Лекция 5. Отчеты о дефектах

Баг (дефект) – расхождение ожидаемого и фактического результата.

Ожидаемый результат – поведение системы, описанное в требованиях.

Фактический результат – поведение системы, наблюдаемое в процессе тестирования.

Цели написания отчёта о дефекте

- Предоставить информацию о проблеме – уведомить проектную команду и иных заинтересованных лиц о наличии проблемы, описать суть проблемы.
- Приоритизировать проблему – определить степень опасности проблемы для проекта и желаемые сроки её устранения.
- Содействовать устранению проблемы – качественный отчёт о дефекте не только предоставляет все необходимые подробности для понимания сути случившегося, но также может содержать анализ причин возникновения проблемы и рекомендации по исправлению ситуации.

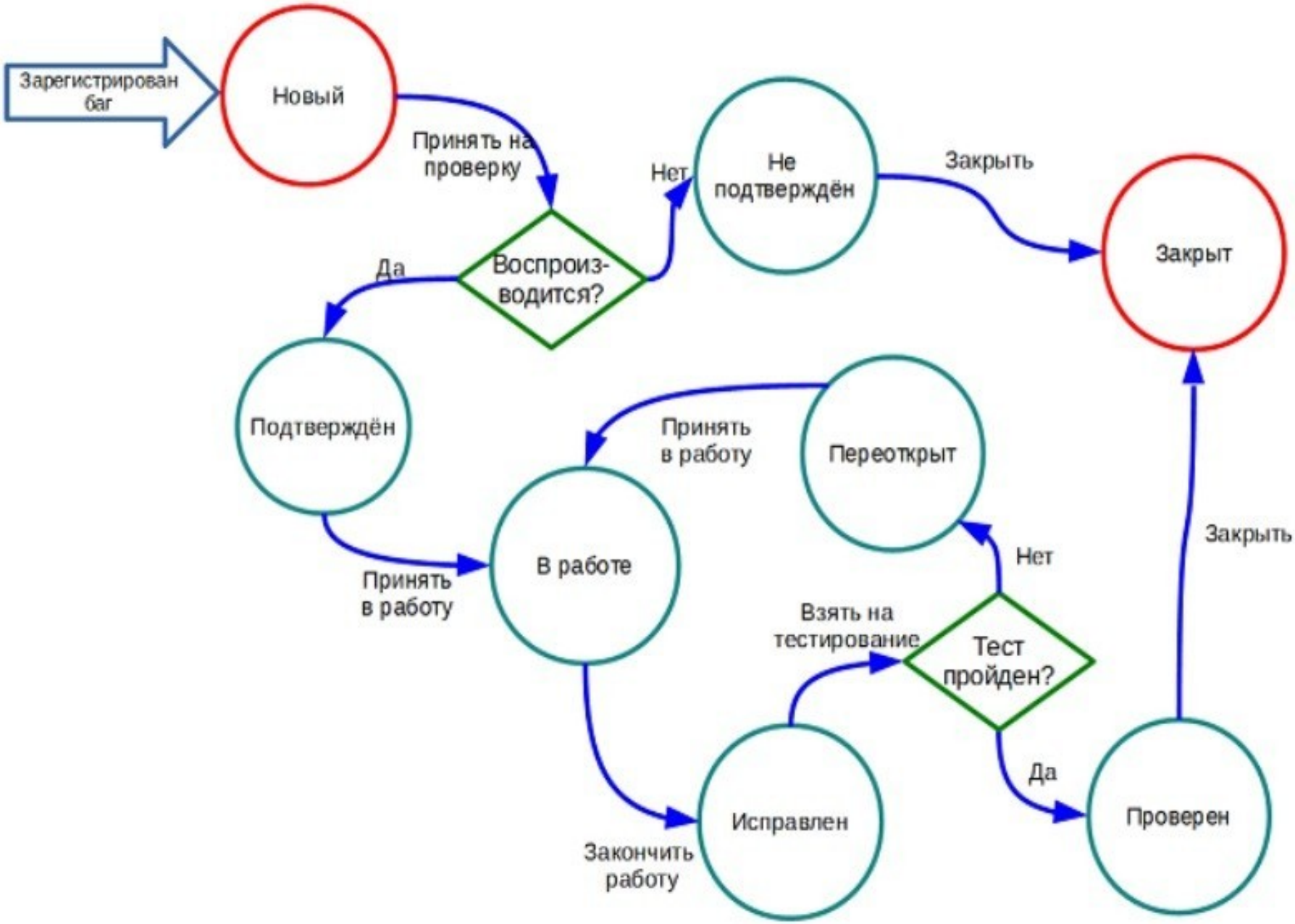
Жизненный цикл “бага”

- **Обнаружен** – начальное состояние отчёта (иногда называется «Новый»), в котором он находится сразу после создания.
- **Назначен** – в это состояние отчёт переходит с момента, когда кто-то из проектной команды назначается ответственным за исправление дефекта.
- **Исправлен** – в это состояние отчёт переводит ответственный за исправление дефекта член команды после выполнения соответствующих действий по исправлению.

- **Проверен** – в это состояние отчёт переводит тестировщик, удостоверившись, что дефект на самом деле был устранён.
- **Закрыт** – состояние отчёта, означающее, что по данному дефекту не планируется никаких дальнейших действий.
- **Открыт заново** – в это состояние (как правило, из состояния «Исправлен») отчёт переводит тестировщик, удостоверившись, что дефект по- прежнему воспроизводится.

- **Рекомендован к отклонению** – в это состояние отчёт о дефекте может быть переведён из множества других состояний, чтобы вынести на рассмотрение вопрос об отклонении отчёта по той или иной причине. Если рекомендация является обоснованной, то отчёт переводится в состояние «Отклонён».
- **Отклонён** – в это состояние отчёт переводится в случаях, подробно описанных в пункте «Закрыт»: если средство управления отчётами о дефектах предполагает использование этого состояния вместо состояния «Закрыт» для тех или иных резолюций по отчёту.

- **Отложен** – в это состояние отчёт переводится в случае, если исправление дефекта в ближайшее время является нерациональным или не представляется возможным, однако есть основания полагать, что в обозримом будущем ситуация исправится (выйдет новая версия библиотеки, вернётся из отпуска специалист по необходимой технологии, изменятся требования заказчика и т.д.)


Зарегистрирован баг
Новый
Принять на проверку
Воспроиз-водится?
Да
Нет
Не подтверждён
Закрыть
Закрыт
Подтверждён
Принять в работу
В работе
Принять в работу
Переоткрыт
Нет
Взять на тестирование
Тест пройден?
Да
Закрыть
Проверен
Исправлен
Закончить работу

Атрибуты отчёта о дефекте

- **Идентификатор** представляет собой уникальное значение, позволяющее однозначно отличить один отчёт о дефекте от другого и используемое во всевозможных ссылках.
- **Краткое описание** должно в предельно лаконичной форме давать исчерпывающий ответ на вопросы «Что произошло?» ,«Где это произошло?», «При каких условиях это произошло?». (*Например*: «Отсутствует логотип на странице приветствия, если пользователь является администратором». Что произошло? Отсутствует логотип. Где это произошло? На странице приветствия. При каких условиях это произошло? Если пользователь является администратором.)

- **Подробное описание** представляет в развёрнутом виде необходимую информацию о дефекте, а также (обязательно!) описание фактического результата, ожидаемого результата и ссылку на требование (если это возможно).
- **Шаги по воспроизведению** описывают действия, которые необходимо выполнить для воспроизведения дефекта (действия прописываются максимально подробно, с указанием конкретных вводимых значений и самых мелких деталей).
- **Воспроизводимость** показывает, при каждом ли прохождении по шагам воспроизведения дефекта удаётся вызвать его проявление. Это поле принимает всего два значения: всегда или иногда.

- **Важность** показывает степень ущерба, который наносится проекту существованием дефекта. В общем случае выделяют следующие градации важности: критическая, высокая, средняя, низкая.
- **Срочность** показывает, как быстро дефект должен быть устранён. В общем случае выделяют следующие градации срочности: наивысшая, высокая, обычная, низкая.
- **Фактический результат** – результат, полученный после прохождения шагов к воспроизведению.
- **Ожидаемый результат** – описывает ожидаемое поведение ПО после прохождения шагов к воспроизведению.

- **Симптом** – позволяет классифицировать дефекты по их типичному проявлению (косметический дефект, повреждение/потеря данных и т.д.)
- **Комментарий** – может содержать любые полезные для понимания и исправления дефекта данные.
- **Приложения** – представляет собой не столько поле, сколько список прикреплённых к отчёту о дефекте приложений (копий экрана, вызывающих сбой файлов и т.д.)

Свойства качественных отчётов о дефектах

Отчёт о дефекте считается некачественным, если в нём нарушено одно из следующих свойств:

- Тщательное заполнение всех полей точной и корректной информацией.
- Правильный технический язык.
- Специфичность описания шагов.
- Отсутствие лишних действий и/или их длинных описаний.
- Отсутствие дубликатов.
- Очевидность и понятность.
- Соответствие принятым шаблонам оформления и традициям.
- Отдельные отчёты для каждого нового дефекта.

Логика создания эффективных отчётов о дефектах

- Обнаружить дефект.
- Понять суть проблемы.
- Воспроизвести дефект.
- Проверить наличие описания найденного дефекта в системе управления дефектами.
- Сформулировать суть проблемы в виде «что сделали, что получили, что ожидали получить».
- Заполнить поля отчёта, начиная с подробного описания.
- После заполнения всех полей внимательно перечитать отчёт, исправить неточности и добавить подробности.
- Ещё раз перечитать отчёт, т.к. в пункте 6 вы точно что-то упустили.

Типичные ошибки при написании отчётов о дефектах

Ошибки оформления и формулировок:

- Плохие краткие описания
- Идентичные краткие и подробные описания
- Отсутствие в подробном описании явного указания фактического результата, ожидаемого результата и ссылки на требование
- Игнорирование кавычек, приводящее к искажению смысла
- Общие проблемы с формулировками фраз на русском и английском языках
- Лишние пункты в шагах воспроизведения
- Копии экрана в виде «копий всего экрана целиком»
- Копии экрана, на которых не отмечена проблема
- Откладывание написания отчёта «на потом»
- Пунктуационные, орфографические, синтаксические и им подобные ошибки

Логические ошибки:

- Выдуманные дефекты
- Отнесение расширенных возможностей приложения к дефектам
- Неверно указанные симптомы
- Чрезмерно заниженные (или завышенные) важность и срочность
- Концентрация на мелочах в ущерб главному
- Техническая безграмотность
- Указание в шагах воспроизведения неважной для воспроизведения ошибки информации
- Отсутствие в шагах воспроизведения информации, важной для воспроизведения дефекта
- Игнорирование «последовательных дефектов»

Шаблон отчета о дефекте

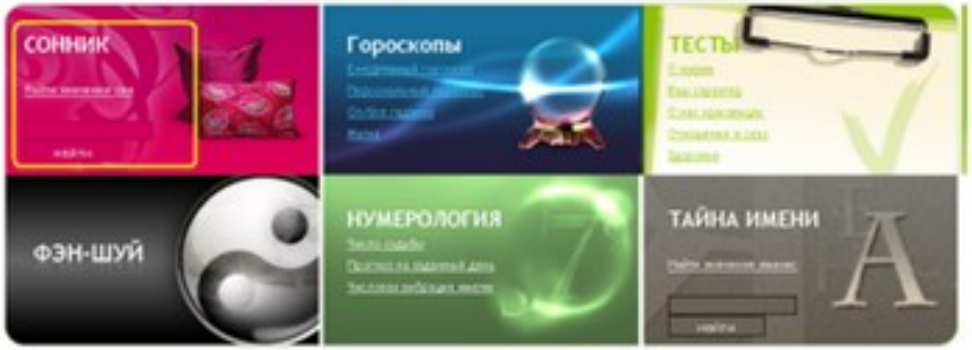
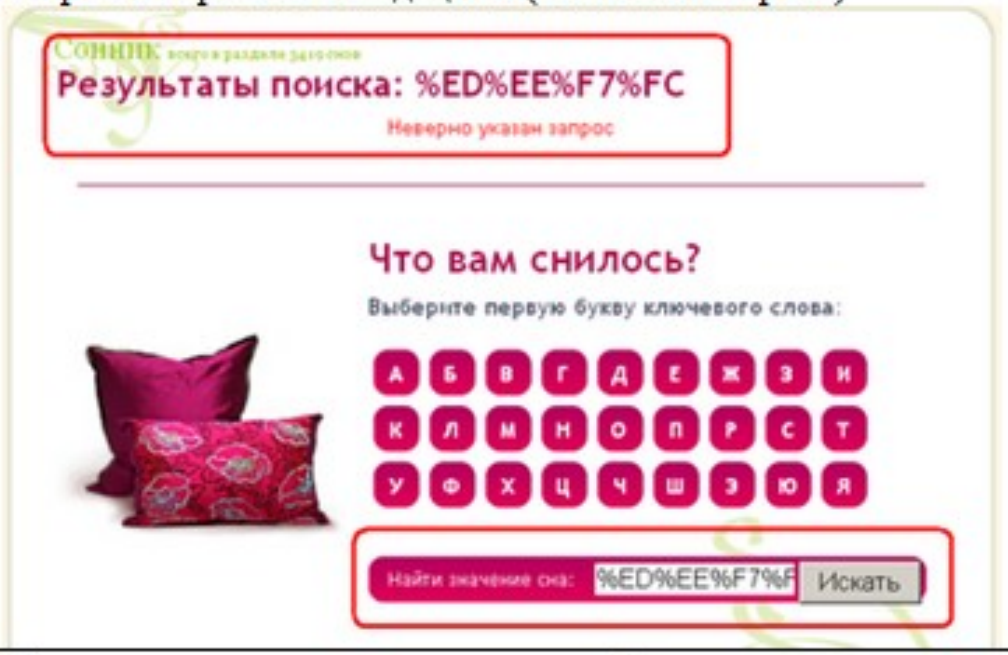
- **Автор:** *Ф.И.О. тестировщика*
- **Заголовок:** *Содержание проблемы: ясное и лаконичное описание сути проблемы (10-12 слов)*
- **Важность:** *<Критическая/ Серьезная/ Низкая/ Пожелание>*
- **Дата:** *Дата, когда была обнаружена проблема*
- **Подсистема:** *Где была обнаружена проблема (подсистема, модуль, руководство пользователя и т.п.)*
- **Описание:** *Детальное описание проблемы (несколько предложений, раскрывающих суть проблемы, указанную в заголовке). Указать параметры и конфигурацию системы, на которой была обнаружена проблема, режим работы или состояние приложения.*

- **Шаги воспроизведения:** *Подробное описание шагов, приведших к возникновению ошибки. Шаги должны быть пронумерованы. Каждый шаг должен представлять собой единое действие с четко определёнными началом и концом. Необходимо включить описание реакции системы, в т.ч. уведомления и сообщения об ошибках.*
- **Ожидаемый результат:** *То, что должно было произойти в результате выполнения последовательности шагов, описанной в разделе "Шаги воспроизведения".*
- **Текущий (фактический) результат**: *Результат последнего шага последовательности, описанной в разделе "Шаги воспроизведения". Следует описать все возможные результаты. Если проблема влечет за собой возникновение других проблем, дополнительные шаги и результаты их выполнения также должны быть описаны в этом разделе.*

Пример отчета

Баг Репорт |

Короткое описание	Поиск в соннике на главной странице, с использованием русских слов, работает не правильно.
Проект	http://www.ameno.ru/
Компонент приложения	Поиск в соннике
Номер версии	0.001
Важность: • S1 Блокирующая (Blocker) • S2 Критическая (Critical) • S3 Значительная (Major) • S4 Незначительная (Minor) • S5 Тривиальная (Trivial)	**S3 Значительная (Major)**
Приоритет: • P1 Высокий (High) • P2 Средний (Medium) • P3 Низкий (Low)	заполняется менеджером
Статус	Новая
Автор	Алексей Булат
Назначен на	имя разработчика
Ш	1. О

Шаги воспроизведения	1. Открыть главную страницу сайта: http://www.ameno.ru/ --> Внизу страницы найти раздел: СОННИК (см. копию экрана - выделено желтой рамкой)  3. Ввести поисковое слово, например "ночь"
Фактический Результат	Запрос не прошел валидацию. (см. копию экрана) 
Ожидаемый результат	Поиск прошел удачно, описание требуемого сна показано верно.

Инструменты управления отчётами о дефектах

Наиболее популярные проекты:

Тестовая документация

Тестовая документация включает в себя:

- тест план;
- тестовая стратегия;
- чек-лист;
- тестовый сценарий;
- тестовый комплект;
- пользовательская история;
- отчет о дефекте.

Тест план

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

Хороший тест план должен описывать следующее:

- Что надо тестировать? Описание объекта тестирования: системы, приложения, оборудования.
- Что будете тестировать? Список функций и описание тестируемой системы и её компонент в отдельности.
- Как будете тестировать? Стратегия тестирования: виды тестирования и их применение по отношению к объекту тестирования.
- Когда будете тестировать? Последовательность проведения работ: подготовка, тестирование, анализ результатов в разрезе запланированных фаз разработки.

Тестовая стратегия

Определяет то, как тестируем продукт. Это набор мыслей и идей, которые направляют процесс тестирования.

В стратегии тестирования описывают:

- Тестовую среду.
- Анализ рисков проекта.
- Инструменты, которые будут использовать, чтобы провести автоматизированное тестирование и для других целей.
- План действий при непредвиденных обстоятельствах.

Стратегия может быть представлена как в виде традиционно расписанного документа, так и в более наглядном формате, например, используя таблицу.

В общем, план тестирования устанавливает цели процесса тестирования, он определяет, что будет проверяться, а стратегия тестирования описывает, как достичь целей, поставленных в плане тестирования.

Чек-листы

Чек-лист – набор идей по тестированию, разработке, планированию и управлению. А также, перечень формализованных тестовых случаев в удобном для проведения проверок виде. Тестовые случаи в чек-листе не должны быть зависимыми друг от друга. Обязательно должен содержать в себе следующую информацию:

- идея проверок;
- набор входных данных;
- ожидаемые результаты;
- булевая отметка о прохождении/непрохождении тестового случая;
- булевая отметка о совпадении/несовпадении фактического и ожидаемого результата по каждой проверке.

Цель – обеспечить стабильность покрытия требований проверками необходимыми и достаточными для заключения о соответствии им продукта. Особенностью является то, что чек-листы компонуются теми тестовыми случаями, которые показательны для определенного требования.

Чек-лист, чаще всего, представляет собой обычный список, который может быть:

- списком, в котором последовательность пунктов не имеет значения (например, список значений некоего поля);
- списком, в котором последовательность пунктов важна (например, шаги в краткой инструкции).
- структурированным (многоуровневым) списком (вне зависимости от учёта последовательности пунктов), что позволяет отразить иерархию идей.

Чек-лист должен обладать рядом важных свойств:

- **Логичность.** Чек-лист пишется не «просто так», а на основе целей и для того, чтобы помочь в достижении этих целей.
- **Последовательность и структурированность.** Структурированность достигается за счёт оформления чек-листа в виде многоуровневого списка.
- **Полнота и неизбыточность.** Чек-лист должен представлять собой аккуратную «сухую выжимку» идей, в которых нет дублирования (которая часто появляется из-за разных формулировок одной и той же идеи) и в то же время ничто важное не упущено.

Правила составления чек-листов:

- Одна операция.
- Пункты чек-листа – это минимальные полные операции.
- Пункты пишутся в утвердительной форме. Цель чек-листа – проверка готовности задачи, поэтому лучше составлять пункты в утвердительной форме от третьего лица.
- Оптимальное количество пунктов – до 20. Чек-листы не должны быть длинными. Если все же это требуется, то лучше разбить задачу на несколько этапов и составить к каждому этапу отдельный чек-лист.

Преимущества использования чек-листов:

- Структурирование информации у сотрудника. При записи необходимых действий у сотрудника чётко вырисовывается нужная последовательность задач.
- Повышение скорости обучения новых сотрудников. Не нужно повторять несколько раз последовательность операций. Достаточно провести короткий инструктаж и дать чек-лист для самостоятельной работы.
- Высокий результат, уменьшение количества ошибок. Чек-листы помогают избежать проколов и ошибок по невнимательности.
- Взаимозаменяемость сотрудников.
- Экономия рабочего времени. Сотрудники будут значительно меньше времени тратить на переделывание задач.

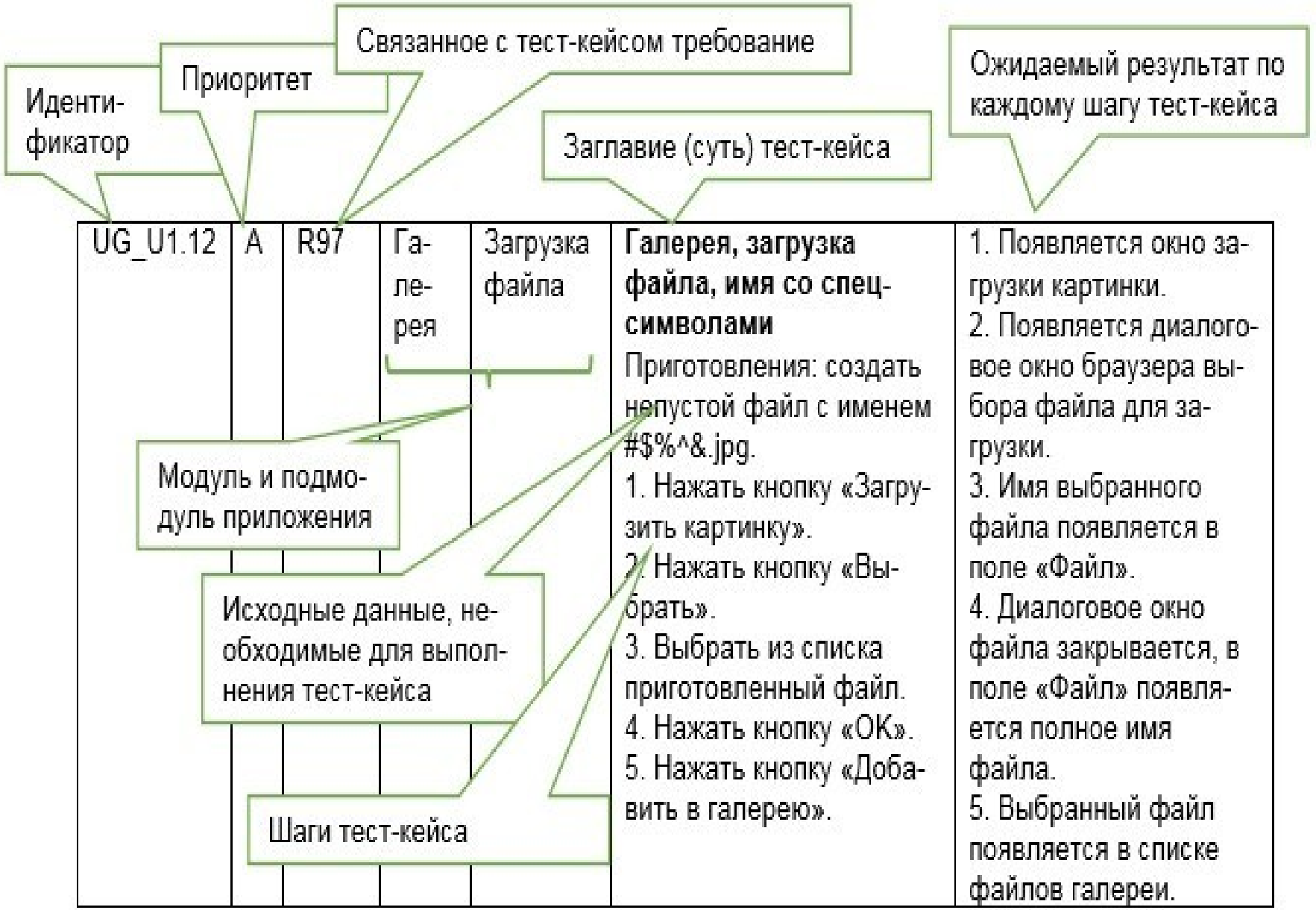
Тест кейсы

- **Высокоуровневый тест-кейс** – тест-кейс без конкретных входных данных и ожидаемых результатов. Как правило, ограничивается общими идеями и операциями, схож по своей сути с подробно описанным пунктом чек-листа. Достаточно часто встречается в интеграционном тестировании и системном тестировании, а также на уровне дымового тестирования.
- **Низкоуровневый тест-кейс** – тест-кейс с конкретными входными данными и ожидаемыми результатами. Представляет собой полностью готовый к выполнению тест-кейс и является наиболее классическим видом тест-кейсов.

Спецификация тест-кейса – документ, описывающий набор тест-кейсов (включая их цели, входные данные, условия и шаги выполнения, ожидаемые результаты) для тестируемого элемента.

Тест-сценарий – документ, описывающий последовательность действий по выполнению теста (также известен как «тест-скрипт»).

Структура тест кейса



UG_U1.12	A	R97	Галерея	Загрузка файла	**Галерея, загрузка файла, имя со спец-символами** Приготовления: создать непустой файл с именем #$%^&.jpg. 1. Нажать кнопку «Загрузить картинку». 2. Нажать кнопку «Выбрать». 3. Выбрать из списка приготовленный файл. 4. Нажать кнопку «ОК». 5. Нажать кнопку «Добавить в галерею».	1. Появляется окно загрузки картинки. 2. Появляется диалоговое окно браузера выбора файла для загрузки. 3. Имя выбранного файла появляется в поле «Файл». 4. Диалоговое окно файла закрывается, в поле «Файл» появляется полное имя файла. 5. Выбранный файл появляется в списке файлов галереи.

1. **Идентификатор** представляет собой уникальное значение, позволяющее однозначно отличить один тест-кейс от другого и используемое во всевозможных ссылках.
2. **Приоритет** показывает важность тест-кейса. Он может быть выражен буквами (A, B, C, D, E), цифрами (1, 2, 3, 4, 5), словами («крайне высокий», «высокий» и т.п.) или иным удобным способом. Приоритет тест-кейса может коррелировать с:

- важностью требования, пользовательского сценария или функции, с которыми связан тест-кейс;
- потенциальной важностью дефекта, на поиск которого направлен тест-кейс;
- степенью риска, связанного с проверяемым тест-кейсом требованием, сценарием или функцией.

3. **Связанное с тест-кейсом требование** показывает то основное требование, проверке выполнения которого посвящён тест-кейс. Не обязательное поле.
4. **Модуль и подмодуль приложения** указывают на части приложения, к которым относится тест-кейс, и позволяют лучше понять его цель. Приложение логически разделяется на компоненты (модули), а те, в свою очередь, на более мелкие компоненты (подмодули).
5. **Заглавие (суть) тест-кейса** призвано упростить и ускорить понимание основной идеи (цели) тест-кейса без обращения к его остальным атрибутам.
6. **Исходные данные**, необходимые для выполнения тест-кейса, позволяют описать всё то, что должно быть подготовлено до начала выполнения тест-кейса.

7. **Шаги тест-кейса** описывают последовательность действий, которые необходимо реализовать в процессе выполнения тест-кейса.

Общие рекомендации по написанию шагов:

- Начинать с понятного и очевидного места, не писать лишних начальных шагов.
- Даже если в тест-кейсе всего один шаг, нумеровать его.
- Использовать безличную форму (например, «открыть», «ввести» и т.п.).
- Соотносить степень детализации шагов и их параметров с целью тест-кейса, его сложностью, уровнем и т.д. Степень детализации может варьироваться от общих идей до предельно чётко прописанных значений и указаний.
- Ссылаться на предыдущие шаги и их диапазоны для сокращения объёма текста (например, «повторить шаги 3–5 со значением…»).
- Писать шаги последовательно, без условных конструкций вида «если… то…».

8. **Ожидаемые результаты** по каждому шагу тест-кейса описывают реакцию приложения на действия, описанные в поле «шаги тест-кейса». Номер шага соответствует номеру результата.

Рекомендации по написанию ожидаемых результатов:

- Описывать поведение системы так, чтобы исключить субъективное толкование.
- Писать ожидаемый результат по всем шагам без исключения.
- Писать кратко, но не в ущерб информативности.
- Избегать условных конструкций вида «если… то…».

Набор тест-кейсов – совокупность тест-кейсов, выбранных с некоторой общей целью или по некоторому общему признаку.

Наборы тест-кейсов можно разделить на

- **свободные** – порядок выполнения тест-кейсов не важен
- **последовательные** – порядок выполнения тест-кейсов важен.

Преимущества свободных наборов:

- Тест-кейсы можно выполнять в любом удобном порядке, а также создавать «наборы внутри наборов».
- Если какой-то тест-кейс завершился ошибкой, это не повлияет на возможность выполнения других тест-кейсов.

Преимущества последовательных наборов:

- Каждый следующий в наборе тест-кейс, в качестве входного состояния приложения, получает результат работы предыдущего тест-кейса, что позволяет сильно сократить количество шагов в отдельных тест-кейсах.
- Длинные последовательности действий куда лучше имитируют работу реальных пользователей, чем отдельные «точечные» воздействия на приложение.

Классификация наборов тест-кейсов

По образованию тест-кейсами строгой последовательности	По изолированности тест-кейсов друг от друга	
	Изолированные	Обобщённые
Свободные	Изолированные свободные	Обобщённые свободные
Последовательные	Изолированные последовательные	Обобщённые последовательные

- **Набор изолированных свободных тест-кейсов:** действия из раздела «приготовления» нужно повторять перед каждым тест-кейсом, а сами тест-кейсы можно выполнять в любом порядке.
- **Набор обобщённых свободных тест-кейсов:** действия из раздела «приготовления» нужно выполнить один раз (а потом просто выполнять тест-кейсы), а сами тест-кейсы можно выполнять в любом порядке.
- **Набор изолированных последовательных тест-кейсов:** действия из раздела «приготовления» нужно повторять перед каждым тест-кейсом, а сами тест-кейсы нужно выполнять в строго определённом порядке.
- **Набор обобщённых последовательных тест-кейсов:** действия из раздела «приготовления» нужно выполнить один раз (а потом просто выполнять тест-кейсы), а сами тест-кейсы нужно выполнять в строго определённом порядке.

Подходы к составлению наборов тест-кейсов:

- На основе чек-листов.
- На основе разбиения приложения на модули и подмодули.
- По принципу проверки самых важных, менее важных и всех остальных функций приложения.
- По принципу группировки тест-кейсов для проверки некоего уровня требований или типа требований, группы требований или отдельного требования.
- По принципу частоты обнаружения тест-кейсами дефектов в приложении.
- По архитектурному принципу: наборы для проверки пользовательского интерфейса и всего уровня представления, для проверки уровня бизнес-логики, для проверки уровня данных.
- По области внутренней работы приложения.
- По видам тестирования.