[Модуль 4. Часть 1. Тест-дизайн](https://lms.skillfactory.ru/courses/course-v1:SkillFactory+INTQAP+2022/course/" \l "block-v1:SkillFactory+INTQAP+2022+type@sequential+block@4eebb2050f634da58109e84b183729cb)

## 4.1 Тест-дизайн: что это, зачем он нужен

**Верификация** — это проверка, соответствует ли продукт спецификации.

**Валидация** — это проверка, соответствует ли продукт требованиям и ожиданиям.

Попробуйте предположить, какие вопросы могут задать начинающие тестировщики в числе первых:

Сколько нужно написать тестов, чтобы быть уверенным, что новая функциональность работает согласно требованиям?

Как правильно выбрать данные для теста?

Как правильно составить шаги проверки?

Если тестов будет много, как сделать так, чтобы ими было удобно управлять и было удобно их поддерживать?

Всё вышеперечисленные и еще n-штук

Вот несколько определений, что такое тестирование:

* **Тестирование** — это процесс, состоящий из всех действий жизненного цикла, как статических, так и динамических, связанных с планированием, подготовкой и оценкой программных продуктов и связанных с ними рабочих продуктов, а также определением того, что они удовлетворяют заданным требованиям, для демонстрации их соответствия назначению и обнаружения дефектов. Тестирование — это процесс, а не отдельное действие, в него входит разные виды деятельности. (*ISTQB*).
* **Тестирование** — это не действие, а интеллектуальная дисциплина, имеющая целью получение надёжного программного обеспечения, без лишних усилий на его проверку. (Борис Бейзер «Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем»).

Вот несколько определений того, что такое тест-дизайн из *ISTQB*:

* + **Тест-дизайн** — это процесс преобразования общих целей тестирования в осязаемые условия тестирования и тест-кейсы.
  + **Тест-дизайн** — это документ, определяющий условия тестирования (элементы покрытия) для элемента тестирования, подробный подход к тестированию и определение связанных с ним тест-кейсов.
  + **Тест-дизайн** — это деятельность, причём активная, направленная на определение тестовых случаев (тест-кейсов) исходя из целей, условий тестирования и определенных критериев качества.

Во время тест-дизайна мы определяем, как мы будем тестировать то, что решили протестировать во время тест-анализа. Используя техники тест дизайна, мы трансформируем общие условия тестирования и общие цели тестирования из плана тестирования в реальные тест-кейсы с нужным уровнем детализации.

Что нужно помнить о тест-дизайне?

* Это часть процесса разработки: этап процесса тестирования ПО, на котором проектируются и создаются тест-кейсы, в соответствии с определёнными критериями качества и целями тестирования.
* Это отдельная дисциплина: набор тестов для объекта тестирования, составленный на основе тестового базиса в соответствии с целями тестирования. Тестовый базис — это документы, на основе которых создаются тесты-кейсы; все документы, из которых можно получить требования компонента или системы. Примеры тестового базиса: требования и спецификация проекта, отчёты по анализу рисков, дизайн и архитектура системы, спецификация интерфейсов.

Техники тестирования помогают нам выбрать хороший набор тестов из общего числа всех возможных тестов для данной системы. Разные техники предлагают нам разные взгляды на тестируемое программное обеспечение. Каждая техника предоставляет набор правил или указаний для тестировщика. Выбранные условия тестирования будут зависеть от стратегии тестирования или тестового подхода. Например, они могут быть основаны на рисках, моделях системы, вероятных сбоях, требованиях соответствия, рекомендациях экспертов.

В каждом конкретном случае, применяемые техники проектирования тестов зависят от многих факторов, таких как:

 Тип системы или компонент, для которого создается тест.

 Сложность системы или компонента.

 Нормативные документы, которые могут накладывать дополнительные ограничения на функциональность разрабатываемой системы.

 Требования пользователей или условия, накладываемые контрактными обязательствами.

 Риски, которые могут нести в себе необнаруженные и, как результат, неисправленные дефекты.

 Доступная документация на разрабатываемую систему.

 Знания и опыт тестировщиков, разрабатывающих тесты и проводящих тестирование.

 Доступные инструменты для выполнения тестов.

 Ресурс времени и бюджет проекта.

 Фаза жизненного цикла разрабатываемой системы.

 Ожидаемое использование системы.

 Ожидаемые типы дефектов разрабатываемого компонента или системы.

 Степень формализации.

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| Методы чёрного ящика | Включаются в большую категорию техник, основанных на спецификации. «Чёрным ящиком» он называется потому, что они рассматривают программное обеспечение как чёрный ящик с входом и выходом, но они не знают, как система или компонент структурирован внутри коробки. По сути, тестировщик концентрируется на том, что делает программа, а не на том, как она это делает. |
| Методы белого ящика | Методы, основанные на структуре (которые также являются динамическими, а не статическими), используют внутреннюю структуру программного обеспечения для создания тест-кейсов. Их называют техниками «белого ящика» или «стеклянного ящика» (подразумевая, что вы можете заглянуть внутрь системы), так как они требуют знаний о том, как реализовано программное обеспечение, как оно работает.  Например, структурная техника может быть связана с выполнением циклов в программе. Различные тест-кейсы могут быть написаны для выполнения цикла один, два и много раз. Это можно сделать независимо от функциональности программного обеспечения. Так как не на всех проектах требуется, чтобы тестировщик разбирался в коде так же хорошо, как и разработчики, эти методы используются чаще разработчиками при тестировании кода. Знать, что они существуют, важно, потому что это значимая часть процессов разработки и обеспечения качества (так называемый код-ревью). |
| Методы, основанные на опыте | Используют знания, навыки и опыт людей (тестировщиков, бизнес-аналитиков, пользователей и других) которые являются основным фактором, влияющим на условия тестирования, и тест-кейсы. Опыт как технических, так и бизнес-пользователей важен, поскольку они привносят разные точки зрения в процесс анализа и проектирования тестов.  Благодаря предыдущему опыту с подобными системами, они могут иметь представление о том, что может пойти не так, что очень полезно для тестирования. Их, как правило, совмещают с методами чёрного и белого ящиков. |

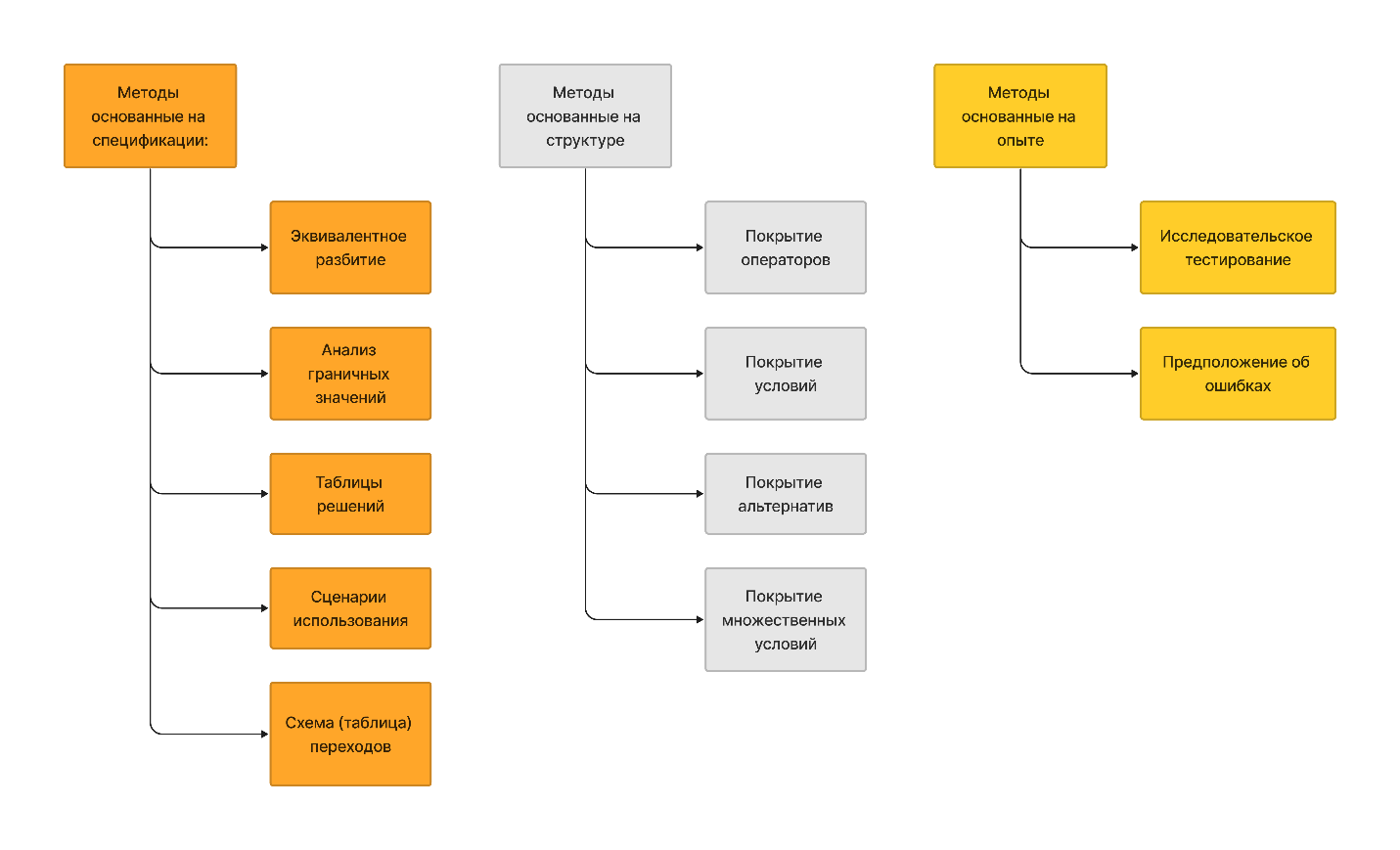
Традиционно принято группировать методы, объединяя: методы чёрного ящика (на основе спецификации), белого ящика (на основе структуры) и методы, основанные на опыте.

**Тестовые данные** — это не система и не элементы системы. Как правило, это артефакты, которые использует пользователь при выполнении шагов: логины, пароли, изображения, файлы и т. п.

**Объект тестирования** (test object или test item) — компонент или система, которую мы будем тестировать. Простыми словами — это то, что будем тестировать.

**Тест анализ** (test analysis) — это процесс изучения того, что может быть использовано для получения тестовой информации. Эта основа для тестов называется тестовый базис (тестовая база).

**Тестовый базис** (test basis) — тестовый базис включает всё, на чём основаны тесты. Это может быть системное требование, техническое задание, сам код (для структурного тестирование) или бизнес-процесс. Иногда тесты могут быть основаны на опытном знании пользователем системы, которое может быть не задокументировано.



**4.2 Тест-кейс как артефакт тестирования**

**Тест-кейс** — это набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определённой цели или тестового условия, таких как выполнения определённого пути программы или же для проверки соответствия определённому требованию.

Другими словами, тест-кейс — это документ, который описывает условия, шаги и ожидаемый результат, благодаря которым тестировщик проверяет, работает ли продукт в соответствии со спецификацией.

**Позитивный тест-кейс** проверяет, что при использовании корректных данных, приложение правильно выполнило функцию в соответствии со спецификацией.  
Пример: при вводе правильного формата электронной почты student@skillfactory.ru при регистрации поле «электронная почта» подсвечено зелёным цветом.

**Негативный тест-кейс** проверяет, что при использовании как корректных, так и некорректных данных, вызываемая функция не выполняется и срабатывает валидация.  
Пример: при вводе неправильного формата электронной почты student@skillfactory при регистрации на портале всплывает сообщение: «Пожалуйста, введите электронную почту в формате student@skillfactory.ru».

**Пример тест-кейса**

**ТК №1. Убедиться, что пользователь может перейти на страницу курса «Тестировщик на Python»**

**Шаги:**

* + - * 1. Открыть браузер.
        2. Ввести в адресной строке *https://skillfactory.ru/*.
        3. Нажать кнопку *Enter*.
        4. На домашней странице сайта *Skillfactory* нажать на кнопку «Все курсы» из меню наверху.
        5. Из ниспадающего списка нажать на ссылку «Программирование».
        6. Промотать экран вниз до заголовка «Курсы программирования» и нажать на баннер со словами «ПРОФЕССИЯ ТЕСТИРОВЩИК-АВТОМАТИЗАТОР». Убедиться, что пользователь перешёл на страницу с описанием курса «Тестировщик на *Python*».

**Ожидаемый результат:**

Пользователь находится на странице с описанием курса «Тестировщик на *Python*».

Основные составляющие тест-кейса это:

* + начальные условия или начальное состояние системы,
  + входные данные,
  + последовательность шагов,
  + ожидаемый результат,
  + конечное состояние системы.

Все эти составляющие должны быть согласованы со спецификацией и требованиями к разрабатываемой системе.

Говоря простыми словами, чтобы правильно написать тест-кейс, нам важно определить:

* 1. предусловия выполнения,
  2. шаги проверки,
  3. ожидаемые результаты,
  4. постусловия выполнения.

Именно эти 4 пункта и формируют структурный состав тест-кейса.

Составляющие тест-кейса будут следующие:

1

***Предусловия (начальные условия или начальное состояние системы)*** — страница портала *lms.skillfactory.ru,* открытая в браузере.

2

***Входные данные*** — это данные, которые мы будем использовать в кейсе. Очевидно, раз мы проверяем функцию авторизации, нам необходимо будет иметь набор аккаунтов для проверки данной функциональности.

3

***Последовательность шагов*** — это действия, которые мы будем выполнять, используя входные данные, от начального до конечного состояния системы.

                 В нашем примере шагами будут:

Нажать на кнопку «Вход» для перехода к форме авторизации.

Ввести *email* пользователя.

Ввести пароль пользователя.

Нажать на кнопку «Вход» формы авторизации.

4

***Ожидаемый результат*** — характерное предсказанное поведение элемента тестирования при указанных условиях на основе его спецификации или другого источника. В нашем примере это авторизация в системе.

5

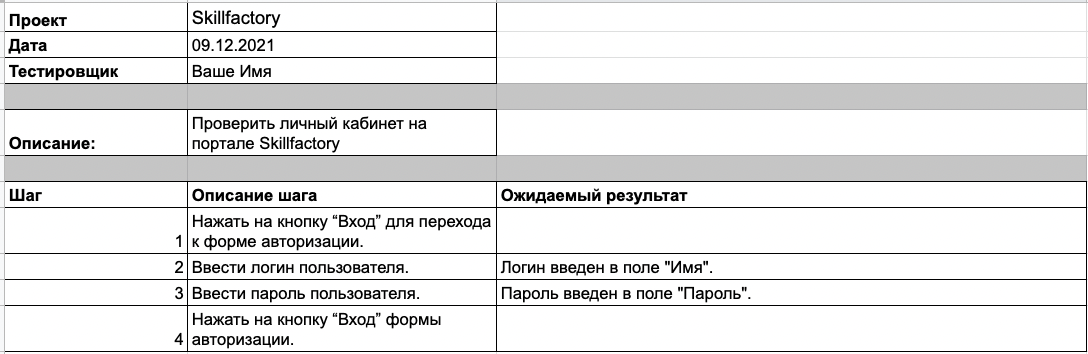
***Постусловия (конечное состояние системы)*** — переход на страницу с доступными курсами. Забегая вперед, необходимо сделать одно важное замечание, что для данной функциональности необходимо использовать не только позитивные тесты, когда верифицируется успешная авторизация пользователя в системе, но и негативные, когда система не авторизует пользователя по определенным причинам. К примеру, при вводе неправильного *email* или пароля.

**Постусловие** — это условие, при котором система возвращается в исходное состояние.

Например, для тест-кейса Авторизация в системе постусловием будет Выход из системы.

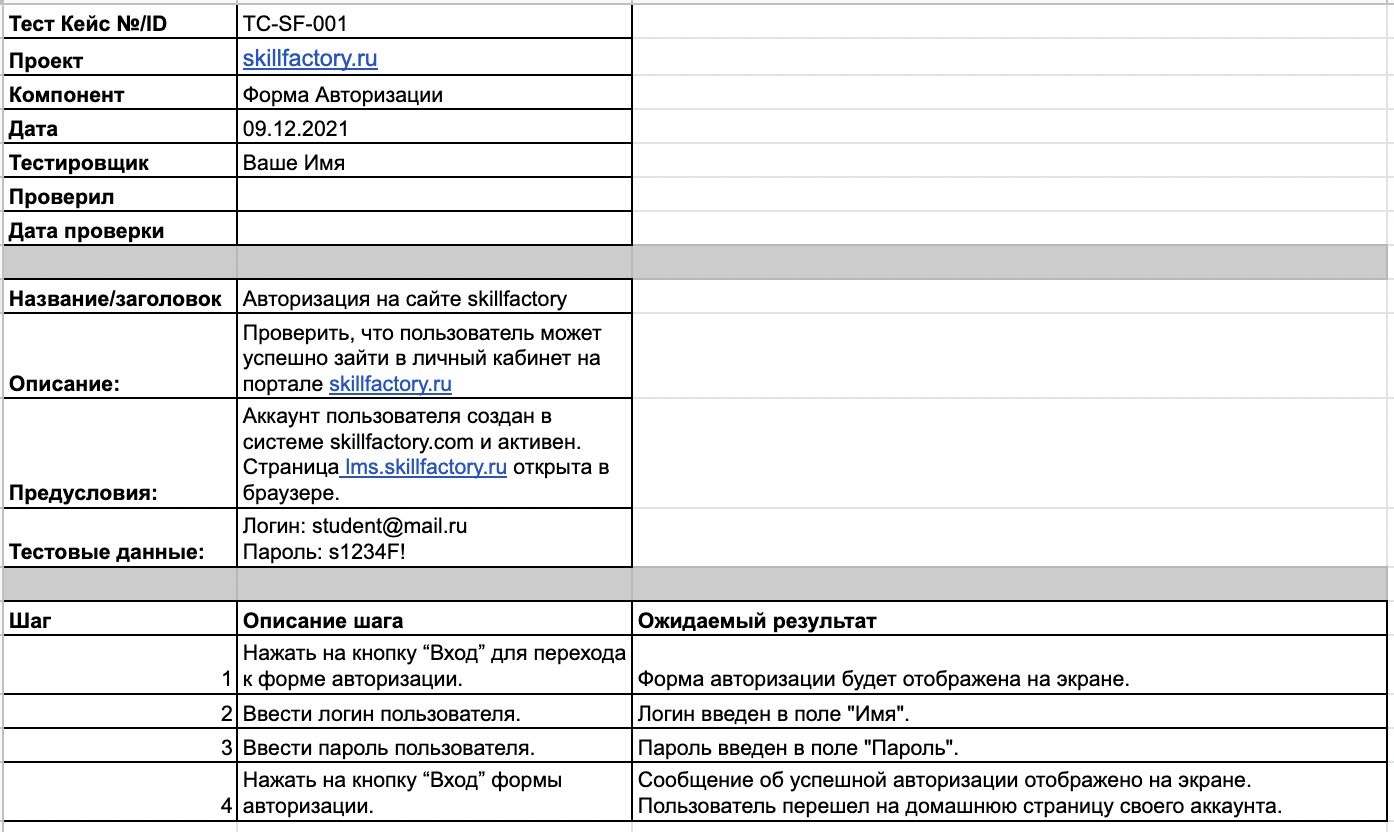
**Конечный результат** — это описание того, что происходит с системой после выполнения шагов. Этот атрибут похож на ожидаемый результат.

Например, для тест-кейса Авторизация в системе конечным результатом будет Открыта страница пользователя или Открыта главная страница, а ожидаемым результатом — Выполнен вход в систему.



Что же неправильного и плохого в этом тест-кейсе?

* 1. Отсутствует уникальный идентификатор (номер) тест-кейса. Невозможно понять, как его найти, уникальный ли он и к чему относится.
  2. Хотя мы понимаем, что проект называется *Skillfactory*, у нас отсутствует какая-либо информация о том, как зайти на этот портал, какой *URL*, какой модуль/компонент мы будем тестировать и так далее.
  3. Нет чёткого описания тест-кейса. «Проверить личный кабинет на портале *Skillfactory*» — это слишком размытое описание. Что конкретно мы хотим проверить?
  4. Предусловия отсутствуют. Совсем не ясно, что нужно подготовить перед тем, как проверять функционал.
  5. Отсутствуют какие-либо тестовые данные, и из-за этого в тестовых шагах не понятно, что такое логин? Какой логин мы вводим? Какой пароль? Активный ли аккаунт этого пользователя у нас в системе?
  6. Отсутствует ожидаемый результат. Не совсем понятно, для чего мы делали это, что хотели проверить и что ожидали увидеть на экране. Выполнение этих шагов без ожидаемого результата не несёт никакой ценности для проекта и проверки функционала.



В этом примере у нас есть следующие поля:

* 1. **Тест Кейс №/ID** — это уникальный идентификационный номер тест-кейса.
  2. **Проект** — название вашего проекта\приложения или ссылка на него.
  3. **Компонент** — компонент или часть системы, к которому относится этот тест-кейс.
  4. **Дата** — дата написания тест-кейса.
  5. **Тестировщик** — имя и фамилия тестировщика.
  6. **Проверил (опционально)** — имя и фамилия *QA*-менеджера, который проверил тест-кейс.
  7. **Дата проверки (опционально)** — дата проверки тест-кейса *QA*-менеджером.
  8. **Название заголовка**— отражает суть проверки.
  9. **Описание** — описание тест-кейса, что конкретно мы проверяем данным тест-кейсом.
  10. **Предусловия** — необходимые начальные условия (аккаунт пользователя создан, домашняя страница приложения открыта в браузере и так далее).
  11. **Тестовые данные** — ваши тестовые данные (пароль, логин, имя, фамилия, номер счета, город и так далее).
  12. **Шаг** — номер шага по порядку от 1 (1, 2, 3 и так далее).
  13. **Описание шага** — описание действий, которые мы выполняем.
  14. **Ожидаемый результат** — ожидаемое состояние системы, исходя из спецификации (откроется страница, будет выведено сообщение об ошибке на экран и так далее). Для данного шаблона ожидаемый результат последнего шага совмещён **с постусловием**.

Можно ли объединять несколько тест-кейсов вместе? Конечно! **Тестовый набор** (Test Suite) или тестовый комплект — это набор из нескольких тест-кейсов для компонента или системы, где постусловие одного тест-кейса может быть использовано как предусловие для следующего тест-кейса. Тест-кейсы объединяют в тест-наборы для большего удобства при прохождении тест-кейсов. Тест-кейсы можно объединять по типу тестирования, по приоритету, по тестируемой функциональности или модулю.

**Тест-кейс** (Test Case) — это набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определённой цели или тестового условия, таких как выполнение определённого пути программы или же для проверки соответствия определенному требованию. Другие названия: тестовый случай, тестовая ситуация, вариант тестирования.

**Позитивный тест-кейс** проверяет, что при использовании корректных данных приложение правильно выполнило функцию в соответствии со спецификацией.

**Негативный тест-кейс** проверяет, что при использовании как корректных, так и некорректных данных вызываемая функция не выполняется, и срабатывает валидация.

**Система управления тестовой документацией** (Test Management Software) — это специальная программа для создания и выполнения тест-кейсов, создания тест-планов и планирования тестовой деятельности, а также создания и хранения отчётов о тестировании.

**Тестовый набор/комплект** (Test Suite) — несколько (набор) тест-кейсов, объединённых между собой по типу тестирования, по приоритету, по тестируемой функциональности или модулю.

**Тест сценарий** — в ISTQB также можно найти термины тест сценарий (test scenario) и спецификация тестовой процедуры (test procedure specification), которые обозначают документ с последовательностью действий для выполнения тестов, также известные как тестовый сценарий (test script) или ручной тестовый сценарий (manual test script). Некоторые из этих терминов выглядят одинаково при переводе на русский язык, что может создать путаницу.

## 4.3. Чек-лист как артефакт тестирования и интеллект-карты

**Интеллект-карта** (*mind map*) известная также как карта мыслей, ассоциативная карта или диаграмма связей — метод структуризации концепций в виде схемы, на которой изображены слова, идеи, задачи или другие понятия, связанные ветвями, отходящими от центрального понятия или идеи.

**Чек-лист** — это тестовый артефакт, содержащий список проверок. Простыми словами, чек-лист — это список шагов, действий или перечень функциональности, который позволяет тестировщику убедиться в корректной работе приложения.

**Тестирование на основе чек-листа** — это одна из техник тест-дизайна на основе опыта, когда опытный тестировщик использует лист, который содержит список правил или проверок, на основе которых можно протестировать данный продукт.

**Веб-приложение** (*web*) — приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

**Десктопное** (*desktop*) приложение — это программа, которая способна работать на компьютере, используя только его аппаратные ресурсы (код самого приложения находится локально на рабочем компьютере).

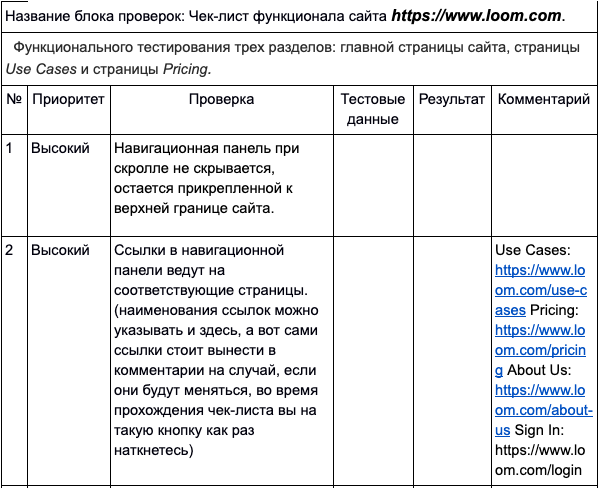
**Agile** — обобщающий термин для целого ряда подходов и практик, основанных на ценностях Манифеста гибкой разработки программного обеспечения и 12 принципах, лежащих в его основе. К гибким методологиям, в частности, относят *Scrum*, *Kanban*, экстремальное программирование (*XP*) и другие.

**RUP** (*Rational Unified Process*) — методология разработки программного обеспечения, созданная компанией *Rational Software*. Была разработана для разработки больших программных систем.

**Преимущества чек-листов в сравнении с тест-кейсами:**

* можно быстро написать и легче поддерживать;
* удобно использовать, если нет требований и спецификаций к проекту;
* позволяет тестировщику действовать более гибко;
* списки проверок могут быть выстроены в порядке приоритета;
* визуально отображает выполненные проверки.

Шаблон чек-листа, составленного в виде таблицы, приведён ниже:



Теперь поясним назначение каждой из колонок ↓

* Колонка **Приоритет** помечает приоритет проверки. Как правило, можно использовать три уровня приоритетов (от высокого до низкого). Количество уровней приоритетов может зависеть от самого тестировщика и его опыта, используемого шаблона или приложения для создания чек-листов.
* Колонка **Проверка**содержит выполняемый тест и ожидаемый результат: например, проверить, что пользователь авторизуется в системе при предоставлении корректных *e-mail* и пароля.
* Колонка **Тестовые данные** содержит данные, которые нужно использовать для проверки.
* Колонка **Результат** содержит результат выполнения теста (прошёл/не прошёл).
* Колонка **Комментарии** содержит дополнительную информацию для тестировщика и команды разработки.

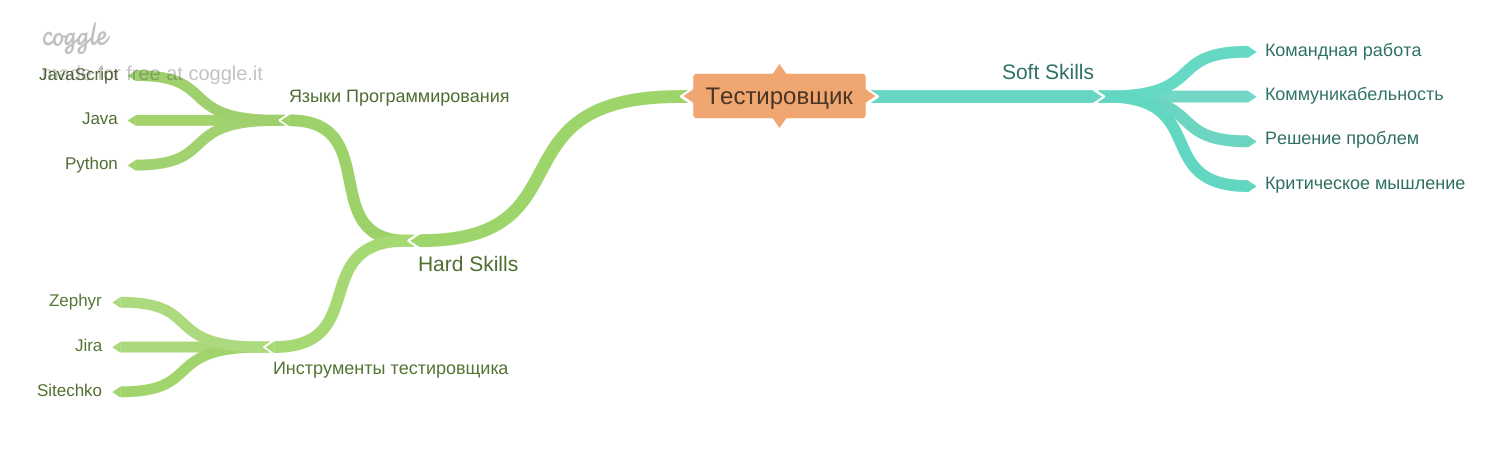
**Преимущества работы с чек-листами:**

1. Меньше времени тратится на их разработку и поддержку, а значит экономится бюджет проекта.
2. Чек-листы удобно использовать, если нет требований и спецификаций к проекту.
3. Чек-листы предоставляют тестировщику визуальное отображение выполненных проверок.
4. В *Agile*-методологиях разработки, по сравнению с «тяжёлыми» методологиями типа *RUP*, меньше времени уделяется документированию, и, так как итерация разработки в *Agile* может быть достаточно короткой и команды разработки, как правило, небольшие,  то времени на разработку полноценной тестовой документации может попросту не быть.
5. При отсутствии документации проекта или при её ограниченном объёме, чек-лист может быть хорошим примером документации для обучения новых *QA* на проекте.

**Недостатки работы с чек-листами:**

1. Опыт тестировщика. Начинающим тестировщикам будет достаточно сложно работать с чек-листами. Это особенно важно, если  в команде больше одного тестировщика.
2. На крупных проектах, в особенности там, где используется методология разработки RUP, или на проектах с высокими бизнес-рисками, использование чек-листов может скомпрометировать качество разработки приложения.
3. В сложных системах и приложениях не всегда легко создать чек-лист для тестирования.
4. Не всегда легко оценить покрытие, и некоторые из элементов чек-листа могут относиться к одному и тому же функционалу, из-за чего тестировщик будет проверять одно и тоже много раз.

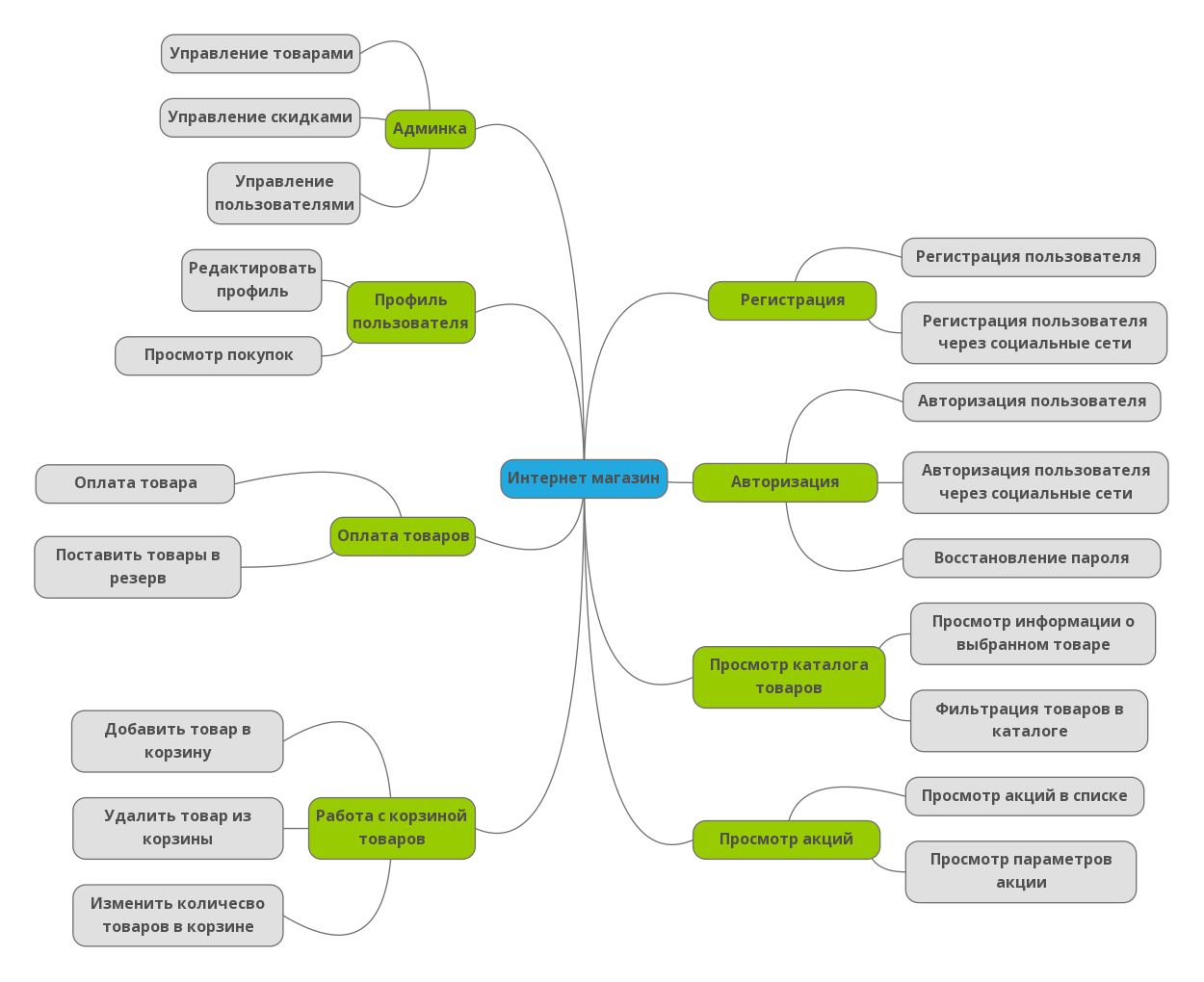
**интеллект-карта**.



Интеллект-карта — это методика, служащая для представления информации в организованной и структурированной форме. Информация может быть любой: приложение, процесс, идеи. Интеллектуальная карта покажет взаимодействие между частями целого, позволит отследить логические связи между частями, детализировать функцию каждой части.

Составленная для приложения, интеллектуальная карта позволит посмотреть на всё приложение целиком, увидеть взаимодействие между его компонентами, выделить функции каждого из компонентов, отметить логические связи между функциональностями различных компонентов, добавить параметры, которыми оперирует каждая из функций.

Как вы понимаете, умение составлять интеллектуальные карты очень важно для тестировщика, ведь ему необходимо покрыть тестами все пути работы приложения.



В центре карты показана сама система, по часовой стрелке, начиная с правой стороны, добавлены ветви, заканчивающиеся блоками, которые представляют компоненты системы или элементы первого уровня. Каждый из блоков поделён на элементы второго уровня, которые являются функциональностями компонентов. Дополнительные взаимосвязи между компонентами показаны дополнительно линиями со стрелками.

**Краткие правила составления интеллектуальных карт:**

* Объекты одного уровня, имеющие одного и того же родителя, разделяются по одному признаку. В сумме эти объекты должны составлять единое целое. Например, объекты первого уровня могут быть компонентами системы.
* Объекты второго уровня могут быть функциями компонентов и так далее. Так, раз за разом добавляя новый уровень, можно дойти до полного представления системы на интеллектуальной карте.

Однако при работе с интеллектуальными картами существует **ряд ограничений:**

* Система должна быть обозрима, поэтому стоит ограничить количество элементов на одном уровне интеллектуальной карты, а также количество уровней детализации. Оптимальное число — это 7∓2 элемента, поэтому самое сложное при составлении интеллектуальных карт — поддерживать баланс между полнотой и простотой системы.
* При работе над большим и сложным проектом в команде тестировщиков различных уровней очень сложно составить карту, одинаково читабельную как для начинающих, так и для опытных тестировщиков.

Существует множество бесплатных и платных **приложений для работы с интеллектуальными картами**. Они различаются по наличию или отсутствию дополнительных функций, интеграций с различными системами. Ниже приведено несколько таких приложений, которые можно эффективно использовать в своей работе:

* [Coggle](https://coggle.it/) — бесплатный план веб-приложения имеет ограничения: можно создать и хранить не более трёх карт.
* *MindMap for Google Drive* — бесплатное веб-приложение от *Google*. Преимущество — все артефакты можно держать на *Google Диске* и давать к ним доступ членам команды.
* [MindMeister](https://www.mindmeister.com/) — бесплатный план веб-приложения имеет ограничения: можно создать и хранить не более трёх карт. Небольшая абонентская плата позволяет расширить возможности: добавить стили, командную работу.
* [XMind](https://www.xmind.net/) — бесплатное (с ограничениями) кроссплатформенное десктоп-приложение.

## 4.4. План тестирования

**Тест-план** или **план тестирования** — это тестовый артефакт, рекомендуемый стандартом тестовой документации [ГОСТ Р 56922-2016/ISO/IEC/IEEE 29119-3:2013](http://docs.cntd.ru/document/1200134998).  
  
Согласно этому стандарту, **план тестирования** — это документ описывающий весь объём работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания, конфигурации, знаний и навыков тестировщиков, а также оценки рисков и вариантов их разрешения.

**Что может быть задокументировано в плане тестирования?**

* + Определение объёма, целей и рисков тестирования,
  + Определение общего подхода к тестированию,
  + Интеграция и координация действий по тестированию в действия жизненного цикла программного обеспечения,
  + Принятие решений, что тестировать, о людях и других ресурсах, необходимых для выполнения различных тестовых действий, и как будут выполняться тестовые действия,
  + Планирование деятельности по анализу, проектированию, внедрению, выполнению и оценке тестов либо на определённые даты (например, в последовательной разработке), либо в контексте каждой итерации (например, в моделях разработки типа *Agile*/*SCRUM*),
  + Выбор метрик для мониторинга и контроля тестирования,
  + Бюджет тестовой деятельности,
  + Определение уровня детализации и структуры тестовой документации (например, путём предоставления шаблонов и примеров документов),
  + Критерии входа (*Entry Criteria*) и выхода (*Exit Criteria*) (желательно иметь критерии, определяющие, когда тестирование должно начинаться и когда оно завершается).

Типичные **критерии входа** включают в себя:

* + Наличие тестируемых требований, пользовательских историй (*user stories*) и/или моделей (например, при следовании стратегии тестирования на основе моделей),
  + Доступность элементов тестирования, соответствующих критериям выхода для любых предыдущих уровней тестирования,
  + Доступность тестовой среды,
  + Наличие необходимых инструментов тестирования,
  + Наличие тестовых данных и других необходимых ресурсов.

Типичные **критерии выхода** включают:

* + Запланированные тесты выполнены,
  + Достигнут определённый уровень покрытия (например, требований, пользовательских историй, критериев приемлемости, рисков, кода),
  + Количество неустранённых дефектов находится в пределах согласованного лимита,
  + Количество предполагаемых оставшихся дефектов достаточно мало,
  + Оценённые уровни надёжности, эффективности работы, удобства использования, безопасности и других соответствующих характеристик качества являются достаточными.

**Мастер тест план** содержит следующие разделы, согласно ГОСТ-Р 56922-2016:

1

**Описание объема работ по тестированию**.  
Что будет протестировано, что будет исключено из тестирования. Например, приложения, как правило, интегрируются с внешними сервисами, и важно спроектировать тесты таким образом, чтобы тестировать приложение и его интеграцию с сервисами, а не сами внешние сервисы.

2

**Ссылки на внутренние и внешние документы проекта**.  
Внутренние документы проекта — это план проекта, план качества проекта, план управления конфигурацией.  
Внешние документы проекта — это могут быть различные стандарты, регламенты и положения законодательства. Для правильного построения процесса тестирования и выполнения особых видов тестирования необходимо учитывать эти документы.

3

**Положения, касающиеся организации процесса тестирования на проекте**.  
К ним относится время проведения работ по тестированию, согласованное с планом разработки и жизненным циклом разрабатываемой системы.

4

**Краткое описание требуемых ресурсов**.К ним относятся: требуемый персонал, инструменты, процедурные требования (требуемый доступ к системе и документации, общие требования безопасности). Например, при тестировании банковских приложений могут потребоваться тестировщики со знаниями в этой области или тестировщики с опытом тестирования защищенности приложений.

5

**Ответственные лица за конкретные задачи по тестированию**.

К ним относятся: требуемый персонал, инструменты, процедурные требования

6

**Описание тестового окружения, инструментов, метрик, получаемых по результатам тестирования**.  
Определение уровней тестирования, задач по тестированию на каждом из уровней, подходы, используемые для выполнения каждой из задач, инструментов, ресурсы, необходимые для выполнения задачи, и ожидаемые результаты выполнения каждой задачи. Например, какие библиотеки будем использовать для написания интеграционных тестов, что будем покрывать ими, какая будет стратегия, какие метрики будем использовать.

7

**Определение требований к тестовой документации**. Например, какие тестовые документы будем составлять, где хранить и как будем управлять этим процессом.

8

**Требования к управлению процессом тестирования**.

**Что такое детальный тест-план**

**Детальный тест-план** описывает план тестирования на определённом уровне тестирования или для определённого вида тестирования, например, производительности. Детальный тест-план включает в себя следующие пункты, согласно стандарту ГОСТ-Р 56922-2016:

* + Объём работ по тестированию. Описание используемых тестовых данных, валидных, невалидных, граничных.
  + Описание объектов тестирования и тестового базиса.
  + Функциональности, которые будут и не будут протестированы.
  + Подходы, которые будут использованы при разработке тест-кейсов для данного плана тестирования. Например, методы тестирования чёрного ящика, белого ящика, подходы к рецензированию кода.
  + Критерии прохождения и не прохождения тестов.
  + Критерии остановки, возобновления тестирования. Например, если количество не прошедших тестов более 30%, то тестирование приостанавливается для выяснения причин.
  + Описание тестовых артефактов, которые будут созданы в ходе работ по тестированию.
  + Описание основных активностей тестирования, таких как подготовка, выполнение тестов.
  + Описание требуемых окружений и инфраструктуры для выполнения тестов.
  + Детализация требуемых ресурсов, описанных в мастер тест плане. Например, необходимые навыки тестировщиков, чтобы выполнить работы по тестированию, описанные в данном тест плане, требуемая подготовка персонала.
  + Сроки, оценка и стоимость.
  + Оценка рисков.
  + Определение способов, которыми качество работ по тестированию и качество тестируемого приложения будет обеспечено.
  + Определение собираемых метрик.
  + Определение подхода к обеспечению покрытия тестами.
* **Тест-план** (План тестирования) — это документ, описывающий весь объём работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания, конфигурации, знаний и навыков тестировщиков, а также оценки рисков и вариантов их разрешения.
* **Критерии входа** (*Entry Criteria*) и **критерии выхода** (*Exit Criteria*) — критерии, определяющие, когда тестирование должно начинаться и когда оно завершается.
* **Мастер тест план** — высокоуровневый (не детальный) документ, описывающий объём работ по тестированию, ссылки на внутренние и внешние документы, описание требуемых ресурсов и так далее.
* **Детальный тест-план** — детальный документ, который описывает план тестирования на определённом уровне тестирования или для определённого вида тестирования.

## 4.5. Отчёт по тестированию

**Отчёт по тестированию**— документ, содержащий в себе информацию о выполненных тестах, результатах их выполнения, открытых дефектах и выводах о качестве разрабатываемого приложения. Отчёт по тестированию тесно связан с таким понятием, как фаза тестирования.

* **Фазы тестирования** или **жизненный цикл тестирования приложений** (*STLC — Software Test Life Cycle*) *—* это последовательность действий, проводимых в процессе тестирования, с помощью которых гарантируется качество программного обеспечения и его соответствие требованиям. Обычно разбиты на фазы.
* **Базис тестирования** (тестовая база) *—* это различные требования, спецификации, архитектурные диаграммы, спецификации интерфейсов, сценарии использования, отчёты по анализу рисков. Тестовым базисом может быть даже исходный код.

**Фазы тестирования** или **жизненный цикл тестирования приложения** (*STLC*) — это определённый набор задач, объединённых в контролируемую фазу проекта. В тестировании можно выделить следующие фазы, которые коррелируются с жизненным циклом продукта:

* + фаза планирования,
  + фаза анализа,
  + фаза проектирования тестов,
  + фаза реализации и выполнения тестов,
  + фаза анализа результатов тестирования (оценка критериев выхода) и отчётов,
  + фаза завершения тестирования.

Также, начиная с самой первой фазы тестирования (планирование), присутствует фаза тест-контроля. **Тест контроль** — это постоянная активность, потому что нам надо сравнивать фактический прогресс с запланированным прогрессом, а также предоставлять отчёты о текущем состоянии тестирования (включая любые изменения или отклонения от плана) менеджеру проекта и заказчикам. А также принимать меры по корректировке, чтобы двигаться в соответствии с планом.

Рассмотрим состав активностей на каждой фазе на примере разработки приложения «Дом питомца».

1

**Планирование тестирования** состоит из действий, которые определяют цели тестирования и подход к достижению целей тестирования с ограничениями, налагаемыми контекстом. Во время планирования тестирования мы удостоверимся, что понимаем цели и задачи заказчика, заинтересованных сторон, проекта, а также риски, связанные с тестированием. Например, мы можем определить подходящие методы тестирования, задачи, которые необходимо решить, сформировать график тестирования. То есть выполнить все те процедуры, которые зафиксированы в плане тестирования (мы детально рассмотрели его в предыдущем уроке).  
  
В проекте «Дом питомца» на этой фазе тестировщик будет работать с требованиями, планировать тестирование и составлять требования к тестовому окружению, тестовым данным, разрабатывать стратегию тестирования.

2

**Анализ и дизайн (проектирование) тестов** — это деятельность, при которой общие цели тестирования преобразуются в реальные тест-условия и тест-дизайн, а также создаются высокоуровневые тест-кейсы.  
  
Анализ тестирования состоит из анализа базиса тестирования для определения тестируемых функций и установления соответствующих тестовых условий. Другими словами, анализ тестирования решает, «что тестировать», с точки зрения измеримых критериев покрытия. Как известно, базис тестирования — это различные требования, спецификации, архитектурные диаграммы, спецификации интерфейсов, сценарии использования, отчёты по анализу рисков. Тестовым базисом может быть даже исходный код. Каждый из этих артефактов анализируется на целостность, полноту, непротиворечивость, возможную избыточность.

На фазе анализа тестировщик также продолжит работу с требованиями проекта «Дом питомца», анализируя и тестируя их. По замечаниям тестировщика требования, возможно, будут обновляться и будут вноситься коррективы в план тестирования.

3

**Проектирование тестов** — это создание высокоуровневых тестовых сценариев. Другими словами, проектирование тестов решает вопрос «как тестировать». Помимо тестовых сценариев определяются необходимые тестовые данные, требования к тестовому окружению.

На этой фазе тестировщик проекта «Дом питомца» разрабатывает тест-кейсы для тестирования приложения, определяет требования к окружению для выполнения тестов, например, что должно быть развернуто там, какие внешние интеграции должны быть настроены, доступ к системе. Также для проекта подбираются тестовые данные, например, какая информация и о каких питомцах должна быть загружена в систему.

4

**Реализация и выполнение тестов** — это создание и/или подготовка необходимого тестового обеспечения для выполнения тестов, включая упорядочение тестовых сценариев в процедурах тестирования с использованием высокоуровневых тест-кейсов, которые мы создали в предыдущей фазе. Таким образом, реализация тестов отвечает на вопрос: «у нас теперь есть всё для запуска тестов?». Реализация тестов — это активность, во время которой процедуры или сценарии тестирования выстраиваются в определённом порядке, чтобы облегчить выполнение тестов: подготовка тестовых данных, составление наборов тестов, построение тестового окружения.

Для проекта «Дом питомца» на этой фазе будет заполнена база данных с картинками и информацией о питомцах. Тест-кейсы будут собраны в тестовые наборы и назначены на исполнителей. Будет настроено тестовое окружение для выполнения тестов.

Во время **фазы выполнения тестов**, тесты выполняются согласно расписанию тестирования, сравниваются результаты тестирования с ожидаемым результатом, происходит составление лога тестирования.

Для проекта «Дом питомца» это означает выполнение тестов после того, как готовое для тестирования приложение будет развёрнуто на тестовом окружении.

5

Во время фазы **анализа результатов тестирования (оценки критериев выхода) и отчётов**составляется отчёт по тестированию, анализируются его результаты. Оценка критериев выхода — это деятельность, в которой выполнение тестов сравнивают с поставленной целью. Это нужно сделать для каждого уровня тестирования, так как для каждого из них мы должны знать, сделали ли мы достаточно тестирования. Чтобы понять, что такое «достаточно», мы можем использовать критерии покрытия или критерии приёма.

Также нам предстоит выяснить, нужны ли нам дополнительные тест-кейсы или нужно изменить критерии выхода. Когда это делается? Например, мы не успели выполнить все запланированные тест-кейсы или же не достигли ожидаемого покрытия. Для проекта «Дом питомца» это значит сравнить результаты тестирования с критериями завершения тестирования, определённые в плане тестирования, включить в отчёт метрики плана тестирования.

6

**Фаза завершения тестирования** — это фаза, на которой анализируются результаты тестирования и всё тестовое обеспечение передаётся заказчику и команде сопровождения. Эта фаза обычно присутствует тогда, когда завершается проект. В качестве исключения эта фаза может присутствовать, когда проект отменили, когда мы достигли конкретного этапа или же когда выполняется отладочный выпуск или обновление ПО. В нашем случае это значит, что тестировщик на проекте «Дом питомца», должен будет подготовить и передать заказчику все артефакты тестирования.

**Отчёт по тестированию** — это документ, формат которого определяется стандартом [ГОСТ Р 56922-2016/ISO/IEC/IEEE 29119-3:2013](http://docs.cntd.ru/document/1200134998). Согласно данному стандарту, цель отчёта по тестированию — это консолидация результатов выполнения тестов для последующего анализа и формирования заключений по результатам тестов.

Отчёт по тестированию содержит следующую информацию:

* + спецификацию документа,
  + сводка выполненного тестирования,
  + отклонения от Плана Тестирования,
  + оценка завершения тестирования,
  + препятствующие факторы,
  + показатели тестирования,
  + остаточные риски,
  + практические результаты тестирования,
  + активы тестирования, допускающие повторное использование,
  + накопленный опыт.

## 4.6. Тест-дизайн. Метод «чёрного ящика»

**Функциональное тестирование** — это тестирование программного обеспечения в целях проверки реализуемости функциональных требований.

**Нефункциональное тестирование** — это тестирование программного приложения или системы на предмет их нефункциональных требований: способа работы системы, а не конкретного поведения этой системы. Например, тестирование безопасности, тестирование удобства пользования, нагрузочное тестирование и так далее.

**Базис тестирования** (тестовая база/тест-базис) — это различные требования, спецификации, архитектурные диаграммы, спецификации интерфейсов, сценарии использования, отчёты по анализу рисков. Тестовым базисом может быть даже исходный код.

**Техники чёрного ящика** (*black box*) — техники, основанные на спецификации. Чёрным ящиком он называется потому, что они рассматривают программное обеспечение как чёрный ящик с входом и выходом, но они не знают, как система или компонент структурирован внутри коробки.

**Техники белого ящика** (*white box*) — техники, основанные на структуре (которые также являются динамическими, а не статическими), используют внутреннюю структуру программного обеспечения для создания тест-кейсов. Их называют техниками белого ящика или стеклянного ящика (подразумевая, что вы можете заглянуть внутрь системы), так как они требуют знаний о том, как реализовано программное обеспечение, как оно работает.

**Техники, основанные на опыте** — техники, которые используют знания, навыки и опыт людей (тестировщиков, бизнес аналитиков, пользователей и так далее), являющихся основным фактором, влияющим на условия тестирования и тест-кейсы.

**Попарное тестирование** (*pairwise*) — тест-дизайн методом чёрного ящика, в котором в тест-кейсах выполняют все возможные отдельные комбинации каждой пары входных параметров.

**RUP** (*Rational Unified Process*) — методология разработки программного обеспечения, созданная компанией *Rational Software*. Была разработана для разработки больших программных систем.

**Тестовое покрытие** — степень, выраженная в процентах, до которой указанный элемент покрытия был протестирован тестовым набором.

**техники тестирования черного ящика** — это техники, опирающиеся на анализ соответствующего базиса тестирования (формальных требований, спецификаций, сценариев использования, пользовательских историй или бизнес-процессов).

Техники чёрного ящика имеют следующие **характеристики**:

* + Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя требования, спецификации, сценарии использования и пользовательские истории.
  + Тестовые сценарии могут использоваться для определения несоответствий и отклонений между требованиями и реализацией.

Стоит отметить, что перечисленные техники тест-дизайна методом чёрного ящика применимы для проектирования тестов на всех уровнях тестирования, в отличие от техник тестирования белого ящика. Так, методы статического анализа кода применяются, как правило, только на компонентном и интеграционном уровнях тестирования.

*ISTQB* выделяет 4 уровня тестирования:

* 1. *Unit* (компонентное) тестирование,
  2. Интеграционное тестирование,
  3. Тестирование системы,
  4. Приёмочное тестировани

## 4.7. Что проверяем в первую очередь