## Лекция №10 Тестовая документация

### **Введение**

**Тестовая документация -**это документация, создаваемая тестировщиками, которая помогает в выполнении различного рода активностей в рамках тестирования программного обеспечения.

Тестовая документация для планирования тестирования:

* [Тест план](https://dist.belstu.by/mod/assign/view.php?id=57446)
* График тестирования
* Матрица устройств
* Матрица прослеживаемости
* Тестовый набор
* [Тест сценарии](https://dist.belstu.by/mod/assign/view.php?id=57444)
* Тест кейсы
* Чеклист

Тестовая документация для отчетности:

* Отчеты об ошибках
* Отчеты о результатах тестирования

### **Тестовая документация для планирования**

#### [**Тест план**](https://dist.belstu.by/mod/assign/view.php?id=57446)

Документ, описывающий весь объем работ по тестированию, начиная с описания тестируемых объектов, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.

**Цели написания**[**тест план**](https://dist.belstu.by/mod/assign/view.php?id=57446)**а:**

* Продумать стратегию тестирования программного обеспечения;
* Описать процесс тестирования на проекте, и как он встраивается в процесс разработки;
* Обеспечить информированность каждого члена команды об активностях QA команды, распределении обязанностей и зон ответственности;
* Скорректировать ожидания заказчика от команды тестирования.

#### **График тестирования**

Документ описывающий последовательность выполнения активностей по тестированию членами QA команды, с указанием дат начала выполнения работ и их завершения.

**Цели:**

* Согласовать работу команды разработки и тестирования;
* Обеспечить информированность каждого члена команды о последовательности задач, а также о сроках их выполнения;
* Обеспечить прозрачность процесса тестирования для заказчика;
* Обеспечить возможность отслеживания отставаний от плана и влияния добавления дополнительных задач команде.

#### **Матрица устройств**

Документ, используемый на проектах, где разрабатываются мобильные приложения, который описывает конфигурацию устройств, где будет проводиться тестирование.

**Задачи, которые решает документ:**

* Определить оптимальное количество устройств, необходимых для тестирования;
* Убедится, что устройства с различными версиями необходимого оборудования участвуют в тестировании;
* Сосредоточить усилия тестировщика на устройствах, оптимально полно покрывающих требования проекта;
* Обеспечить прозрачность тестирования для заказчика.

#### **Матрица трассировки**

Документ, используемый для определения покрытия требований проверками, оформленными в виде соответствующей тестовой документации.

Матрица представляет собой таблицу, где соотносятся ID требований с ID чеклистов, тест кейсов, тест сценариев и другой документации, которая используется на проекте. **Цель**- обеспечить должное покрытие всех функциональных и нефункциональных требований тестами.

#### **Тестовый набор**

Документ, вмещающий в себя набор тестов/тестовых случаев/тестовых сценариев. Тестовый набор может быть сформирован для определенного вида тестирования, уровня тестирования или для компонента с определенным приоритетом.

**Процесс создания тестового набора:**

1. Тестовые случаи в виде соответсвующих документов оформляются в системе управления тестовой документацией.
2. У каждого тестового случая проставляется label (отметка).
3. В соответствии с необходимостью, по определенному label фильтруются все проверки и в группу попадают только нужные.

Один тестовый случай может иметь несколько labels и может быть в нескольких тестовых наборах.

#### **Тест сценарий**

Документ, описывающий последовательность шагов, которые необходимо выполнить тестировщику, с целью выявления дефектов в приложении.

**Виды тестирования, для которых пишут сценарии:**

* [Исследовательское тестирование](https://dist.belstu.by/mod/assign/view.php?id=57436);
* End-to-end тестирование;
* Тестирование, основанное на ролях.

#### **Тест кейс (тестовый случай)**

Документ, описывающий последовательность шагов и ожидаемый результат. Документ направленный на проверку атомарных требований.

**Виды тестирования, для которых пишут тест кейсы:**

* [Позитивное тестирование](https://dist.belstu.by/mod/assign/view.php?id=57434);
* [Негативное тестирование](https://dist.belstu.by/mod/assign/view.php?id=57435);
* CRUD тестирование.

#### **Чеклист**

Документ, перечисляющий идеи для проверки. Документ, который очень поверхностно указывает, что необходимо проверить в приложении, но не указывает, как это сделать.

**Виды тестирования, для которых пишут чеклисты:**

* Тестирование совместимости;
* Тестирование инсталляции продукта;
* [Исследовательское тестирование](https://dist.belstu.by/mod/assign/view.php?id=57436);
* CRUD тестирование
* Тестирование на прерывания и т.д.

### **Тестовая документация для отчетности**

#### **Отчет об ошибке (bug report)**

Документ, описывающий проблему, найденную в процессе тестирования в приложении.

**Цели**:

* Предоставить информацию о проблеме:
* уведомить проектную команду и иных заинтересованных лиц о наличии проблемы, описать суть проблемы;
* Приоритизировать проблему:
* определить степень опасности и желаемые сроки её устранения;
* Содействовать устранению проблемы:
* предоставить необходимые подробности для понимания сути случившегося, а также анализ причин возникновения проблемы и рекомендации по исправлению ситуации.

#### [**Отчет о результатах тестирования**](https://dist.belstu.by/mod/assign/view.php?id=57445)

Документ, описывающий результаты проведенного уровня или вида тестирования, который дает ясное понимание о количестве и серьезности найденных ошибок, количестве пройденных успешно и неуспешно тестов, а также о качестве самого приложения.

**Цели:**

* Предоставить статистику по количеству проверок и результатам их выполнения;
* Предоставить статистику по количеству и серьезности найденных дефектов в приложении;
* Повысить осведомленность команды о качестве текущей версии приложения;
* Обратить внимание команды и владельцев бизнеса на области, которые нуждаются в исправлении дефектов
* Дать рекомендации относительно выпуска или наоборот задержки выпуска версии в релиз.

## Лекция №11 Техники тест дизайна

### **Введение**

Во время анализа тестирования базис тестирования анализируется с целью определить, что будет тестироваться, то есть определяются тестовые условия.

Тестовые условия определяются как сущности или события, которые будут проверяться одним или несколькими тестовыми сценариями (например, функция, транзакция, характеристика качества или структурный элемент).

Установление трассируемости от тестовых условий назад к спецификациям и требованиям позволяет как определить последствия при изменяющихся требованиях, так и установить покрытие требований набором тестов. При анализе тестирования реализуется детализированный подход к тестированию с целью выбора методов проектирования тестов, основываясь, в том числе, на выявленных рисках.

Во время проектирования тестов определяются и специфицируются тестовые сценарии и тестовые данные.

Тестовый сценарий состоит из набора входных значений, предусловий выполнения, ожидаемых результатов и постусловий, определяемых для покрытия определенных тестовых условий (или тестового условия) или целей (цели) тестирования. Содержание спецификаций проектирования тестов (включая тестовые условия) и спецификаций тестовых сценариев описывается в стандарте «Документация при тестировании программ» (IEEE STD 829-1998).

Ожидаемые результаты должны создаваться как часть спецификаций тестовых сценариев и включать в себя выходные данные, изменения в данных и состояниях, и любые иные последствия теста. Если ожидаемые результаты не были определены, правдоподобные, но ошибочные результаты могут быть приняты за корректные.

В идеальных условиях ожидаемые результаты должны быть определены до момента выполнения теста.

Во время реализации теста тестовые сценарии разрабатываются, реализуются, получают приоритеты и формируют спецификацию процедуры тестирования (IEEE STD 829-1998). Процедура тестирования (или ручной сценарий тестирования) описывает последовательность действий для выполнения теста.

Если тесты запускаются с использованием инструмента выполнения тестов, последовательность действий описывается автоматизированным тестовым сценарием.

Различные процедуры тестирования и автоматические тестовые сценарии собираются в расписание выполнения тестов, определяющее, в какой очередности, когда и кем эти тестовые процедуры и сценарии должны быть выполнены.

Расписание выполнения тестов должно учитывать такие факторы, как регрессионные тесты, приоритеты и технические и логические зависимости.

### **Категории методов проектирования тестов**

Целью метода проектирования тестов является определение тестовых условий и тестовых сценариев. Классическим является разделение методов тестирования на методы черного и белого ящиков.

**Методы черного ящика** (включающие в себя методы разработки тестов на основе спецификаций и на основе опыта) – это способ определить и выбрать тестовые условия или сценарии для компонента или системы (как функциональные, так и не функциональные), на основе анализа базиса тестирования и опыта разработчиков, тестировщиков и пользователей, без отсылки к внутренней структуре компонента или системы.

**Методы белого ящика** (также называемые структурными, или основанными на структуре) основываются на анализе структуры компонента или системы. Оба этих метода могут быть объединены с методом создания тестов на основе опыта, чтобы использовать опыт разработчиков, тестировщиков и пользователей для определения того, что должно быть протестировано. Некоторые методы могут быть однозначно отнесены к определенной категории, другие же сочетают в себе несколько категорий.

Данная программа относит подходы, основанные на спецификациях или опыте к методам черного ящика, и подходы, основанные на структуре – к методам белого ящика. Также рассматриваются методы создания тестов на основе опыта.

**Общие признаки подходов, основанных на спецификациях:**

* для описания задач, которые должны быть решены, программных продуктов или их компонентов, используются модели - формальные или неформальные.
* из этих моделей систематически выводятся тестовые сценарии.

**Общие признаки подходов, основанных на структуре:**

* тестовые сценарии выводятся на основе информации о том, как спроектировано программное обеспечение (например, на основе программного кода и подробного описания проектного решения).
* для программного обеспечения может быть измерена величина покрытия для имеющихся тестовых сценариев, и последующие тестовые сценарии могут разрабатываться для систематического увеличения покрытия.

**Общие признаки методов на основе опыта:**

* для определения тестовых сценариев используются человеческие знания и опыт.
* знания тестировщиков, разработчиков, пользователей и заинтересованных лиц о программном продукте, его использовании и окружении, являются одним из источников информации.
* знания о вероятных дефектах и их распределении являются другим источником информации.

### **Методы черного ящика**

#### **Эквивалентное разбиение**

Входные данные для программного обеспечения или системы разбиваются на группы, от которых ожидается сходное поведение, то есть они должны обрабатываться аналогичным образом. Эквивалентные области (или классы) могут быть определены как для валидных, так и для невалидных данных, то есть тех значений, которые должны отвергаться. Области также могут быть определены для выходных данных, внутренних значений, значений, зависящих от времени (например, до или после некоторого события) и для параметров интерфейса (например, во время интеграционного тестирования). Тесты могут разрабатываться для покрытия всех валидных и всех невалидных классов. Эквивалентное разбиение применимо на всех уровнях тестирования.

Эквивалентное разбиение 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.

#### **Анализ граничных значений**

Поведение на границах эквивалентных областей имеет наибольшие шансы быть некорректным, таким образом границы являются потенциальным источником дефектов. Минимальные и максимальные значения сегмента являются граничными значениями. Граничное значение для валидного сегмента является валидным граничным значением, для невалидного сегмента – невалидным. Тесты могут разрабатываться для покрытия как валидных, так и невалидных граничных значений. При разработке тестовых сценариев выбирабтся тесты для каждого граничного значения.

Анализ граничных значений может применяться на всех уровнях тестирования. Он относительно легок в применении и эффективен при поиске дефектов. Для выделения интересующих нас границ крайне полезны подробные спецификации.

Данный метод часто рассматривается как дополнение к методу эквивалентного разбиения. Он может использоваться для классов эквивалентности данных, вводимых на экране, так и, например, для классов эквивалентности временных диапазонов (например, таймауты или требования по быстродействию транзакций) или для размерности таблиц (например, размер таблицы 256\*256).

#### **Тестирование таблицы решений**

Таблицы решений – хороший метод для сбора системных требований, содержащих логические условия и документирования внутреннего дизайна системы. Они могут использоваться для записи сложных бизнес-правил, которые должна реализовывать система. Анализируются спецификации и определяются условия и действия системы. Входные условия и действия чаще всего формулируются таким образом, чтобы они могли принимать логические значения «истина» или «ложь».

Таблица решений содержит триггерные условия, обычно комбинации значений «истина» и «ложь» для всех входных условий, и результирующие действия для каждой комбинации условий. Каждый столбец таблицы соотносится с бизнес-правилом, определяющим уникальную комбинацию условий и результат выполнения действий, связанных с этим правилом. Стандартом покрытия для тестирования таблицы решений обычно является наличие хотя бы одного теста для каждой колонки, что обычно включает в себя покрытие всех комбинаций триггерных условий.

Сильной стороной тестирования таблицы решений является то, что она создает комбинации условий, которые могли бы быть не проверены в ходе тестирования иным способом. Этот метод может быть применен ко всем ситуациям, в которых действие программного продукта зависит от нескольких логических альтернатив.

#### **Тестирование таблицы переходов**

Система может показывать различные отклики в зависимости от текущих условий или предшествовавшей истории состояний. Данный метод позволяет тестировщику рассматривать систему с точки зрения её состояний, переходов между состояниями, входов или событий, активизирующих изменения состояний (переходы) и действия, к которым приводят эти переходы. Состояния системы или тестируемого объекта разделяемы, определяемы и конечны.

Таблица состояний демонстрирует связи между состояниями и входами и может подсказать возможные некорректные переходы. Тесты могут разрабатываться для покрытия типовой последовательности состояний, для покрытия каждого состояния, для выполнения каждого перехода, для выполнения определенных последовательностей переходов или же для тестирования некорректных переходов.

Тестирование таблицы переходов чаще всего используется в индустрии встроенного программного обеспечения и автоматизации в целом. Однако эта методика также подходит для моделирования бизнес-объектов, имеющих определенные состояния, или тестирования последовательностей диалоговых блоков (т.е. интернет-приложений или бизнес-сценариев).

#### **Тестирование по сценариям использования**

Тесты могут базироваться на сценариях использования. Сценарий использования описывает взаимодействия между участниками (включая пользователей и систему) приводящие к полезным результатам для заказчика или пользователя системы. Сценарии использования могут быть описаны на уровне абстракций (бизнес сценарий использования, уровень бизнес-процессов, не связанный с технологией) или на системном уровне (сценарий использования системы на уровне системного функционала). Каждый сценарий использования имеет предусловия, необходимые для успешного выполнения сценария. Каждый сценарий использования завершается постусловиями, являющимися наблюдаемыми результатами и итоговым состоянием системы после выполнения сценария. Сценарий использования обычно имеет основной (наиболее вероятный) сценарий и альтернативные сценарии.

Сценарии использования описываются как «потоки процессов» в системе, основанные на типовом предполагаемом использовании. Таким образом, тестовые сценарии, построенные на основе сценариев использования, являются наиболее полезными для выявления дефектов в потоках процессов во время реального использования системы. Сценарии использования очень полезны для разработки приёмочных тестов с участием заказчика или пользователей. Также они могут выявить дефекты интеграции, вызванные взаимодействием различных компонентов, не выявляемые индивидуальным тестированием компонентов.

Проектирование тестовых сценариев на основе сценариев использования может быть объединено с другими методами, основанными на спецификации.

### **Методы белого ящика**

Тестирование на основе структуры, или тестирование методом белого ящика, основывается на конкретной структуре программного продукта или системы, как рассмотрено в следующих примерах:

* Компонентный уровень: структура компонента программного обеспечения, т.е. операторы, альтернативы, ветви или определенные пути
* Интеграционный уровень: структура может быть представлена деревом вызовов (диаграмма, в которой модули вызывают другие модули)
* Системный уровень: структура может представлять собой структуру меню, бизнес-процессов или же схему веб-страницы.

В этом разделе рассматриваются три метода проектирования тестов на основе структуры для покрытия кода: основанные на операторах, ветвях и альтернативах. Для визуализации тестирования альтернатив может использоваться диаграмма потока управления.

#### **Тестирование операторов и покрытие**

В компонентном тестировании покрытие операторов - это доля операторов, проверенных во время выполнения набора сценария тестирования. При тестировании операторов тестовые сценарии создаются таким образом, чтобы выполнять определенные операторы и обычно увеличивать покрытие операторов.

Величина покрытия операторов определяется как отношение числа выполняемых операторов, покрытых тестовыми сценариями (разработанными или выполненными) к общему числу операторов в тестируемом коде.

#### **Тестирование альтернатив и покрытие**

Покрытие альтернатив, связанное с тестированием ветвей - это доля результатов альтернатив (например, вариантов «Истина» и «Ложь» для оператора «Если»), проверенных набором сценариев тестирования. В методе тестирования альтернатив тестовые сценарии создаются для выполнения определенных результатов альтернатив. Ветви исходят из точек альтернатив в программном коде и показывают передачу управления различным участкам кода.

Покрытие альтернатив определяется отношением числа всех результатов альтернатив, покрытых разработанными или выполненными тестовыми сценариями к числу всех возможных результатов альтернатив в тестируемом коде.

Тестирование альтернатив - это вид тестирования потока управления, так как оно описывает прохождение определенного потока через точки альтернативы. Покрытие альтернатив более строгое, чем покрытие операторов: 100% покрытие альтернатив обеспечивает 100% покрытие операторов, но не наоборот.

#### **Другие методы, основанные на структуре**

Имеются и более высокие уровни покрытия структуры после покрытия альтернатив. Например, покрытие условий и покрытие множественных условий. Принцип покрытия также может быть применен на других уровнях тестирования. Например, на уровне интеграции процент модулей, компонентов или классов, проверенных набором тестовых сценарием, может быть выражен как покрытие модулей, компонентов или классов. Для структурного тестирования крайне полезна инструментальная поддержка.

### **Методы, основанные на опыте**

Когда тесты создаются на основе мастерства тестировщика, его интуиции и опыте работы с подобными приложениями или технологиями, это называется тестированием, основанным на опыте. Данный вид тестирования полезен как дополнение более систематических методов для разработки специальных тестов, не всегда очевидных при использовании более формальных методик, особенно когда используется после таких методик. Однако полезность этого метода может сильно варьироваться в зависимости от опыта тестировщика.

Наиболее часто используемым методом, основанным на опыте, является  предположение об ошибках. Зачастую тестировщики ожидают дефекты, исходя из своего опыта. Организованным подходом к предположению об ошибках является создание списка возможных дефектов и разработка тестов для атаки этих дефектов. Данный подход называется атакой, или атакой на недочеты. Списки дефектов и отказов могут быть созданы на основе опыта, доступной информации о дефектах и отказах и общего представления о том, почему программное обеспечение может отказать.

[Исследовательское тестирование](https://dist.belstu.by/mod/assign/view.php?id=57436) - это параллельная разработка тестов, их выполнение, протоколирование тестирования и изучение, основанные на концепции тестирования, включающей в себя цели тестирования, и проводимые в определенных временных рамках. Данный подход наиболее полезен при наличии неполных или неактуальных спецификаций и жестких временных ограничений, или при усилении или дополнении более формального тестирования. Он может служить проверкой процесса тестирования для уверенности в том, что наиболее важные дефекты обнаружены.

## Лекция №12 Тестовые случаи (Test Cases)

### **Тестовый случай**

Формально описанный алгоритм тестирования программы, специально созданный для определения возникновения в программе определённой ситуации, определённых выходных данных.

**Цель написания тестовых случаев:**

* Подготовиться к тестированию
* Детально описать шаги тестирования и ожидаемый результат;
* Задокументировать требования;
* Облегчить передачу знаний по проекту;
* Подготовиться к автоматизации тестирования.

#### **Атрибуты тестового случая**

**Идентификатор (ID)** - уникальный идентификатор. Должен отличаться от ID любой другой тестовой или проектной документации.

**Краткое название тестового случая (Summary, Title)** - обязательное поле, название тестового случая, которое формируется по следующим правилам:

* должно быть уникальным
* должно быть кратким
* не должно содержать глаголов

Например, "Успешный вход в систему", "Регистрация пользователя с невалидными данными".

**Цель тестового случая (Goal, Aim, Description)** - обязательное поле, цель тест кейса, которая описывает "что" должно происходить в системе и "при каких условиях". Например, "Проверить, что при введении существующего в системе логина и пароля, и нажатия на кнопку "Войти", пользователь входит в систему".

**Предусловия (Precondition)**- обязательно поле, в котором нумерованным списком перечислены шаги, которые необходимо выполнить тестировщику, чтобы добраться до проверяемого компонента. Описание шагов не должно содержать ожидаемый результат.

**Шаги (Steps)** - обязательное поле, в котором подробно и однозначно описано, какие шаги тестировщику нужно выполнить, чтобы проверить цель тестового случая.

**Ожидаемый результат (Expected result)** - обязательное поле, в котором подробно и однозначно описано, как поведет себя система в ответ на действия тестировщика.

**Постусловия (Postcondition)** - это не обязательное поле, которое описывает шаги, которые тестировщику необходимо выполнить, чтобы привести систему в изначальное состояние.

**Статус (Status)** - обязательно поле, в котором фигурирует одно из нижеперечисленных значений:

* + не пройден (not run) - выставляется, когда тестовый случай только создан, и еще ни разу не был пройден;
  + успешно пройден (passed) - выставляется, когда актуальный результат совпадает с ожидаемым после действий тестировщика;
  + неуспешно пройден (failed) - выставляется, когда актуальный результат НЕ совпадает с ожидаемым после действий тестировщика. Сопровождается ID отчета об ошибке;
  + заблокирован (blocked) - выставляется, когда тестировщик не смог выполнить шаги или предусловия тестового случая.

**Уровень (Level)** - это не обязательное поле, в котором выставляется уровень тестирования, которому соответствует тестовый случай (может быть компонентный уровень, интеграционный, системный или приемочный).

**Приоритет (Priority)** - это не обязательное поле, в котором может быть три значения: высокий, средний, низкий. Приоритет указывает последовательность выполнения тестовых случаев. Тестовые случаи с высоким приоритетом выполняются в первую очередь.

**Автор (Author)**- поле, которое автоматический выставляется системой управления тестовыми случаями.

**Комментарии (Notes, Comments)**- необязательное поле, которое может содержать любую дополнительную информацию, помогающую пройти тестовый случай или, которую нужно учитывать при прохождении тестового случая.

#### **Правила написания тестовых случаев**

* Один тестовый случай должен проверять только одно требование. Допустимо объединять одно функциональное требование со связанными с ним требованиями к внешнему виду (например, когда речь идет о внешнем виде сообщения об ошибке). Но проверять два функциональных требованиям в одном тестовом случае не допустимо.
* Количество шагов не должно превышать 5-7. Если шагов больше, то возможно в тестовом случае проверяется несколько требований и его нужно разделить;
* В Steps / Expected Results должно быть описано только то, что относится к цели тест кейса;
* Шаги, которые не относятся к цели тест кейса – это либо Preconditions, либо Post-Conditions;
* Желательно располагать тестовые случаи таким образом, чтобы тестировщик "двигался" последовательно по приложению, максимально уменьшая количество повторяющихся действий;
* Для повышения читаемости и поддержки тестовые случаи должны быть разделены на смысловые части (как правило, по компонентам);
* В тестовых случаях необходимо избегать ссылок на сторонние документы, так как это уменьшает удобство работы с проверками и приложением одновременно.

### **Инструменты для управления тестированием**

**Критерии выбора лучших инструментов управления тестированием**

Выбор инструмента начинается со сравнения функциональных характеристик, цен и отзывов о продуктах. Как правило, сравнение инструмента осуществляется по следующим критериям:

* Система управления тестированием – создание и управление проектами и членами команды.
* Возможность планирования тестирования – создание [тест план](https://dist.belstu.by/mod/assign/view.php?id=57446)ов, тестовых случаев, тестовых сценариев или пользовательских историй.
* Наличие выполнения тестирования -создание тестовых прогонов, управление членами процесса тестирования, удобство использования и эффективность.
* Создание протоколов испытаний по качеству и количеству работ.
* Инструмент управления задачами – доступная и быстрая система управления отчетами об ошибках.
* Использование дополнительных функций – импорт / экспорт, интеграция с другими системами, анализ сбоев и т. д.

#### Инструменты

**Система TestRail**, разработанная компанией Gurock Software GmbH, не является лучшей в каждом из критериев, но каждая вышеупомянутая функция доступна в использовании. Помимо создания тестовых случаев в ней возможно:

* Выполнять тестирование на основе созданных тестовых сценариев
* Создавать дефекты.
* Загружать их в встроенные системы отслеживания ошибок такие как JIRA, GitHub, YouTrac и другие.
* Создавать проекты.
* Делать понятные и информативные отчеты.
* Настроить систему под свои нужды с помощью открытого TestRail API.

**TestLink**- это один из проектов с открытым исходным кодом в списке представленных инструментов управления тестированием. Он имеет простой интерфейс и «технический» дизайн без излишеств. Несмотря на некоторые трудности с установкой, он используется многими командами разработчиков и  QA инженерами. Тестирование жизненного цикла начинается со создания проекта, добавления членов команды и назначения им ролей. TestLink не имеет своего собственной системы управления отчетами об ошибках, но успешно интегрируется с другими аналогичными системами для отслеживания проблем.

**PractiTest**- это бесплатный облачный сервис по управлению тестированием. Пользователь может:

* Создать тестовую библиотеку.
* Описывать различные типы тестовых случаев.
* писать сценарии тестирования.
* поддерживать пользовательские истории с помощью этой службы.
* Описывать и загружать требования к разрабатываемому продукту.
* Создать тестовый набор, назначить его тестеру и назначить точные даты тестирования.
* Сообщать о дефектах, улучшениях или задачах.

PractiTest интегрируется с JIRA, Pivotal tracker, Redmine и другими. Если на проекте имеются атомизированные тесты, работающие с Selenuim или Jenkins, этот инструмент управления тестированием также может работать с ними, используя API.

Лекция №13 Тестовые сценарии

### **Введение**

На этапе контроля качества реализованной функциональности используется тестовая документация, в которой записаны стандартные и альтернативные сценарии работы с приложением, используемые при тестировании очередной версии приложения.

Тестовая документация состоит из тестовых сценариев, то есть специальным образом разработанного описания последовательности действий в системе и ожидаемого поведения. Тестовые сценарии используются для проведения различных видов ручного тестирования:

* функционального тестирования;
* приемочного тестирования;
* нагрузочного или стресс-тестирования;
* исследовательского тестирования;
* smoke-тестирования и др.

Для разработки тестовых сценариев и выполнения тестов используются системы управления тестированием, существенно повышающие производительность тест-дизайнеров и тестировщиков, а также обеспечивающие видимость уровня качества приложений среди всех участников проекта.

Тестовые сценарии неразрывно связаны с требованиями, изменения в которых должны своевременно отражаться в тестовой документации, что позволяет сделать система управления жизненным циклом разработки приложений, при помощи механизма трассировок.

При выполнении теста тестировщик отмечает результат прохождения одного шага или всего тестового сценария, прикрепляет обнаруженные ошибки и другую вспомогательную информацию: скриншоты, дампы, логи и т.п.

### **Атрибуты тестового сценария**

* Идентификатор (ID);
* Краткое название (Summary, Title);
* Описание сценария (Description);
* Участники (Primary Actors);
* Предусловия (Precondition);
* Основной сценарий (Basic Flow);
* Альтернативный сценарий (Alternative Flow);
* Исключения (Exceptional Flow);
* Статус (Status);
* Приоритет (Priority).

Следует отметить, что в названию тестового сценария не предъявляются такие строгие требования, как к названию тестового случая.  
  
Участников тестового сценария могут быть несколько (если сценарий пишется для тестирования, основанного на ролях) и один (если сценарий пишется для en-to-end тестирования).

#### **Основной сценарий**

Последовательность шагов или действий, направленных для достижения цели. Это прямой путь, при котором используются значения по умолчанию и выбираются действия, для которых система непосредственно предназначена.

**Пример:**

1. Пользователь просматривает товар
2. Пользователь добавляет товар в корзину
3. Пользователь выбирает оплату картой
4. Пользователь вводит данные карты
5. Пользователь подтверждает оплату

#### **Альтернативный сценарий**

Последовательность шагов или действий, ведущая к достижению цели пользователя, но которая подразумевает отклонение от направления по умолчанию.

**Пример:**

*Основной сценарий:*

1. Пользователь просматривает товар
2. Пользователь добавляет товар в корзину
3. Пользователь выбирает оплату картой
4. Пользователь вводит данные карты
5. Пользователь подтверждает оплату

*Альтернативный сценарий:*

       3.а. Пользователь выбирает оплату Web Money

#### **Исключительный сценарий**

Последовательность шагов или действий, которая препятствует достижению цели и которая подразумевает отклонение от правильной работы приложения.

**Пример:**

*Основной сценарий:*

1. Пользователь просматривает товар
2. Пользователь добавляет товар в корзину
3. Пользователь выбирает оплату картой
4. Пользователь вводит данные карты
5. Пользователь подтверждает оплату

*Исключительный сценарий:*

      4. а. Пользователь вводит данные просроченной карты