[Модуль 4. Часть 2. Техники тест-дизайна](https://lms.skillfactory.ru/courses/course-v1:SkillFactory+INTQAP+2022/course/" \l "block-v1:SkillFactory+INTQAP+2022+type@sequential+block@7b1b9d7551f94844b35cd11f4a486a7c)

У негативных тест-кейсов, как правило, может быть два результата:

* 1. Непредусмотренные действия обрабатываются системой, определяются и выдаётся подсказка.
  2. Непредусмотренные действия вызывают системную ошибку. Такое поведение и необработанное исключение — это стопроцентный дефект.

Цель негативного тестирования — исключить возникновение подобных ситуаций, так как они могут привести к более серьёзным дефектам.

**Негативное тестирование** направлено, прежде всего, на:

* + проверку устойчивости системы к различным воздействиям;
  + валидацию неверных данных;
  + обработку исключительных ситуаций.

Сценарии **позитивного тестирования**, в свою очередь, направлены на проверку работы системы с теми типами данных, для которых она разрабатывалась.

Создание позитивных тест-кейсов, как правило, предшествует созданию негативных тест-кейсов. Сначала мы проверяем работу системы, когда пользователь работает с системой «правильно», а потом приступаем к проверке отклика системы на пользователя, который допускает различные ошибки (ввод неверных данных, например). В этом случае тестируемая система должна обработать неверный запрос.

Стоит отметить, что позитивные и негативные тест-кейсы могут существовать не только для функционального, но и для нефункционального тестирования.

Рассмотрим **нагрузочное тестирование**. Пусть нефункциональные требования к системе — 10 поисковых запросов в секунду со временем отклика не хуже 100 миллисекунд. Тогда позитивным тестом будет проверка того, что система действительно может обслужить такое количество запросов с заданным временем отклика. Отрицательным кейсом будет попытка отправить 100 запросов в секунду. Система должна корректно отработать эту ситуацию, пусть с меньшим временем отклика, и не рухнуть под такой нагрузкой.

Рассмотрим несколько советов-подсказок по негативным кейсам, которые связаны с проверкой форм ввода данных.

**Обязательные поля ввода данных**

Необходимо убедиться, что система действительно считает поля ввода данных обязательными к заполнению и не отправляет форму, пока они не заполнены. Если пользователь получает ошибку о необходимости заполнения поля, то такой тест-кейс будет считаться пройденным. Если форма отправляется даже без заполнения обязательного поля, то тест-кейс считается непройденным, и соответствующий дефект должен быть занесён в баг-трекинговую систему (англ. *bug tracking system* — система отслеживания ошибок).

Пример: устраивает ли разработчика и владельца сайта ситуация, при которой пользователь при регистрации может авторизоваться в системе, не вводя *email* и пароль.

**Формат данных в полях**

Числовые поля должны принимать только числовые данные. Строковые поля принимают любые символы. Однако очень часто строковые поля принимают данные определённого формата, например, *email* или номер телефона. Поля, принимающие дату, должны принимать данные в формате даты.

**Ограничения на вводимые данные**

Например, количество товаров в корзине не может быть отрицательным, а если есть период, определяемый двумя значениями дат, то дата начала должна быть меньше даты окончания.

**Peer-to-peer** (используется также как P2P, peer-2-peer) — это термин, обозначающий взаимное обучение. Оно позволяет формировать связи между участниками образовательного процесса через обмен знаниями, опытом и мнениями друг с другом. Этот формат предполагает, что один или несколько студентов взаимно обучают остальных участников группы.

Внимательно ознакомьтесь с критериями оценивания и напишите своему будущему коллеге развёрнутый комментарий:

* + почему вы оценили именно так,
  + какие рекомендации вы могли бы оставить,
  + как можно было улучшить выполненную работу.

## 4.10. Разделение на классы эквивалентности и анализ граничных значений

Если следовать логике в ошибочном применении условных операторов, изложенной выше, то ошибиться можно не только в условиях > или <, но также в условиях ≤ или ≥. Это значит, что систему управления паролями необходимо также проверить на работу с паролями, чья длина близка к граничным значениям.

## **Подведём итоги**

В этом уроке мы вместе решали задачи по применению двух техник тест-дизайна:

* разбиение на **классы эквивалентности**;
* анализ **граничных значений**.

## 4.11 Причина — следствие

**ISTQB**  
  
**таблица причинно-следственных решений (cause-effect decision table)**: См. таблица решений.  
  
**таблица решений (decision table):**Таблица, отражающая комбинации входных данных и/или причин с соответствующими выходными данными и/или действиями (следствиями), которая может быть использована для проектирования тестовых сценариев.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Правило 1** | **Правило 2** | **Правило 3** | **Правило 4** | **Правило 5** | **Правило 6** |
| **Условия** |  |  |  |  |  |
| Роль | Администратор | Учитель | Ученик | Администратор | Учитель | Ученик |
| Лицензия активна? | Да | Да | Да | Нет | Нет | Нет |
| **Действия** |  |  |  |  |  |  |
| 1 | Вход как администратор | Вход как учитель | Вход как ученик | Переход на страницу продления лицензии | Переход на страницу продления лицензии | Переход на страницу продления лицензии |

Всё, что находится в разделе «Условия», представляет собой входные условия.

Всё, что находится в разделе «Действия», представляет собой действия, которые будут выполнены, или события, которые произойдут при выполнении этих условий.

Каждое правило в вертикальной колонке может рассматриваться как тест-кейс, а действие как ожидаемый результат. В итоге мы получаем 6 позитивных тест-кейсов для тестирования входа в систему.

## **Программа подготовки сертифицированных тестировщиков (ISTQB®) о таблицах решений (альтернатив)**

**Таблицы альтернатив** — хороший способ записи сложных бизнес-правил, которые должны быть реализованы в системе. В процессе создания таблицы тестировщик определяет условия (входы) и результирующие действия системы (выходы). Пары условий и действий образуют строки таблицы, при этом условия указываются сверху, а действия — снизу. Каждый столбец представляет собой бизнес-правило с уникальной комбинацией условий и действий, связанных с этим правилом.

Значения условий часто отображаются в виде логических (истина или ложь) или дискретных (красный, синий, зелёный) значений, но могут быть также в виде чисел или числовых диапазонов. В одной таблице могут сочетаться значения разных типов.

## рис **Обозначения, используемые для таблиц альтернатив**

**Условия**:

«Y» означает, что условие истинно (используется также обозначение Т или 1).

«N» означает, что условие ложно (используется также обозначение F или 0).

«—» означает, что значение условия может быть любым (используется также обозначение N/A; подробнее читаем [здесь](https://dic.academic.ru/dic.nsf/ruwiki/191452)).

**Действия**:

«Х» означает, что действие должно быть выполнено (используется также обозначение Y, Т или 1).

Пустое поле означает, что действие не должно выполняться (используется также обозначение N, F или 0).

Полная таблица альтернатив содержит по столбцу на каждую комбинацию условий. Таблицу можно сократить, убрав столбцы, которые содержат несуществующие комбинации или комбинации, не влияющие на результат.

## 4.12. Предугадывание ошибки

Техники тест-дизайна, основанные на опыте, используют, как правило, при создании тестов, которые невозможно получить, применяя другие, более системные методы. Одной из таких техник является **предугадывание ошибок** или **прогнозирование ошибок** (согласно ISTQB). К этой технике мы уже обращались, разбирая один из примеров о тестировании формы авторизации. Думается, будет полезным повторить.

**ISTQB**  
  
**Предположение об ошибках** — это способ предотвращения ошибок, дефектов и отказов, основанный на знаниях тестировщика, включающих:

* + историю работы приложения в прошлом;
  + наиболее вероятные типы дефектов, допускаемых при разработке;
  + типы дефектов, которые были обнаружены в схожих приложениях.

**Предугадывание ошибки** — это ситуация, при которой тестировщик думает над тем, какие ошибки могли быть допущены в процессе разработки, а также определяет пути их появления, используя интуицию, знания и опыт.

## **Преимущества техники «предугадывание ошибки»**

Техника «предугадывание ошибки» очень полезна при отсутствии строгих спецификаций, при исследовательском тестировании и в дополнение к другим техникам. К примеру, при выделении классов эквивалентности в тестовых данных можно определить какие-то особые наборы внутри одного класса эквивалентности, предполагая ошибки, исходя из своего опыта.

Безусловно, нужный опыт накапливается постепенно. Однако умение предвидеть ошибки, которые могли выпустить из внимания, следует тренировать постоянно. Задавайте себе вопрос: «А что если.... я в первый раз… загружаю фото в профиль пользователя… (или выполняю другое действие), что я могу сделать не так».

Другое преимущество этой техники — использование в дымовом тестировании. Мы помним: в дымовом тестировании нужно обязательно выполнить основные позитивные кейсы, чтобы убедиться в общей готовности функциональности продукта. Однако опытный тестировщик, зная наиболее часто допускаемые разработчиками ошибки, выполнит и несколько негативных тестов, построив их на основе своего опыта. Если выполнение этих негативных тестов приведёт к критической или — что ещё хуже — блокирующей ошибке, то тестирование необходимо будет остановить.

## **Недостаток техники «предугадывание ошибок»**

Техника не может использоваться изолированно, чтобы обеспечить полное тестовое покрытие. Если в ходе её применения будет найдено определённое количество ошибок — невозможно гарантировать то, что весь функционал был протестирован.

Например, если был найден дефект синхронизации при попытке записи файлов в каталоги, доступные только по чтению, это ещё не означает, что все возможные дефекты синхронизации были выявлены. Поэтому, именно из-за отсутствия полного охвата объекта тестами, предугадывание ошибки обычно применяется в сочетании с иными техниками тест-дизайна.

Другим существенным недостатком, вызывающим затруднения в использовании этой техники, является отсутствие серьёзного профессионального опыта и знаний у тестировщика. Ошибочное представление, что тестировщику не нужно глубокое знание не только технологий, на которых построено тестируемое приложение, но и предметной области тестируемого приложения, также не способствует применению техники «предугадывание ошибок».

## **Подведём итоги**

Первое, на что стоит обратить внимание при использовании данной техники — это входные параметры. Часто разработчики намеренно не тратят время на продумывание и внедрение состояний, которые, по их мнению, невозможны. Бесспорно, это может быть и проблема аналитиков, составивших не полные требования.

Важно чётко понимать главное: продуктом будут пользоваться люди с самым разным мышлением, и они могут вводить в систему самые разные данные, которые могут некорректно обрабатываться и вызывать ошибки.

Задача тестировщика состоит как раз в том, чтобы предусмотреть все возможные варианты поведения пользователя и проверить их. Конечно, могут быть ошибки, связанные не только с поведением пользователя, но и со взаимодействием приложения с окружением или с протоколом передачи данных. Но это уже другая история, а пока переходим к диаграммам перехода состояний и таблицам решений.

## 4.13. Диаграммы перехода состояний

Любая система может вести себя по-разному в зависимости от прошлых условий и состояния. Описать прошлое системы можно с помощью концепции состояний. Диаграмма состояний и переходов показывает начальное и конечное состояния системы, а также описывает переходы между состояниями. Каждый переход вызывается событием (например, вводом данных пользователем). Если одно и то же событие может привести к разным переходам, выбор перехода может задаваться контрольным условием. Смена состояния может завершаться выполнением какого-либо действия (вывод результатов или сообщения об ошибке и так далее)».

**Состояние** — это режим работы, при котором система ожидает одного или нескольких событий. Находясь в определённом состоянии, система помнит то, что получила в прошлом. Это прошлое состояние определяет, как система отреагирует на последующие события при их возникновении.

События могут вызывать переходы в другое состояние или инициировать определённые действия. В общем случае состояние представлено значениями переменных внутри системы состояния. На диаграммах состояние будет представлено окружностью.

**Переход**— представляет собой изменение от одного состояния к другому, как результат возникновения некоторого события. Переход обозначается на диаграмме стрелками.

**Событие** — это то, что приводит к изменению состояния системы. Событие обозначается меткой над стрелкой. Событие находится вне системы и воздействует на неё через некоторый интерфейс.

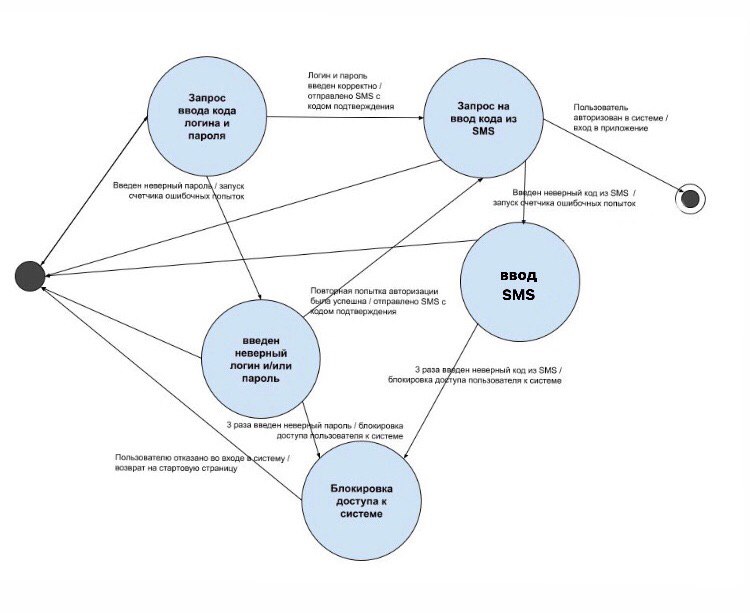
Например, в Zoom, популярной платформе для видеоконференций, пользователь, желающий присоединиться к онлайн-встрече (митингу), перейдя по ссылке, может ожидать момента, когда инициатор митинга стартует его. До этого момента пользователь видит на странице митинга вращающийся спиннер.

Событие может быть и внутри системы, например, таймер (как в нашем случае). События могут быть независимые и зависимые, например, событие A наступает только после события B. Когда событие случается, система может изменить состояние, остаться в том же самом состоянии или выполнить действие.

**Действие** — операция, инициируемая сменой состояния. Действия происходят на переходах между состояниями. Действия описываются после события на метке над стрелкой. Событие и действия разделяются знаком «/».

Также на диаграмме присутствуют точки входа и выхода, обозначаемые закрашенными кружками и кольцом с закрашенным центром.

Полная диаграмма показана на рисунке 2.



Помимо диаграмм состояний и переходов для фиксации требований и понимания работы системы можно использовать таблицы состояний и переходов. Если диаграмма визуализирует работу системы, то таблица состояний и переходов помогает построить тест-кейсы для тестирования системы.

Таблица состояний и переходов включает 4 колонки: текущее состояние, событие, действие, следующее состояние. Ячейки таблицы содержат описания этих сущностей.

Покажем построение таблицы состояний и переходов на примере кейса авторизации в онлайн-банке.

|  |  |  |  |
| --- | --- | --- | --- |
| **Состояние** | **Событие** | **Действие** | **Следующее состояние** |
| Стартовая страница. | - | - | Запрос ввода логина и пароля. |
| Стартовая страница. | N/A | N/A | Запрос ввода кода из *SMS*. |
| Стартовая страница. | N/A | N/A | Запрос ввода логина и пароля. Предыдущий ввод был неудачный. |
| Стартовая страница. | N/A | N/A | Запрос ввода кода из *SMS*. Предыдущий ввод был неудачный. |
| Стартовая страница. | N/A | N/A | Блокировка доступа пользователя к системе. |
| Стартовая страница. | N/A | N/A | Успешный вход в систему. |
| Запрос ввода логина и пароля. | Возврат в систему. |  | Стартовая страница. |
| Запрос ввода логина и пароля. | Пароль и логин введён корректно | Отправка *SMS* с кодом подтверждения. | Ожидание ввода кода из *SMS*. |
| Запрос ввода логина и пароля. | Пароль введён некорректно. | Запуск счётчика некорректных попыток ввода пароля. | Запрос ввода логина и пароля. Предыдущий ввод был неудачный. |
| Запрос ввода логина и пароля. | N/A | N/A | Запрос ввода кода из *SMS*. Предыдущий ввод был неудачный. |
| Запрос ввода логина и пароля. | N/A | N/A | Блокировка доступа пользователя к системе. |
| Запрос ввода логина и пароля. | N/A | N/A | Успешный вход в систему. |
| Запрос ввода логина и пароля. Предыдущий ввод был неудачный. | Возврат в систему. |  | Стартовая страница. |
| Запрос ввода логина и пароля. Предыдущий ввод был неудачный. | N/A | N/A | Запрос ввода логина и пароля. |
| Запрос ввода логина и пароля. Предыдущий ввод был неудачный. | Пароль и логин введён корректно. | Отправка *SMS* с кодом подтверждения. | Запрос ввода кода из *SMS*. |
| Запрос ввода логина и пароля. Предыдущий ввод был неудачный. | N/A | N/A | Запрос ввода кода из *SMS*. Предыдущий ввод был неудачный. |
| Запрос ввода логина и пароля. Предыдущий ввод был неудачный. | Три попытки ввода пароля были не удачны. | Блокировка пользователя в системе. | Блокировка доступа пользователя к системе. |
| Запрос ввода логина и пароля. Предыдущий ввод был неудачный. | N/A | N/A | Успешный вход в систему. |

В приведённой выше таблице информация не полная. Данная таблица ставит соответствие состояний системы между собой и определяет возможные состояния и действия в системе для того, чтобы перейти между состояниями.

N/A означает, что таких действий не существует и не должно существовать. Например, нельзя перейти из состояния «запрос пароля» в состояние «авторизован», не пройдя всю цепочку.

## **В чём преимущество таблиц состояний и переходов?**

Когда мы тестируем приложения, в которых недопустимы ошибки при переходе между статусами, то требуется протестировать все комбинации статусов между собой. Использование таблицы помогает вскрыть недокументированные или порой даже неразрешённые пути и переходы от одного состояния к другому. На основе диаграммы состояний и переходов и таблицы состояний и переходов легко создать наборы тест-кейсов для тестирования системы.

При этом рекомендуется использовать следующие подходы:

* 1. Создать тесты, которые проходят несколько состояний за тест, например, позитивный тест на авторизацию в системе.
  2. Создать тесты, в которых происходит несколько событий и действий, например, позитивный тест на авторизацию в системе.
  3. Создать тесты, в которых есть только два состояния, например, тест на возврат в исходное состояние или тест на проверку блокировки доступа.

## 4.14. Техника Pairwise

***Pairwise Testing*** — техника составления уникальных комбинаций значений для нескольких параметров таким способом, который позволяет составить наименьшее число комбинаций, при этом обязательно выполняется условие о наличии всех возможных парных комбинаций значений всех параметров

## **Что делать, если есть пустые значения в наборах?**

Обратили внимание, что количество значений в каждом параметре различно, и ряд значений в наборах данных пуст? Выход из этой ситуации — заполнить пустые ячейки в полученных наборах данных значениями. Значения подбираются по принципу приоритетных к проверке.

Вы хотите выполнить больше тестов для пользователей с минимальным доступом к CRM-системе. Значит, в результирующей таблице, в ячейке, где значение пользовательского доступа пропущено, вы подставляете минимальный доступ. При этом наборы данных для тест-кейсов не должны дублироваться. Следуя этой логике, можно добавить к результирующей таблице дополнительные наборы данных для проверки наиболее критичных тест-кейсов.

Предположим, что в предыдущей итерации тестирования был обнаружен баг: вход в CRM-систему организации-партнёра с полной лицензией под учётной записью администратора. Значит, тест-кейс с этими данными нужно включить в проверки, если в результирующем наборе данных такой комбинации нет.

Выполненная задача достаточно простая, так как в ней используются дискретные параметры. Как правило, в реальности всё намного сложнее.

* 1. В книге «*A Practitioner's Guide to Software Test Design»* *Lee Copeland*  можно также узнать о такой технике, как тестирование по области значений (подробнее и на русском языке [здесь](https://habr.com/ru/post/462837/)).
  2. Кратко о технике тестирования по сценариям использования можно сказать, что данный подход используется при наличии требований или задокументированных сценариев использования, составляемых аналитиками. В отличие от техник тест-дизайна, рассмотренных выше, за исключением техники диаграммы состояний и переходов, техника тестирования по сценариям использования не концентрируется на отдельной функциональности, а тестирует всю систему в целом.

**Пример 1**: пользователь осуществляет перевод денег со счёта на счёт в приложении онлайн-банк — это один из сценариев использования системы. Для тестирования этого сценария нам потребуется выполнить ряд шагов, включая авторизацию в приложении.

**Пример 2**: совершение покупки в интернет-магазине. Этот сценарий использования включает следующие шаги: поиск товара, выбор его из справочника, помещение его в корзину, оплата покупки. Соответственно, тест этого сценария будет включать в себя каждый из этих шагов, декомпозированный на отдельные этапы. Такой тест называют **системным** или ***end-to-end test***.

## **Может ли быть тестирование исчерпывающим?**

**Исчерпывающее тестирование** — это вид тестирования, при котором используются все возможные комбинации входных данных.

С одной стороны, исчерпывающее тестирование может показаться самым надежным методом проверить работоспособность системы при всех комбинациях входных параметров. Но, как мы уже убедились, это невозможно, в особенности, если какой-то параметр представлен непрерывным или числовым значением. Техники тест-дизайна для того и созданы, чтобы оптимизировать тестирование и при меньших затратах достигать наилучшего результата.

Однако есть области, в которых даже исчерпывающее тестирование имеет право на жизнь. Например, тестирование автономных виртуальных беспилотных моделей автомобилей в симуляторах. Подобные симуляторы имитируют поведение реального автомобиля в виртуальном окружении, например, в виртуальных городе, лесу или на открытой местности. Можно представить, какое громадное количество параметров необходимо учесть для того, чтобы составить многочисленные тесты. Перечислим некоторые из них:

* + Параметры оборудования на борту автомобиля (двигатель, подвеска, трансмиссия, тормозная система и прочее (которые тоже в свою очередь будут представлены своими моделями).
  + Параметры окружения, например, городские улицы, дома, пешеходы, транспортные потоки.
  + Имитация погодных условий.

Это лишь часть того, что нужно учесть. Для таких сложных информационных систем проверять придётся всё, так как на кону жизнь человека, после того, как программный код всех систем автомобиля, протестированный в виртуальной среде, будет использоваться в реальных беспилотных автомобилях.

## **Подведём итоги**

Мы начали практиковать базовые техники проектирования тестов, с помощью которых:

* + Разбиваем массивы данных на классы эквивалентности.
  + Выделяем граничные значения.
  + Составляем таблицы решений.
  + Генерируем наборы данных с помощью приложения pairwise online tool.