**[Модуль 5. Баг-репорты](https://lms.skillfactory.ru/courses/course-v1:SkillFactory+INTQAP+2022/course/" \l "block-v1:SkillFactory+INTQAP+2022+type@sequential+block@f278d14d381b4097b436cf0dd8d80d87)**

**5.1. Задача и отчёт по ошибке**

**Отчёт о помехе** (bug report): См. отчёт о дефекте.  
**Отчёт о проблеме** (problem report): См. отчёт о дефекте.  
  
**Отчёт о дефекте (defect report)**: Документ, содержащий отчёт о любом недостатке в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию.

## рис****Из каких элементов состоит каждая задача****

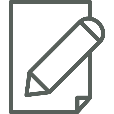
Как видите, у каждой задачи есть номер, заголовок, автор, исполнитель, которому назначена задача.

Кроме того, может быть указан дедлайн или граничный срок, до которого надо выполнить задачу. К задаче можно прикреплять документы или вставлять скриншоты.

Каждая задача проходит определенный рабочий процесс или рабочий цикл. У каждой задачи есть статусы, которые меняются по определенным правилам, называющимся ***Workflow***. Также под этим словом можно понимать порядок рассмотрения задач, баг-репортов и прочих документов в багтрекинговой системе.

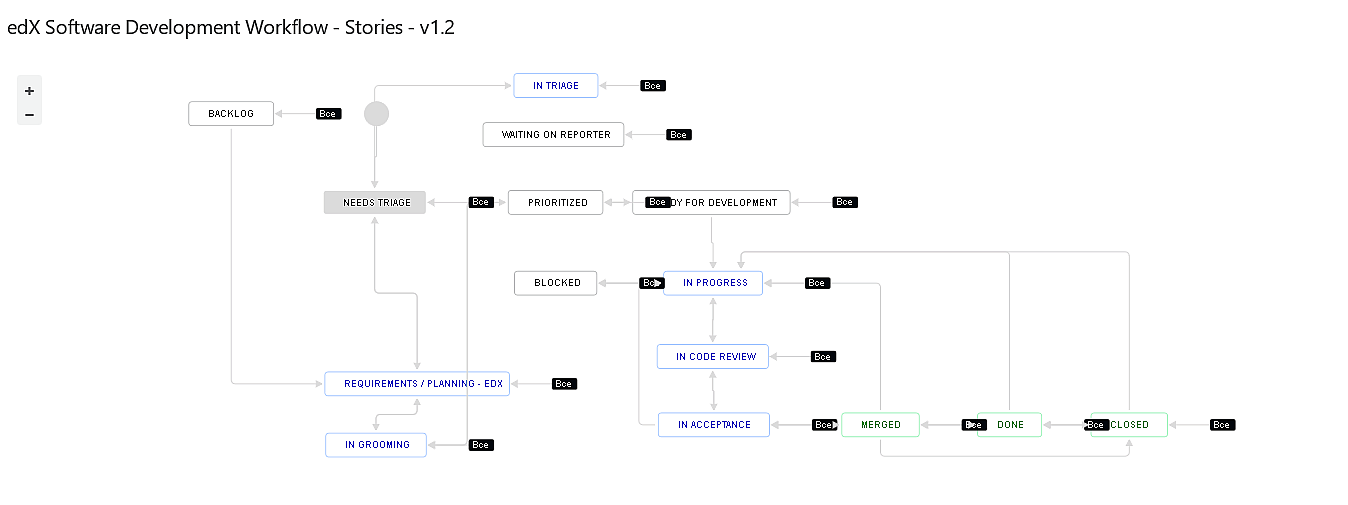
## 5.2. Жизненный цикл бага или задачи

**Жизненный цикл бага или задачи (workflow)**— это последовательность этапов, которые проходит баг (задача, связанная с нахождением бага) на своём пути с момента его создания до окончательного закрытия.

Простой жизненный цикл выглядит так:

* ***Новый***— статус присваивается автоматически после создания баг-репорта.
* ***Открыт/Взять в работу*** — баг получает данный статус после проверки актуальности руководителем команды. Этот статус может ставить и сам тестировщик.
* ***Отклонён***— присваивается после анализа нового бага руководителем команды тестирования или разработки, в случае, если описанная ошибка уже ранее была внесена в систему (дубликат) или по каким-то причинам не требуется её исправление (не слишком критична, но требует много ресурсов, например).
* ***В работе*** — программист работает над исправлением ошибки.
* ***Исправлено/Выполнено***— присваивается программистом, после того, как ошибка, по его мнению, была устранена.
* ***Повторно открыт***— при повторном возникновении ошибки этот статус ставит QA после того, как был проставлен статус «Исправлено».
* ***Закрыт*** — после исправления бага и проведения дополнительной проверки.

Часто в компаниях используют более сложные жизненные циклы. Пример жизненного цикла (рабочего процесса) в баг-трекере *openEDX* представлен на рисунке 1.



После обнаружения бага (ошибки/дефекта) тестировщик должен его зафиксировать и составить отчёт.

|  |  |
| --- | --- |
|  | **Три основных составляющих отчёта об ошибке (баг-репорта):**   1. Заголовок. 2. Шаги для воспроизведения бага (ошибки). 3. Полученный результат и ожидаемый результат. |

|  |  |
| --- | --- |
|  | 1. Предусловия — что надо сделать до того, как выполнять шаги,   воспроизводящие ошибку.   1. Постусловия — что сделать после выполнения шагов. Постусловия очень   редко указывают в реальных багрепортах, и это нормально.   1. Скриншоты и/или видео ошибки. 2. Ссылки на документацию или требования. |

Жизненный цикл будет примерно таким:

* 1. Вы заводите баг-репорт в *Jira*. Нажимаете кнопку **«Создать»** > Тип задачи (*issue type*) : Баг (*Bug*), Статус бага: «**Новый**» (автоматически проставляется).
  2. *QA-Lead* проверяет баг и переводит его на разработчика (назначает исполнителя). *QA-Lead* ставит статус бага: «**Открыт**».
  3. Разработчик, на которого назначили баг, начинает работу с ним. Он ставит статус: «**В работе**».
  4. Разработчик считает, что всё исправил, и переводит баг на того, кто составил отчет — на вас. Статус: «**Исправлено**».
  5. Вы проверяете баг и обнаруживаете, что его не исправили или исправили, но не до конца, тогда вы описываете недостатки и назначаете багу статус **«Открыт повторно**», а исполнителем ставите разработчика, который над ним работал.
  6. В ходе проверки всё работает как надо — баг получает статус **«Закрыт»**.

## 5.3. Верификация и валидация бага

## ****Верификация****

Слово «верификация» происходит от латинского слова veritas — «истина». То есть **верификация** — это установление истинности какого-либо факта.

Фактами, основными постулатами для тестировщика являются требования.

**Верифицируемое утверждение** — это такое утверждение, на которое можно ответить либо «да, истина», либо «нет, ложь» (мнемоника для запоминания: не **вери**ть, а про**вери**ть).

## ****Валидация****

Слово «валидация» происходит от латинского слова vale — «здравый», «хорошо себя чувствующий». Позднее это слово также приобрело значения «законный», «имеющий законные права» или «соответствующий законам». Это понятие более широкое, общее, чем простая проверка истинности.

Например, флорины могут быть настоящими деньгами (верифицированными), но это не значит, что они являются законным платёжным средством в любой стране мира (не валидны для платежей за пределами Флоринии).

**Валидирование** — это проверка соответствия набору правил или же проверка согласованности взаимодействия правил между собой. Выполняется применением логики и здравого смысла к работе конкретного приложения. (Мнемоника для запоминания: Здоровый **Вали** не за**валит**).

## ****Что делать, если есть вопросы к работе системы****

В работе тестировщика, особенно на начальном этапе, возможна ситуация, когда вы не можете ни верифицировать проблему (так как не знаете всех требований), ни валидировать её (так как ещё не знаете или не полностью представляете работу системы).

Не бойтесь общения! Уточняйте любые непонятные и тревожащие моменты в работе системы у более опытных тестировщиков.  Это хорошо и правильно. Вначале спрашиваете вы, после — будут спрашивать с вас.

## 5.4. Заголовок баг-репорта

**Заголовок**

Заголовок отчёта должен кратко отражать суть возникшей ошибки и давать возможность быстро понять, что же внутри.

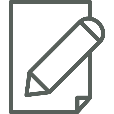
Например, вы тестируете форму входа, и обнаружили две ошибки:

1. Поле «Логин» должно принимать только *email*, однако принимает любой текст.
2. Поле «Пароль» не позволяет ввести пароль длиной в 8 символов, хотя такой пароль допустим согласно требованиям.

**По какой схеме можно составить заголовок**

Рекомендуемая схема составления заголовков — Что? Где? Когда?

* **Что** работает неправильно?
* **Где:** в каком месте программы или сайта?
* **Когда:** что надо сделать, чтобы получить баг, как часто он получается?

**Пример (1)**

**Что**: неправильный расчёт суммы заказа.

**Где**: в модуле «Корзина», страница содержимого «Корзины».

**Когда**: после ввода в поле «Количество» дробного значения.

|  |
| --- |
| **Итак, заголовок отчёта об ошибке должен:** |

1. Содержать предельно краткую, но в то же время достаточную для понимания сути проблемы информацию о баге.
2. Ответить на три вопроса: Что? Где? Когда? (Или: Как? При каких условиях?).
3. Быть достаточно коротким, чтобы быть видимым в багтрекинговых системах без обрезания и необходимости скроллинга.
4. Содержать информацию об окружении, под которым был обнаружен баг (в зависимости от типа проекта).
5. Быть законченным предложением русского или английского языка, построенным в соответствии с правилами грамматики.

|  |
| --- |
| **Как нельзя писать заголовки** |

Заголовки нельзя писать в вопросительной и повелительной форме:

* Чего это логотип на главной ведёт на сторонний сайт?
* Быстро пофиксите форму регистрации — там подтверждение пароля не принимает.

Нельзя никого обвинять и переходить на личности:

* Какой криворук прикручивал кнопку «Оплатить»? Она же всю вёрстку рвёт!
* Верстальщик, опять у тебя этот глюк с пагинацией в разделе «Наши двери»

## 5.5. Как описывать баг-репорты: шаги и результаты

**Шаги**

Шаги — это важная часть отчёта об ошибке. Они не должны упускать каких-то моментов (нажатие кнопки, ввод данных) и не должны содержать избыточных действий. Например, если заполнение поля «АБВ» никак не влияет на получение ошибки, не нужно указывать в шагах, что его надо заполнить.

Начинающим тестировщикам особенно важно отслеживать внимательно свои действия и записывать все, которые приводят к ошибке. Пока у вас нет опыта, обязательно выполняйте самостоятельно записанные шаги по несколько раз.

**Самопроверка**

Несколько проверок, которые вам рекомендуется проводить при составлении баг-репорта:

1. В баг-репорте содержится вся необходимая информация для полученного результата (бага) (сокращают «Получено» или «*AR*» — *actual result*).
2. В баг-репорте содержится ожидаемый результат (сокращают *ОР* или *ER* — *expected result*).
3. Каждый шаг можно понять, исходя из предусловий, предыдущих шагов и видимого на экране состояния программы.
4. Каждый шаг нельзя выполнить неверно.
5. Если я дам баг-репорт коллеге, сможет ли он воспроизвести проблему?

**Глоссарий ISTQB**

**Ожидаемый результат (*expected result*)**: Поведение компонента или системы при установленных условиях, которое определено спецификацией или другими источниками.  
  
**Фактический результат (*actual result*):** Наблюдаемое или генерируемое поведение компонента или системы во время тестирования.

### **Пример отчёта об ошибке**

**Заголовок**: При клике «Купить» на любом кольце товар не появляется в «Корзине».

**Предусловие**: Зайдите на сайт: <https://silverandsilver.ru/>.

**Шаги**:

1. Наведите мышкой в шапке сайта на раздел «Для женщин».
2. В выпадающем меню кликните на пункт «Кольца».
3. Кликните на любое из колец в открывшемся каталоге.
4. В открывшемся превью товара выберите любой доступный размер кольца.
5. Кликните на кнопку «Купить».
6. Вместо кнопки появится надпись «Товар добавлен в корзину. Перейти в корзину».
7. Кликните на фразу «Перейти в корзину».

**Полученный результат**: В корзине нет добавленного товара.

**Ожидаемый результат**: В корзине отображается кольцо, которое было выбрано.

**Ошибки при составлении баг-репортов**

**Перечень типичных ошибок в отчёте:**

* 1. Написание шага в личном виде «Я нажимаю на кнопку ОПЛАТА» вместо «Нажмите на кнопку ОПЛАТА».
  2. Написание ожидаемого результата в желательном виде, размытость формулировок.

Плохо: «Надо, чтоб все сообщения открывались, а не только прочитанные».

Хорошо: «В окне видны все сообщения (в том числе непрочитанные)».

* 1. Неточный и не отражающий суть ошибки заголовок.
  2. Пропуск шагов.
  3. Отсутствие ожидаемого результата.
  4. Смертный грех — отсутствие полученного результата.

## ****Важный момент!****

Читая источники в сети, вы можете найти достаточно много информации по разбору правил описания фактического (Actual result), он же полученный, и ожидаемого результатов (Expected result), которые являются обязательными атрибутами при оформлении баг-репорта.

Споры на тему, какой из результатов должен стоять первым, ведутся активно. Как [пишет один из авторов блога](https://training.qatestlab.com/blog/technical-articles/results-bug-report/) на QATestLab: «Чтобы дать ответ на этот вопрос, стоит подумать, с какой целью составляется отчет — для того, чтобы скорее донести до разработчиков продукта и всей команды, что что-то пошло не так».

Предлагаем не спорить, а проработать самостоятельно примеры. Составьте баг-репорты в личном аккаунте в Jira по заданиям 5.5.1. - 5.5.3., изменив их так, чтобы они отвечали всем рекомендациям, приведенным в данном уроке.

**Внимание!** Вы не можете поделиться ссылками c однокурсниками в Slack. Однако, можете желающих пригласить в свой проект. Или поделиться скриншотами, скринкастами. О том, как их сделать вы узнаете в следующем уроке.

## 5.6. Отчёт об ошибке. Дополнение

Вы обнаружили баг! **Не спешите открывать баг-трекинговую систему**, она же баг-трекер, и заводить задачу с заголовком «Не работает…».

**Начните** с того, что воспроизведите ошибку несколько раз для того, чтобы определить и зафиксировать минимальное количество шагов. Вам нужно достаточное количество шагов для воспроизведения проблемы, и ничего лишнего.

Когда вы поняли, какие именно данные и какие ваши действия приводят к проблеме, кратко сформулируйте ее суть — **сформулируйте заголовок баг-репорта**. Он должен быть не длинным, но емким, раскрывающим проблему. Используйте прием «Что? Где? Когда?».

Опытные тестировщики говорят о том, что и заголовок, и описания шагов — это руководство к действию для тех, кто будет исправлять проблему, поэтому лучше формулировать с помощью активных глаголов «перейти по ссылке», «выбрать в фильтре», «нажать на кнопку» и прочее.

Теперь **переходим в баг-трекер, создаем задачу** типа «Баг». Записываем заголовок с указанием номера. В некоторых системах есть поля «Подробное описание» и «Шаги для воспроизведения». В Jira вы заполняете поле «Описание». В нем описываете проблему более подробно, уточняя детали, отсутствующие в заголовке. Если вы не понимаете сути проблемы, лучше не заниматься предугадыванием и предположением. В этом же поле запишите шаги, которые вы определили для воспроизведения проблемы. Не забудьте вписать результаты.

## ****Предусловия****

Предусловия не обязательны, но часто используются, чтобы указать действия, которые надо выполнить до начала теста (например, запустить определенный браузер, открыть определенный раздел сайта).

## ****Постусловия****

Иногда после выполнения бага нам нужно «вернуть к жизни» программу для дальнейших тестов. В таких случаях задаются постусловия — перелогиниться, обновить страницу, перезапустить браузер или вообще операционную систему.

## ****Описание окружения, указание версии****

Если продукт выпущен для нескольких платформ, имеет ряд версий, то это надо указывать в баг-репорте. Например, операционная система (ОС) Windows 10 имеет возможность переключения версии всего языкового интерфейса в версии Home Pro, но не имеет такой опции в версии Starter. Если мы тестируем Windows 10 Home Pro и не находим опцию переключения языка интерфейса, то разумно будет не только указать на сам баг, но и упомянуть версию ОС.

Ссылка для скачивания приложения: <https://app.prntscr.com/ru/>.

Еще одна бесплатная программа, на которую стоит обратить внимание — браузерное приложение [Joxi](https://joxi.ru/" \t "_blank).

Скриншот, который вы увидите ниже, сохранен в «облаке» и может быть доступен по ссылке. В некоторых ситуациях удобнее дать ссылку на просмотр скриншота, но не вставлять само изображение.

Приучите себя размечать скриншоты: добавлять текстовые объекты, рамки, стрелки, подчеркивания, чтобы акцентировать внимание на проблеме.

**Видео**

Иногда события на экране происходят крайне динамично. В таком случае нужно использовать запись видео, чтобы показать ошибку.

**Пример**

Запись видео для демонстрации бага «Дребезжание изображения товара при определенной позиции скрола».

На скриншоте не видно, как «дребезжит» изображение, но видно благодаря разметке, что снятым видео можно поделиться или скачать его.

Задача тестировщика определить, в каком формате файл более наглядно подтвердит ошибку. Возможно, нужно прикрепить к баг-репорту и скриншот, и видео.

Бесплатных, удобных, надежных программ для видеозаписи достаточно много. Часть из них устанавливается, как правило, локально и является не только полноценными редакторами для записи и монтажа видео, но и используется для потоковой трансляции — стриминга.

В задании 5.6.5. были приведены варианты таких программ, например, *[Camstudio](https://camstudio.org/" \t "_blank)* и [OBS Studio](https://studio-obs.ru/).

В официальном обзоре о возможностях и функционале OBS studio мы читаем следующее:

«***OBS studio*** — это продвинутая модификация самой популярной утилиты для стриминга, которая расшифровывается как Open Broadcaster Software. Она хорошо знакома тем, кто пытался запустить трансляцию через Youtube, Hitbox, Twitch, а так же программа может записывать видео с экрана монитора со звуком». ([*Источник*](https://studio-obs.ru/))

Умеет работать с видео и скриншотами и очень удобна в использовании программа ***Sharex***. За счет множества модулей ее можно очень гибко настроить, например, сделать так, чтобы скриншот сразу добавлялся в существующий или новый отчет об ошибке в Jira.

## 5.7. Приоритет бага. Определение priority и severity

## ****Приоритет бага или Priority****

Самый высокий приоритет имеет блокирующий баг — Blocker, самый низкий — баг с приоритетом Trivial.

***Blocker*** означает, что выявленный баг делает невозможным целый ряд тестов. Например, нам надо тестировать форму регистрации, а поле для ввода пароля вообще не принимает никакой ввод значений. Если мы планировали зарегистрироваться и тестировать что-то на сайте после этого, такая ошибка будет блокировать ещё больше работы.

***Critical*** означает, что выявленный баг затрагивает основную функциональность, без работы которой нельзя получить ожидаемый результат для тестового случая.

***Major*** означает, что выявленный баг затрагивает функциональность так, что получить результат можно, но для этого нужны «хитрости» или дополнительные действия, либо баг может затрагивать функционал не тестируемый в данный момент.

Blocker, Critical и Major — три главных **метрики для определения критичности** багов.

С меткой Minor ставят баги типа незначительных опечаток или небольшого визуального «выбивания» цвета, верстки, элемента из стиля, а Trivial — совсем уж мелкие баги, которые и не заметить можно (типа пробела после слова и перед запятой). Как вы понимаете, баги с меткой Trivial крайне редко встречаются в системах отслеживания багов.

Помните, что от целей и задач тестируемого сайта «вес» бага может сильно отличаться. Если для сайта по продаже картин отсутствие изображений — это Critical, то для сайтов с текстом и данными (например, Торговая биржа) это может быть Major, Minor или даже Trivial.

Иногда (но очень редко) на некоторых проектах используют еще одну метрику, чтобы подчеркнуть важности или серьезность бага — *Severity.* И даже в тестовых вопросах (на собеседовании или при сертификации) встречается вопрос: «В чем разница между *Priority* и *Severity»?*

Давайте попробуем разобраться.

Итак, определения согласно [глоссарию ISTQB](https://www.rstqb.org/ru/istqb-downloads.html?file=files/content/rstqb/downloads/ISTQB%20Downloads/ISTQB_Glossary_English_v2.3.pdf) такие:

***priority:***  
The level of (business) importance assigned to an item, e.g. defect.

**приоритет:**

уровень важности, веса (в т. ч. для бизнеса), присвоенный элементу, например, дефекту.

***severity:***  
The degree of impact that a defect has on the development or operation of a component or system.

**серьезность:**

степень влияния дефекта на разработку или работу компонента или системы.

А в реальной жизни поле означающее степени влияния дефекта в любом баг-трекере называется… (бум-бу-бу-бум-бум…) ... ***priority*** или приоритет.

## 5.8. Сопровождение бага от написания и до закрытия

## ****Что делает тестировщик с момента обнаружения бага****

1. **Воспроизводит баг**

Как только баг обнаружен – надо повторить путь его получения, а лучше повторить дважды или трижды (особенно в начале вашей карьеры). Это нужно чтобы понять: ничего не упустили; ошибка в работе не была случайностью в какой-то момент работы, и вам это не показалось.

2. **Убеждается, что в системе нет дубликатов**

Затем надо сделать поиск по баг-трекеру, по нескольким ключевым словам, которыми баг мог быть описан, чтобы избегать дубликатов.

3. **Собирает информацию**

Задача тестировщика — предоставить исчерпывающую информацию об инциденте. Постарайтесь минимизировать количество шагов для его воспроизведения, удостоверьтесь, что нет ничего лишнего. Если используются какие-то входные данные, убедитесь, что и они не содержат лишней информации.

4. **Составляет баг-репорт**

Пишите заголовок в конце. Когда вы описали шаги и полученный результат (дефект), то потом кратко сформулируйте его суть, продумывая тему (описание) баг-репорта. При необходимости прикрепите ссылки, скриншоты, видео.

5. **Убеждается, что баг исправлен программистами**

После заведения баг-репорта почти всегда баг исправляют программисты и возвращают его вам же. Что с ним делать теперь?

Еще раз пройти по шагам и убедиться, что приложение возвращает ожидаемый результат. Проверить, не породило ли исправление новые баги, или, возможно, тот же баг проявляется в других условиях.

Если всё хорошо (баг проходит верификацию и валидацию) — баг закрывается.

Если есть замечания — баг возвращается на доработку.