий приоритет, сжатые сроки | Квалифицированные тест-дизайнеры |

## Тестирование бизнес – логики

Бизнес - логика – это те действия, ограничения, параметры, которые влияют на поведение программы.

Рассмотрим пример создания поста на форуме тестировщиков по кнопке «Открыть тему».



Необходимо проанализировать от чего зависит поведение кнопки.

Бизнес-логика ПО:

• Если пользователь зарегистрирован, открыть окно новой темы, если нет – не открывать окно;

• Если пользователь оставлял сообщения последние 60 секунд, выдать ошибку (на случай, если это спам), если нет – не выдавать ошибку, открыть окно новой темы;

• Если пользователь – модератор, то ограничение на 1 сообщение в 60 секунд недействительно.

Сколько тестов необходимо выполнить, чтобы проверить вышеупомянутые правила?

Количество правил - 3,

Количество тестов - 2^3 = 8 (2 в степени числа правил)

Составляем **таблицу решений** (или еще ее называют **таблица альтернатив**).

**Как она работает:**

**Условия** перечислены в **левой верхней** части таблицы, а **действия** ‒ в **левой нижней.** Остальные столбцы содержат бизнес-правила. Каждое правило утверждает (вкратце): **«В этой конкретной комбинации условий необходимо выполнить конкретную комбинацию действий».** Заполняем правила значениями полным перебором для булевых значений, где Y- YES, N – NO. Количество столбцов, т. е. количество бизнес-правил, равняется 2 в степени числа условий: 2^3 = 8



Получение тестовых сценариев в этом примере простое: каждый столбец соответствует тестовому сценарию. Когда дело дойдет до выполнения тестов, будут созданы условия, которые являются входными данными для тестов. Условия «Истина/Ложь» (Y/N) будут заменены на реальные входные значения для перечисленных условий. И будут проверены действия, которые являются ожидаемыми результатами для тестов.

Рассмотрим составленную таблицу. Некоторые тесты здесь не имеют смысла или невозможны! Например, тестовые сценарии 5, 6 и 7. Убираем из таблицы лишние проверки.



**Итого** остается пять тестовых сценариев. В результате есть понимание, что покрытие полное.

И напоследок небольшое видео ☺ <https://pointschool.ru/razbor-kombinatornyh-tekhnik-test-dizajn/>