

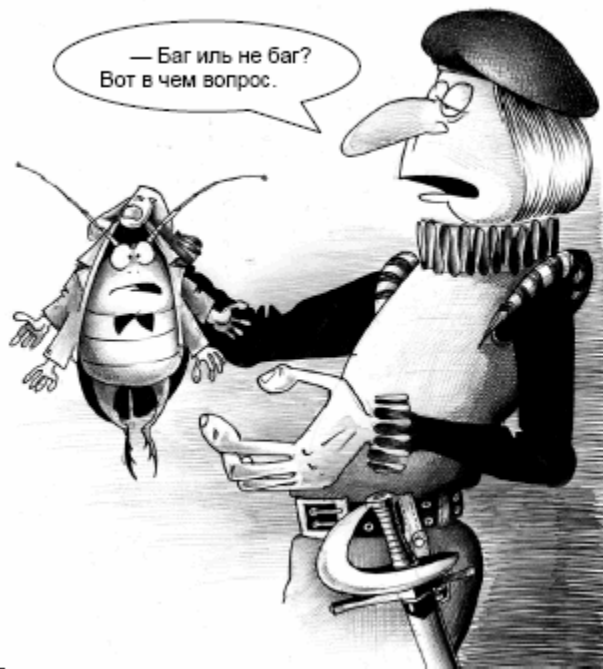
Поиск и документирование дефектов



Введение

- Тестирование — это не поиск ошибок!
- *Главное правило тестирования* – не надо стараться найти как можно больше ошибок, надо стараться пропустить как можно меньше!

Как определить: есть дефект в программе или нет?



- Программа не делает то, что она должна делать согласно требований
- Программа делает что-то, чего она не должна делать согласно требований
- Программа делает что-то, о чём в требованиях не упоминалось
- Программа не делает чего-то, о чём не говорится в требованиях, однако подразумевается, что она должна это делать
- Программа трудна для понимания и неудобна в использовании

Определения дефекта

Рассмотрим несколько определений дефекта:

«Программная ошибка – ни что иное, как изъян в разработке программного продукта, который вызывает несоответствие ожидаемых результатов выполнения программного продукта и фактически полученных результатов.» (Роберт Калбертсон, Крис Браун, Гэри Кобб «Быстрое тистирование»)

«Баг (bug) – это отклонение фактического результата (actual result) от ожидаемого результата (expected result). В соответствии с законом исключённого третьего у нас есть баг при наличии любого фактического результата, отличного от ожидаемого.» (Роман Савин «Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах»)

«В целом, разработчики различают дефекты программного обеспечения и сбои. В случае сбоя программа ведёт себя не так, как ожидает пользователь. **Дефект** – это ошибка/неточность, которая может быть (а может и не быть) следствием сбоя.» (Википедия.)

«Дефект – поведение программы, затрудняющее или делающее невозможным достижение целей пользователя или удовлетворение интересов участников. Подразумевает возможность исправления. При невозможности исправления переходит в разряд ограничения технологии.» (Сергей Мартыненко блог «255 ступеней»)

Определение дефекта

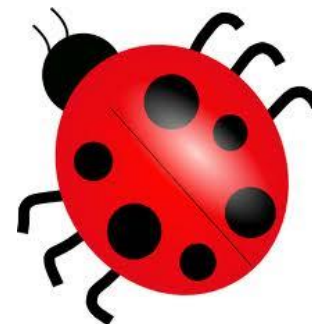
Дефект – это несоответствие требованиям или функциональным спецификациям.

- Также следует помнить, что к багам относится любое некорректное поведение программы, не соответствующее оправданным ожиданиям пользователя, даже в том случае, если это поведение не документировано в требованиях и спецификациях.
- Баги могут встречаться в любой документации, в архитектуре и дизайне, в коде программы и т.д. Иногда баг на самом деле является не ошибкой в программе, а результатом неверного конфигурирования программы и/или окружения.

Вспоминаем...

По легенде «Днём рождения» первого компьютерного бага считается 9 сентября 1945 года.

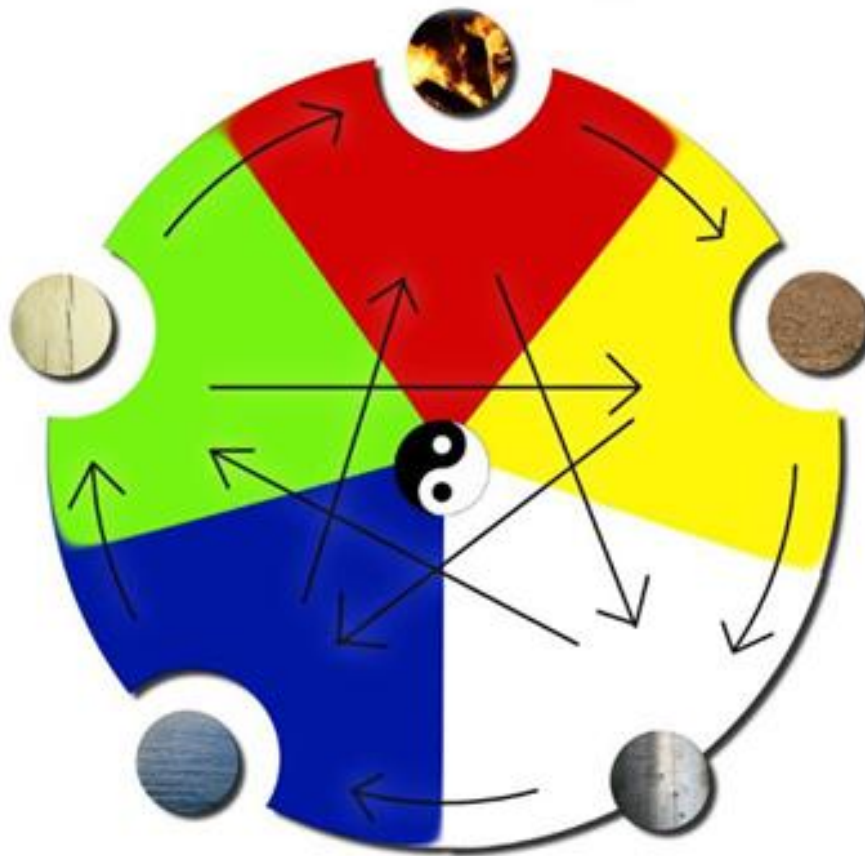
В Гарвардском университете, где работал этот компьютер, в этот день с машиной возникли проблемы, и исследование показало, что мотылёк попал между контактами панели. Операторы извлекли мотылька и сделали соответствующую запись в журнале: «Обнаружен первый настоящий баг». (Англ. «bug» – жук, насекомое).



Кто может задокументировать дефект?

- Задокументировать дефект может кто угодно, а именно обнаруживший некорректное поведение программы:
 - ✓ Тестировщики и специалисты по обеспечению качества
 - ✓ Разработчики
 - ✓ Представители службы технической поддержки
 - ✓ Продавцы и специалисты по маркетингу
 - ✓ Представители заказчика
 - ✓ Конечные пользователи

Жизненный цикл дефекта



Жизненный цикл дефекта

Новый (New). Когда тестировщик находит дефект, то он представляет его на рассмотрение в систему управления дефектами. С этого момента баг начинает свою официальную жизнь и о его существовании знают необходимые люди.

Назначен (assigned). Далее разработчик рассматривает дефект и назначает его исправление кому-то из команды разработчиков.

Исправлен (fixed). Разработчик, которому было назначено исправление дефекта, исправляет его и сообщает о том, что задание выполнено.

Проверен (verified). Тестировщик, который обнаружил ошибку проверяет на новом билде (в котором исправление данной ошибки заявлено), исправлен ли дефект на самом деле. И только в том случае, если ошибка не проявится на новом билде, тестировщик меняет статус бага на Verified.

Открыт заново (reopened). Если баг проявляется на новом билде, тестировщик снова открывает этот дефект. Баг приобретает статус Reopened.

Жизненный цикл дефекта

Отклонён (reject) – не баг (not a bug). Баг может быть отклонён. Во-первых, потому, что для заказчика какие-то ошибки перестают быть актуальными. Во-вторых, это может случиться по вине тестировщика из-за плохого знания продукта, требований (дефекта на самом деле нет).

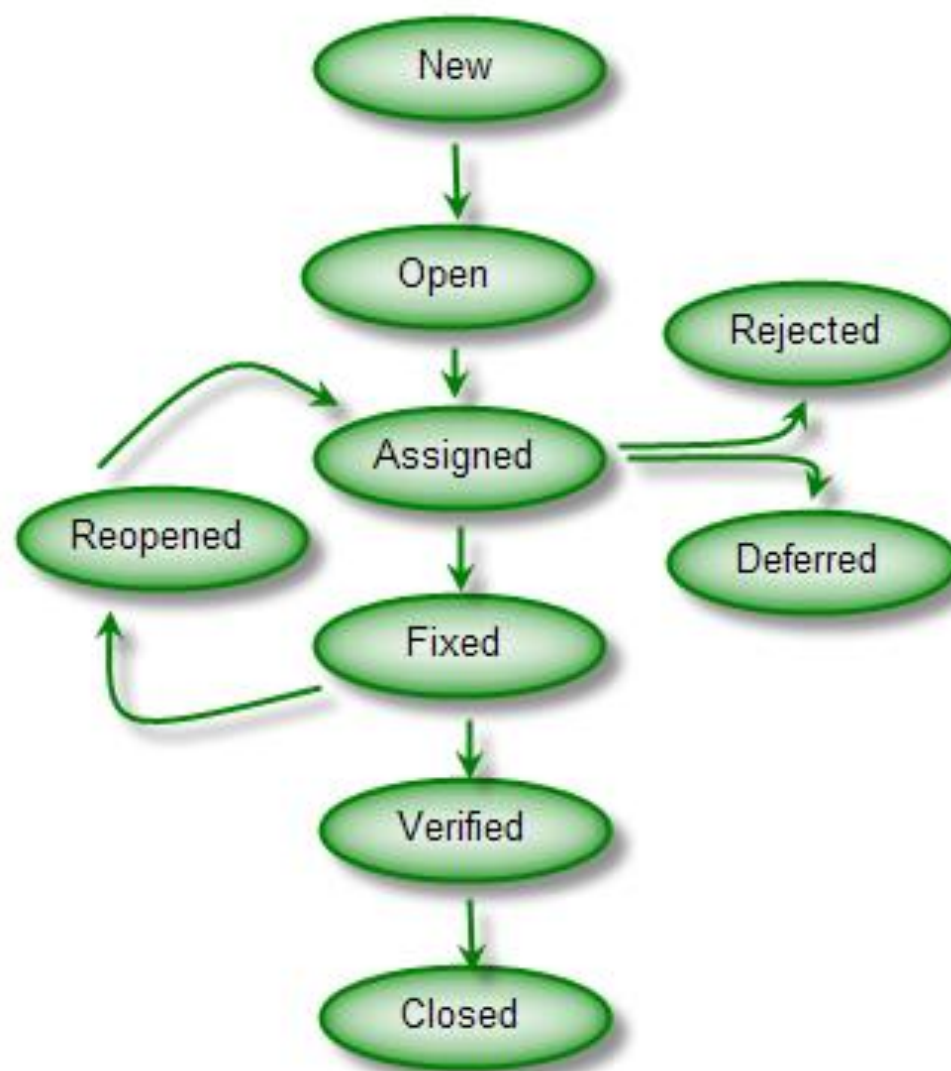
Отложен (deferred). Если исправление конкретного бага сейчас не очень важно или заказчик пока думает, или мы ждём какую-то информацию, от которой зависит исправление бага, тогда баг приобретает статус Deferred.

Дублирован (duplicate). Если уже существует открытый баг, который направлен на выявление той же самой ошибки.

Не могу воспроизвести (can not reproduce). Если в описании баг нет всей необходимой информации, программист не сможет его воспроизвести, что бы починить.

Не чинить (will not fix). Формально это баг, но чинить его не будут, не могут или не хотят.

Жизненный цикл бага



Открытые и закрытые баги

- **Закрытые (closed)** баги. Закрытым считается баг в состояниях
 - ✓ **Проверен** (*verified*)
 - ✓ **Отклонён** (*not a bug, will not fix*)
 - ✓ **Дублирован** (*duplicated*)
- **Открытые (open)** баги. Открытыми являются баги в состояниях
 - ✓ **Открыт** (*opened*),
 - ✓ **Назначен** (*assigned*),
 - ✓ **Открыт заново** (*reopened*).

Иногда к открытым относят и баги в состояниях

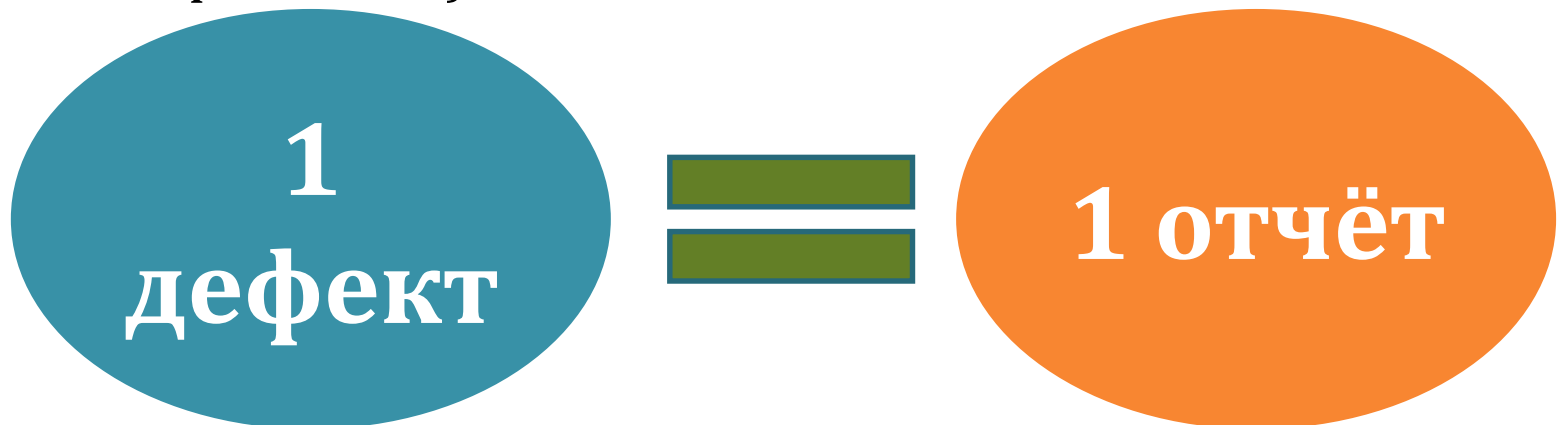
- ✓ **Исправлен** (*fixed*) и
- ✓ **Отложен** (*deferred*).

Отчеты об ошибках

Отчёт об ошибке (Баг/Дефект Репорт (Bug Report)) - это **документ (!)**, описывающий ситуацию или последовательность действий приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

Отчёт об ошибке («баг-репорт», «bug report») – один из основных результатов работы тестировщиков.

Именно этот результат работы видят коллеги (другие тестировщики и люди, не входящие в команду тестировщиков).



Цель написания отчета об ошибке

- ❑ Отчеты об ошибках пишутся с целью:
 - предоставить информацию о проблеме, ее свойствах и последствиях;
 - приоритизировать проблему по важности и скорости устранения;
 - помочь программистам обнаружить и устранить источник проблемы.
- ❑ Однако, **основная цель** написания отчёта об ошибке – это **устранение ошибки.**

Атрибуты отчета об ошибке

Основные атрибуты:

- Идентификатор (*id*)
- Краткое описание (*summary*)
- Подробное описание (*description*)
 - 1) Шаги воспроизведения (*steps to reproduce, STR*)
 - 2) Актуальный результат (*Actual Result*)
 - 3) Ожидаемый результат (*Expected Result*)
- Воспроизводимость (*reproducible*)
- Важность (*severity*)
- Срочность (*priority*)
- Симптом (*symptom*)

Атрибуты отчета об ошибке

Дополнительные (необязательные) атрибуты:

- Возможность «обойти баг» (*workaround*)
- Дополнительная информация (*additional information*)
- Приложения (*attachments*)

Идентификатор (*id*)

У каждого отчета об ошибке должен быть уникальный идентификатор.

Как правило, системы управления ошибками (*bug tracking systems*) позволяют формировать идентификатор в виде некоторого шаблона, например:

Аббревиатура проекта + дата + порядковый номер **VELIQ20160412C001**

или **VELIQ_20160412_C001**

Краткое описание (*summary*)

Хорошее краткое описание должно давать ответы на три вопроса:

Где?

В каком месте интерфейса или архитектуры программного продукта находится проблема.

Что?

Что происходит или не происходит согласно спецификации требований.

**Когда?
При каких
условиях?**

В какой момент работы программного продукта, по наступлению какого события или при каких условиях проблема проявляется.

Краткое описание (*summary*)



- Что:** Отсутствует (Нет) выпадающее меню
- Где:** в пункте Actions
- Когда:** при не выбранном документе.

Подробное описание (*description*)

В отличие от краткого описания, которое, как правило, является одним предложением, здесь можно и нужно давать подробную информацию. Хорошее подробное описание содержит необходимую информацию об ошибке, а также (обязательно!) описание ожидаемого результата, фактического результата и ссылку на требование (если это возможно).

Например:

«Если в систему входит администратор, в окне приветствия отсутствует логотип.

Ожидаемый результат:

Логотип присутствует в левом верхнем углу страницы.

Фактический результат:

Логотип отсутствует. Требование: R245.3.23b»

Шаги воспроизведения (*steps to reproduce, STR*)

- Несколько рекомендаций:
 - ❖ Описывайте каждый шаг, пока не столкнётесь с дефектом.
 - ❖ Найдите точный путь, чтобы воспроизвести дефект.
 - ❖ Попытайтесь найти кратчайший путь.
 - ❖ Повторите все описанные шаги несколько раз и убедитесь, что всё верно.
 - ❖ Описывайте каждое действие, которое вы делаете, в отдельном шаге.
- Ещё одна рекомендация (особенно актуальная для начинающих тестировщиков) состоит в том, чтобы последним шагом писать «Возникает ошибка <предельно сжатое описание ошибки>». Это позволяет исключить ситуацию, когда тестировщик забывает записать последний, самый важный шаг, который как раз и приводит к возникновению ошибки.

Шаги воспроизведения (*steps to reproduce, STR*)

- **Пример:**

1. Перейти по ссылке:
<http://www.site.com/login/>
2. Ввести в поле «Логин» значение «admin».
3. Ввести в поле «Пароль» значение «admpwd».
4. Кликнуть по кнопке «Войти».

Actual result: В левом верхнем углу вместо логотипа – пустое место

Expected result: В верхнем углу есть логотип

Воспроизводимость (*reproducible*)

Это поле показывает, воспроизводится ли баг всегда («always») или лишь иногда («sometimes»).

Баги, воспроизводящиеся всегда, гораздо проще диагностировать. Однако, очень важно подчеркнуть, что баг воспроизводится не каждый раз при выполнении шагов воспроизведения, если так и есть. Иначе программист сразу же поставит вашему багу статус Отклонён (*declined*) с резолюцией «Не могу воспроизвести» (*can not reproduce*), если при первом же проходе по шагам воспроизведения баг не появится.

Важность (*severity*)

Это поле показывает, насколько серьёзна найденная ошибка для функциональности приложения (конечного пользователя). Обычно, выделяют следующие уровни важности:

- **Критическая** (*critical*). Это ошибки, выражающиеся в крахе приложения или операционной системы, серьёзных повреждениях базы данных, падению веб-сервера или сервера приложений.
- **Высокая** (*major*). Серьёзные ошибки, такие как: потеря данных пользователя, падение значительной части функциональности приложения, падение браузера или иного клиента и т.п.
- **Средняя** (*medium*). Ошибки, затрагивающие небольшой набор функций приложения. Как правило, такие ошибки можно «обойти», т.е. выполнить требуемое действие иным способом, не приводящим к возникновению ошибки.
- **Низкая** (*minor*). Ошибки, не мешающие непосредственно работе с приложением. Как правило, сюда относятся всевозможные косметические дефекты, опечатки и т.п.

Срочность (*priority*)

Это поле показывает, как быстро необходимо исправить ошибку, указывает на серьезность дефекта с точки зрения его важности для проекта.

Обычно, выделяют такие значения срочности:

- **Наивысшая** (*ASAP, as soon as possible*). Присваивается ошибкам, наличие которых делает невозможным дальнейшую работу над проектом или передачу заказчику текущей версии проекта.
- **Высокая** (*high*). Присваивается ошибкам, которые нужно исправить в самое ближайшее время.
- **Обычная** (*normal*). Присваивается ошибкам, которые следует исправлять в порядке общей очереди.
- **Низкая** (*low*). Присваивается ошибкам, которыми отделу разработки следует заниматься в последнюю очередь (когда и если на них останется время).

Симптом (*symptom*)

Это поле показывает, к какой категории относится ошибка. Наиболее широко распространённые симптомы:

- **Косметический дефект** (*cosmetic flaw*).
- **Повреждение/потеря данных** (*data corruption/loss*).
- **Проблема в документации** (*documentation issue*).
- **Некорректная операция** (*incorrect operation*).
- **Проблема инсталляции** (*installation problem*).
- **Ошибка локализации** (*localisation issue*).
- **Нереализованная функциональность** (*missing feature*).
- **Низкая производительность** (*slow performance*).
- **Крах системы** (*system crash*).
- **Неожиданное поведение** (*unexpected behavior*).
- **Недружественное поведение** (*unfriendly behavior*).
- **Расхождение с требованиями** (*variance from spec*).
- **Предложение по улучшению** (*enhancement*).

Возможность «обойти баг» (*workaround*)

Это поле косвенно влияет на важность и срочность устранения ошибки.

Если некоторое действие можно выполнить в обход сценария, приводящего к ошибке, поле принимает значение «да» («yes»), в противном случае – поле принимает значение «нет» («no»).

Дополнительная информация

(additional info)


В это поле можно писать всё то, что вы считаете необходимым отметить, но что не подходит для размещения в других полях.

Рассуждения, комментарии, мысли, анализ возможных причин появления бага и путей его устранения – всё это пишется здесь.

Приложения (*attachments*)

Лучший способ указать на баг – приложить к баг-репорту некую наглядную информацию: скриншоты, видеоролики, логи (журналы событий).

Пример отчета об ошибке

Assigned To		
Priority	low	Severity
Status	new	Resolution
Product Version		
Target Version		Fixed in Version
Summary	0031339: при отсутствии сети - не выполняется переход по ссылке в новости	
Description	при отсутствии сети - не выполняется переход по ссылке (при просмотре новости)	
Steps To Reproduce	<ol style="list-style-type: none">1. Зайти в "Ведомости" - "Новости"2. Загрузить список новостей3. Перейти в просмотр конкретной новости. <p>Условие: отсутствие сети</p> <ol style="list-style-type: none">4. Нажать на ссылку любого сайта в загруженной новости <p>Результат: ничего не происходит</p> <p>Ожидаемый результат: Вариант 1. Появилось окно с уведомлением о невозможности подключения к сети Вариант 2. Выполнился переход по ссылке в браузер, кот. сообщил о невозможности</p>	
Tags	No tags attached.	
Attach Tags	(Separate by ",") <input type="text"/> Existing tags <input type="button" value="Attach"/>	
Attached Files	 IMG_4.PNG [^] (19,266 bytes) 2012-04-12 15:33 [Delete]	

Кто, что, для чего читает отчет?

➤ **Заказчик** – читает **заголовок** дефекта

Цель – понять, какие в проекте существуют проблемы

➤ **Руководитель разработки** – читает **заголовок** дефекта

Цель – понять, кому на исправление нужно направить дефект

Кто, что, для чего читает отчет?

➤ **Разработчик** – читает все **составляющие** дефекта

Цель – понять детали для исправления дефекта

➤ **Аналитик** – в зависимости от ситуации может читать различные **составляющие** дефекта

Цель – понять «масштаб бедствия»

Кто, что, для чего читает отчет?

➤ **Тестировщик** – читает все **составляющие** дефекта

Цель – воспроизвести дефект и проверить исправление

➤ **Руководитель QA** читает все **составляющие** дефекта

Цель – составление отчетов, контроль работы команды

Как обращаться с найденными багами?

У БАГОВ ТОЖЕ ЕСТЬ ЧУВСТВА

НАЙДЯ БАГ:
СООБЩИТЕ О НЕМ

БАГИ НЕ ЛЮБЯТ
БЫТЬ ЗАБЫТЫМИ



НАЙДЯ БАГ:
ИЗУЧИТЕ ЕГО

БАГИ ЛЮБЯТ
БЫТЬ ПОНЯТЫМИ



У нее 3
пятнышка

НАЙДЯ БАГ:
СДЕЛАЙТЕ СНИМОК

БАГИ ЛЮБЯТ ХРАНИТЬ
ПАМЯТЬ О ВСТРЕЧЕ



НАЙДЯ БАГ:
УЗНАЙТЕ ЕГО ДРУЗЕЙ

БАГИ ОБЩИТЕЛЬНЫ



НАЙДЯ БАГ:
СООБЩИТЕ О НЕМ БЫСТРО
ИНАЧЕ БАГИ ОБЖИВУТСЯ И
ПОСТРОЯТ СЕБЕ ДОМА



НАЙДЯ БАГ:
БУДЬТЕ ЧЕСТНЫ

БАГИ НЕ ЛЮБЯТ
СПЛЕТЕН



НАЙДЯ БАГ:
ЗАПОМНИТЕ КАК ВЫ
ВСТРЕТИЛИ ЕГО

БАГИ РОМАНТИЧНЫ



НАЙДЯ БАГ:
НЕ ИГНОРИРУЙТЕ ЕГО
БАГИ МОГУТ УКУСИТЬ,
ЕСЛИ ИХ НЕ ЦЕНИТЬ



Основные ошибки описания дефектов и как их избежать

Сокращение инструкции по воспроизведению ошибки:

- Использование сокращений
- Частое применение аббревиатур
- Опускание «маловажных» подробностей

Неправильно

1. Открыть СП
2. 5

Результат: грамматическая
ошибка

Правильно:

1. Запустить приложение
2. Открыть страницу помощи
3. Перейти на 5 страницу

Результат: грамматическая
ошибка в заголовке

Основные ошибки описания дефектов и как их избежать

Отсутствие описания ошибочного поведения:

- *Необходимо указывать, в чем ошибочность полученного результата!*

Неправильно

1. Запустить приложение
2. Нажать кнопку «Редактировать»

Результат: Форма для редактирования появляется

Правильно:

1. Запустить приложение
2. Нажать кнопку «Редактировать»

Результат: Форма для редактирования появляется, *все кнопки не активны*

Основные ошибки описания дефектов и как их избежать

Использование нечётких или неоднозначных формулировок:

Неправильно

1. Запустить приложение
2. Перейти в библиотеку
3. Выбрать любую книгу

Результат: книга
разблокирована

Правильно:

1. Запустить приложение
2. Перейти в библиотеку
3. Выбрать любую книгу

Результат: книга доступна для
редактирования

Основные ошибки описания дефектов и как их избежать

Ожидаемый результат слишком краток либо отсутствует:

Неправильно

Ожидаемый результат:
смотри спецификацию

Правильно:

Ожидаемый результат:
страница помощи должна открываться при помощи кнопки «HELP».
Смотри спецификацию – страница 10, раздел «Помощь», пункт 5.

Основные ошибки описания дефектов и как их избежать

Используются личные предложения, и не делается чёткого вывода, как должно быть реализовано:

Неправильно

Ожидаемый результат: я думаю, что должно быть ограничение на минимальный размер окна или уменьшение размера должно быть заблокировано

Правильно:

Ожидаемый результат: уменьшение размера должно быть заблокировано

Основные ошибки описания дефектов и как их избежать

Неинформативное описание и заголовок, нет точной причины:

Неправильно

Заголовок: функция приложения документов не соответствует спецификации

Описание: при добавлении файла, его путь не показывается

Правильно:

Заголовок: путь файла не показан на форме добавления документов

Описание:

1. Запустить приложение
2. Перейти на форму добавления документов
3. Добавить файл

Результат: Путь файла не показан. Отображено только имя.

Основные ошибки описания дефектов и как их избежать

Заголовок не должен содержать сленга!
Отсылка на приложенный файл к дефекту без описания, нет результата:

Неправильно

Заголовок: при
сворачивании прилаг
она крэшится

Результат:
смотри аттачмент 5

Правильно:

Заголовок: работа приложения
неожиданно
останавливается после
сворачивания

Описание:
1. Запустить приложение
2. Свернуть приложение

Результат: приложение
неожиданно
останавливается. Смотри
видео в приложении.

Какой отчёт об ошибке является плохим?

- Отчёт, который не даёт достаточной информации «Программа не работает», «Приложение виснет».
- Отчёт об ошибках в той части функциональности, которая не заявлена как реализованная в данном билде.
- Отчёт, дающий некорректную информацию (в любом из полей).
- Отчёт, написанный с грамматическими ошибками и/или с использованием жаргонной, непонятной большинству лексики.
- Отчёт, критикующий работу программиста.

Формула совершенного отчета об ошибке

- Формула совершенного баг-репорта состоит из трёх простых пунктов:
 - Что мы сделали (*steps required to reproduce the problem*)
 - Что мы получили (*actual results*)
 - Что мы ожидали получить (*expected results*)
- Кроме того, нужно сообщить, где именно произошла проблема, при каких условиях, а также дать ошибке название.

В каких случаях баг может остаться неисправленным?

- Программист так и не смог воспроизвести у себя ошибку по той или иной причине.
- Ошибке выставлены неверная важность и/или приоритет.
- Отсутствует описание некорректного поведения (актуального результата).
- Отсутствует описание ожидаемого результата или оно не обосновано (нет ссылок на требования).
- Отчёт написан безграмотно, расплывчато, непонятно.

В каких случаях баг может остаться неисправленным?

- Отсутствуют необходимые для понимания ошибки скриншоты, логи и т.д.
- Для описания новой ошибки, похожей на старую, тестировщик сделал «повторное открытие» уже исправленной ошибки.
- Тестировщик не сумел убедить команду в важности проблемы.
- У тестировщика плохая репутация.

Рекомендации по написанию хороших отчётов об ошибках

- Тщательно объясните, как воспроизвести ошибку. Сообщите всю необходимую для этого информацию, а также свои размышления о возможных причинах возникновения ошибки.
- Описывайте всё максимально подробно. Особенно это относится к ожидаемому и фактическому результатам, а также шагам воспроизведения.
- Пишите отчёт понятно. Используйте общеупотребимую лексику, точные названия элементов ПС, аккуратно оформляйте написанное.
- Если это возможно, обязательно давайте ссылку на соответствующее требование, к нарушению которого приводит фактический результат выполнения ПО.

Рекомендации по написанию хороших отчётов об ошибках

- Если существует какая-либо информация, которая может помочь быстро обнаружить и исправить ошибку, – сообщите эту информацию.
- Чётко указывайте окружение (ОС, браузер, настройки и т.п.), под которым произошла ошибка.
- **В одном отчёте описывайте ровно одну проблему. Если вы видите две ошибки – пишите два отчёта.**
- Если вам хватает знаний, проведите начальный анализ возможных причин возникновения ошибки и опишите его результаты в разделе «Комментарии».
- Попробуйте найти наиболее серьёзные последствия ошибки. Возможно, то, что казалось незначительным вначале, на самом деле может привести к очень серьёзным неприятностям.

Рекомендации по написанию хороших отчётов об ошибках

- Пишите отчёт об ошибке сразу же, как только вы обнаружили ошибку. Откладывание записи «на потом» приводит к тому, что вы или вообще забудете об этой ошибке, или забудете о каких-то важных деталях. Также несвоевременное написание отчёта об ошибке не позволяет проектной команде реагировать на её обнаружение в реальном времени.
- После написаний отчёта ещё раз внимательно его перечитайте. Убедитесь, что все необходимые поля заполнены, и всё написано верно.
- Помните, что вам же самим потом придётся верифицировать баг по своему же баг-репорту.

Преимущества хорошего отчёта об ошибках

Хороший отчёт об ошибках помогает:

- Сократить количество ошибок, «возвращаемых» разработчиками (отклонённых или открытых заново).
- Ускорить устранение ошибки.
- Сократить стоимость исправления ошибки.
- Повысить репутацию тестировщика.
- Улучшить взаимоотношения между командами тестирования и разработки.

Баг-трекинг-овые системы

- JIRA
- Redmine
- Bugzilla
- QC

Контрольные вопросы:

1. Как определить дефект в программе: есть или нет?
2. Описать жизненный цикл бага.
3. Какие состояния багов считаются открытыми и закрытыми?
4. Что представляет собой баг-репорт? Основная цель написания.
5. Описать основные атрибуты баг-репорта.