Вопрос 5. Исправление дефектов, зафиксированных в базе данных

1. Основные понятия

Error (Пользователя) — действие человека, которое приводит к неожиданному результату (к сбою). В качественной программе предусмотрены такие ситуации и выдаются сообщение об ошибке (error message).

Bug (defect) — изъян в компоненте или системе который ведет к отклонению фактического результата выполнения программы от ожидаемого (к сбою).

Failure (Сбой) — отклонение компонента или системы от ожидаемого выполнения, эксплуатации или результата.

Баг Репорт (Bug Report) — отчет об ошибке, документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

Серьезность (Severity) — это атрибут, характеризующий влияние дефекта на работоспособность приложения. Градации серьезности дефекта:

S1 Блокирующая (Blocker) — блокирующая ошибка, приводящая приложение в нерабочее состояние, в результате которого дальнейшая работа с тестируемой системой или ее ключевыми функциями становится невозможна. Решение проблемы необходимо для дальнейшего функционирования системы.

S2 Критическая (Critical) — критическая ошибка, неправильно работающая ключевая бизнес-логика, дыра в системе безопасности, проблема, приведшая к временному падению сервера или приводящая в нерабочее состояние некоторую часть системы, без возможности решения проблемы, используя другие входные точки. Решение проблемы необходимо для дальнейшей работы с ключевыми функциями тестируемой системой.

S3 Значительная (Major) — значительная ошибка, часть основной бизнес логики работает некорректно. Ошибка не критична или есть возможность для работы с тестируемой функцией, используя другие входные точки.

S4 Незначительная (Minor) — незначительная ошибка, не нарушающая бизнес логику тестируемой части приложения, очевидная проблема пользовательского интерфейса.

S5 Тривиальная (Trivial) — тривиальная ошибка, не касающаяся бизнес логики приложения, плохо воспроизводимая проблема, малозаметная посредствам пользовательского интерфейса, проблема сторонних библиотек или сервисов, проблема, не оказывающая никакого влияния на общее качество продукта.

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. Градации приоритета дефекта:

P1 Высокий (High) Ошибка должна быть исправлена как можно быстрее, т.к. ее наличие является критической для проекта.

P2 Средний (Medium) Ошибка должна быть исправлена, ее наличие не является критичной, но требует обязательного решения.

P3 Низкий (Low) Ошибка должна быть исправлена, ее наличие не является критичной, и не требует срочного решения. .

Серьезность выставляется тестировщиком, а приоритет— менеджером, тимлидом или заказчиком

2. Управление дефектами в тестировании программного обеспечения

Баг/ошибка является следствием ошибки кодирования, в то время как дефект - это отклонение от первоначальных бизнес-требований.

Эти два термина имеют очень тонкую черту различия. Оба эти сущности являются недостатками, которые должны быть исправлены и, поэтому часто эти термины используются взаимозаменяемо.

Когда тестер выполняет тестовые случаи, он может столкнуться с результатом теста, который противоречит ожидаемому результату. Это изменение в результатах теста называется дефектом программного обеспечения. Эти дефекты или вариации обозначаются разными именами в разных организациях, таких как проблемы, проблемы, ошибки или инциденты.

Сообщение об ошибке направляется разработчику в виде отчета, который должен содержать следующую информацию:

1. Шапка — короткое описание (Summary) проблемы, явно указывающее на причину и тип ошибочной ситуации. В шапку входит следующая информация:

- Defect_ID - уникальный идентификационный номер для дефекта.
- Дата повышения - дата возникновения дефекта.
- Проект (Project) — название тестируемого проекта.
- Компонент приложения (Component) — название части или функции тестируемого продукта.
- Номер версии (Version) версия на которой была найдена ошибка.
- Серьезность (Severity): как влияет дефект на приложение в целом.
- Приоритет, связанный с срочностью устранения дефектов.
- Статус (Status) — зависит от используемой процедуры и жизненного цикла бага (bug workflow and life cycle).
- Автор (Author) — создатель баг репорта.
- Назначен на (Assigned To) — имя сотрудника, назначенного на решение проблемы.
- Окружение (Environment) — Информация об окружении, на котором был найден баг: операционная система, сервис пак, для WEB тестирования — имя и версия браузера (ОС / Сервис Пак и т.д. / Браузера + версия/).
- Ссылка на документы

2. Описание (Description) - подробное описание дефекта, включая информацию о модуле, в котором обнаружен дефект. Требования, дизайн, архитектура или даже скриншоты ошибки, чтобы помочь понять дефект.

- Шаги воспроизведения (Steps to Reproduce), подробные, со скриншотами, по которым можно легко воспроизвести ситуацию, приведшую к ошибке.
- Фактический Результат (Result), полученный после прохождения шагов к воспроизведению.
- Ожидаемый результат (Expected Result) — ожидаемый правильный результат

3. Дополнения

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

Обнаружено - имя / идентификатор тестера, который выявил дефект

Состояние - состояние дефекта.
Исправлено - Имя / ID разработчика, который это исправил
Дата закрытия - дата закрытия дефекта

Без составления отчетов, сообщения о дефектах делаются устно, и вскоре все становится очень сложным. Для контроля и эффективного управления ошибками необходим четкий процесс.

Пример: *Тестировщики нашли 84 дефекта, менеджер проекта передает список разработчикам. Через неделю разработчики исправляют 61 дефект и передают информацию менеджеру проекта. Тестировщики повторно тестируют и отвечают: мы пофиксили 61 дефекта, но нашли 10 новых. И так по кругу.*

Управление дефектами - это систематический процесс выявления и устранения ошибок.

Цикл управления дефектами содержит следующие этапы:

1. Обнаружение дефекта
2. Категоризация дефекта
3. Устранение дефекта разработчиками
4. Проверка тестерами
5. Закрытие дефекта
6. Отчеты о дефектах в конце проекта

На этапе обнаружения проектные группы должны обнаружить как можно больше дефектов, прежде чем конечный клиент сможет их обнаружить. Считается, что дефект обнаружен и изменен на статус принятого, когда он признан и принят разработчиками.

Пример: *Группа тестирования обнаружила некоторые проблемы по проекту. Они рассматривают их как дефекты и сообщают команде разработчиков, но есть проблема: разработчики отвечают, что это не дефекты и требуют повторного тестирования. В таком случае, менеджер проекта должен взять на себя роль судьи и принять решение, является ли проблема продукта дефектом или нет. При работе в системе управления дефектами таких проблем просто не может возникнуть.*

Классификация дефектов помогает разработчикам программного обеспечения определять приоритеты своих задач и в первую очередь устранить те дефекты, которые больше прочих угрожают работоспособности продукта.

Критический дефект должен быть устранен немедленно, иначе это может привести к большим потерям для продукта.

Например: *функция входа на сайт банка не работает должным образом. Вход в систему является одной из основных функций банковского сайта, если эта функция не работает, это серьезные ошибки.*

Высокий дефект негативно влияет на основные функции продукта **Например:** *производительность сайта слишком низкая.*

Средний дефект вносит минимальные отклонения от требований к к продукту **Например:** *не корректно отображается интерфейс на мобильных устройствах.*

Низкий — оказывает минимальное нефункциональное влияние на продукт.

Например: некоторые ссылки не работают.

После того, как дефекты приняты и классифицированы, вы можете выполнить следующие шаги, чтобы исправить их:

- Назначение: проблемы отправлены разработчику или другому техническому специалисту для исправления и изменило статус на отвечающий.
- График устранения: сторона разработчика берет на себя ответственность на этом этапе, они создадут график для устранения этих дефектов в зависимости от их приоритета.
- Исправление: пока группа разработчиков устраняет дефекты, диспетчер тестов отслеживает процесс устранения проблем, исходя из графика.
- Сообщить о решении: получите отчет об устранении бага от разработчиков, когда дефекты устранены.

После того, как команда разработчиков исправила дефект и сообщила об этом, команда тестирования **осуществляет верификацию,** то есть проверяет, действительно ли устранены все заявленные дефекты.

После устранения и проверки дефекта его статус меняется на закрытый. Если он не устранен, вы отправляете уведомление в отдел разработки, чтобы они проверили дефект еще раз.

Далее необходимо составить отчет: сообщить правлению текущую ситуацию с дефектами, чтобы получить от них обратную связь. Они должно видеть и понимать процесс управления дефектами, чтобы поддержать вас в случае необходимости.

Показатели дефекта можно измерить, оценив тем самым качество выполнения теста:

- Коэффициент отклонения брака (Defect reject ratio, DRR) = количество отклоненных отчетов об ошибках, деленное на общее количество отправленных отчетов об ошибках.
- Коэффициент утечки дефектов (Defect leakage ratio, DLR) = количество пропущенных при тестировании дефектов/общее количество дефектов.

Чем меньше значение DRR и DLR, тем, соответственно, лучше качество тестирования. Приемлемый диапазон отклонений может быть определен и принят за основу исходя из целей проекта, либо аналогичных ему.

Чтобы уменьшить эти коэффициенты, необходимо:

- улучшать навыки тестирования участников проекта,
- тратить больше времени на выполнение тестов и просмотр результатов.

Пример: *предположительно, в продукте всего 64 дефекта, но ваша группа по тестированию обнаружила только 44 дефекта, т.е. они пропустили 20 дефектов. Следовательно, DLR = 20/64 = 0,312 (31,2%).*

Рекомендуемое значение показателей качества тестирования должно составлять 5 ~ 10%. Это означает, что качество выполнения теста низкое.

Литература:

1. Тестирование. Фундаментальная теория / https://qa-uide.ru/forums/topic/teorija_testirovanija/
2. Управление дефектами в тестировании программного обеспечения / https://logrocon.ru/news/defects