Глава 3.4 Что такое баг-репорт и как его составить.

Во время тестирования тестировщик записывает свои наблюдения, найденные баги и другую полезную информацию, полезную разработчикам и менеджменту. Вся эта информация — баг-репорт.

**Баг-репорт** - это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

Баг – это некорректная работа ПО. Причиной является ошибка в коде, которую допустил разработчик в процессе его написания.

**Структура баг-репорта.**

Детальный баг-репорт — важный итог, полученный в результате тестирования. Он содержит следующую информацию(атрибуты):

1. Заголовок.

Краткий заголовок, описывающий проблему в общем. Не должен быть слишком длинным, но должен быть понятным и точным.

1. Описание.

Как правило, повторяет заголовок баг-репорта, так что это поле иногда пропускают.

1. Версия.

Production-версия продукта, в которой был найден баг. Можно потратить много времени, пытаясь воспроизвести ошибку на других версиях (в которых, возможно, ее просто нет)

1. Приоритет.

Приоритет дефекта:

• P1 Высокий (High)

• P2 Средний (Medium)

• P3 Низкий (Low)

1. Серьезность.

Пятиуровневая система градации серьезности дефекта:

• S1 Блокирующий (Blocker)

• S2 Критический (Critical)

• S3 Значительный (Major)

• S4 Незначительный (Minor)

• S5 Тривиальный (Trivial)

1. Статус.

В каждый момент времени баг может быть в одном из состояний: «Active» (активный), «Ready for Testing» (готов к тестированию), Closed (закрыт). Баг становится «активным», когда он найден; готов к тестированию — когда разработчик его исправляет. Тестировщик может перевести его в состояние «Closed» после исправления и повторного тестирования.

1. Окружение.

Информация об окружении, на котором был найден баг: операционная система, сервис пак, для WEB тестирования - имя и версия браузера и т.д.

1. Шаги воспроизведения.

Перечисление действий, которые необходимо выполнить, чтобы воспроизвести найденный баг.

1. Фактический результат.

Результат, полученный после прохождения шагов к воспроизведению.

1. Ожидаемый результат.

Ожидаемый правильный результат.

1. Вложения.

Файл с логами, скриншот или любой другой документ, который может помочь прояснить причину ошибки или указать на способ решения проблемы.

1. Дополнительная информация.

В этом поле могут быть указаны пояснения от тестировщика (что он еще делал для воспроизведения бага, и каковы были результаты).

Баг-репорт представляет собой технический документ, следовательно, его необходимо формировать в техническом стиле. Не должно быть никаких художественных элементов и размытых формулировок. Чтобы сделать все максимально правильно, нужно пользоваться шаблонами, которые приняты в конкретной компании. Несколько важных нюансов, которые необходимо учесть при составлении данного документа:

* **Размер заголовка**. Эта часть баг-репорта помогает разработчику быстро понять суть проблемы. Заголовок не должен быть слишком длинным или коротким.
* **Локализация**. Тестировщик должен не просто найти баг, но еще и правильно его описать. В противном случае разработчики будут решать проблему, которая не имеет никакого отношения к их обязанностям.
* **Вложения**. Если баг визуальный или UX (к примеру, нашлась некорректная вёрстка или не функционирует кнопка), то для наглядности необходимо сделать скриншоты. Так разработчик поймет, что видит пользователь при возникновении ошибки.
* **Шаги воспроизведения**. Не стоит начинать с этапа «Включить компьютер». При этом шаги должны быть описаны максимально точно, без размытых формулировок.
* **Взгляд на проблему**. Специалист должен поставить себя на место заказчика. К примеру, если текст не помещается в поле, то такая ошибка мешает бизнесу? Это необходимо, чтобы правильно оценить серьезность и приоритет дефекта.
* **Фактический и ожидаемый результаты**. Если тестировщик опишет проблему непонятно, то разработчику придется задать ему уточняющие вопросы. К примеру, фактически результат— кнопка не работает, а ожидаемый результат — кнопка работает. Такое описание не дает разработчику никакого понимания сути проблемы.

Обычно после того как баг был обнаружен, он попадает на стадию «Активный». Когда все работы по исправлению будут завершены, он получит статус «Закрыт».

Однако между этими стадиями есть еще четыре промежуточных **этапа**, к которым может быть отнесен баг:

* **Отклонен**. Эта стадия нужна для багов, которые не получилось воспроизвести из-за неточностей в «Шагах воспроизведения» или повторном попадании дефекта в категорию «Новый». Кроме того, баг может быть отклонен в том случае, если его исправление разрешается перенести на более позднюю дату.
* **Открыт**. Если баг требуется исправить в кратчайшие сроки.
* **Исправлен**. Сюда относят уже исправленные ошибки.
* **Переоткрыт**. Если ранее баг был отсрочен или отклонен, но потом решение поменялось.

**Степени серьезности и приоритетов в баг-репортах.**

**Степень серьезности в баг-репорте** (**Severity**) — это атрибут, который характеризует уровень критичности (влияния) ошибки для ПО. Иными словами, данный параметр указывает на масштабность изменений в программе, произошедшими из-за ошибки в коде.

Есть пять базовых степеней серьезности:

1. **Блокирующий(Blocker)**. Вся программа не сможет функционировать, если ошибку не исправить.
2. **Критический(Critical)**. Большинство составляющих программы работает некорректно.
3. **Значительный(Major)**. Ошибка мешает функционированию одной из основных логических цепочек ПО. Программа работает, но при этом задача не решается с помощью предназначенного для этого способа.
4. **Незначительный(Minor)**. Основные логические цепочки программы работают нормально. ПО можно использовать с незначительными потерями качества.
5. **Тривиальный(Trivial).**Вся программа будет нормально работать, даже если ошибку не исправлять.

**Приоритет (Priority)** - это атрибут, указывающий на очередность выполнения задачи или устранения дефекта. Можно сказать, что это инструмент менеджера по планированию работ. Чем выше приоритет, тем быстрее нужно исправить дефект.

1. **Высокий (High)**  
   Ошибка должна быть исправлена как можно быстрее, т.к. ее наличие является критической для проекта.
2. **Средний (Medium)**  
   Ошибка должна быть исправлена, ее наличие не является критичной, но требует обязательного решения.
3. **Низкий (Low)**  
   Ошибка должна быть исправлена, ее наличие не является критичной, и не требует срочного решения.

Порядок исправления ошибок по их приоритетам:

**High -> Medium -> Low**

**Основные ошибки при написании багов репортов.**

Тестировщик должен иметь определенные знания и навыки, чтобы правильно формировать баг-репорты.

* **Неправильно составлен заголовок.**

Он должен отвечать на вопросы «Что? Где? Когда?». В загаловке не должны быть лишние сведения (версии, окружения, учетные данные пользователей и т. п.).

* **Недостаточность предоставленных данных**  
  Не всегда одна и та же проблема проявляется при всех вводимых значениях и под любым вошедшим в систему пользователем, поэтому настоятельно рекомендуется вносить все необходимые данные в баг репорт
* **Определение серьезности**  
  Очень часто происходит либо завышение, либо занижение серьезности дефекта, что может привести к неправильной очередности при решении проблемы.
* **Язык описания**  
  Часто при описании проблемы используются неправильная терминология или сложные речевые обороты, использование «жаргона», которые могут ввести в заблуждение человека, ответственного за решение проблемы.
* **Отсутствие ожидаемого результата**  
  В случаях, если вы не указали, что же должно быть требуемым поведением системы, вы тратите время разработчика, на поиск данной информации, тем самым замедляете исправления дефекта. Вы должны указать пункт в требованиях, написанный тест кейс или же ваше личное мнение, если эта ситуация не была документирована.
* **Нет скриншота / видеозаписи для UI/UX багов.** Либо на скриншоте не выделен дефект.

Если начинающий тестировщик будет заранее знать об этих часто встречаемых ошибках, то он сможет написать более качественный баг-репорт. При составлении баг-репорта многие новички слишком сосредоточены на деталях. Нужно стараться идти от частного к целому. Это позволяет более объективно рассмотреть проблему. Необходимо понять, какое влияние баг оказывает на процессы, функциональность и удобство пользователя.