Модуль 2

Дефекты: понятие, виды,
жизненный цикл. Test result report

Agenda

- Понятия дефекта: жизненный цикл, виды
- Test result report

STLC. Жизненный цикл тестирования

STLC (Software testing life cycle) - означает процесс тестирования программного обеспечения, разделенный на определенные этапы, которые выполняются последовательно с целью улучшения качества тестируемой системы/продукта.

Основные фазы :

- ~~Анализ требований~~
- ~~Планирование~~
- ~~Разработка тест кейсов~~
- ~~Развертывание тестового окружения~~
- **Тестирование**
- **Завершение жизненного цикла тестирование (анализ и отчетность)**

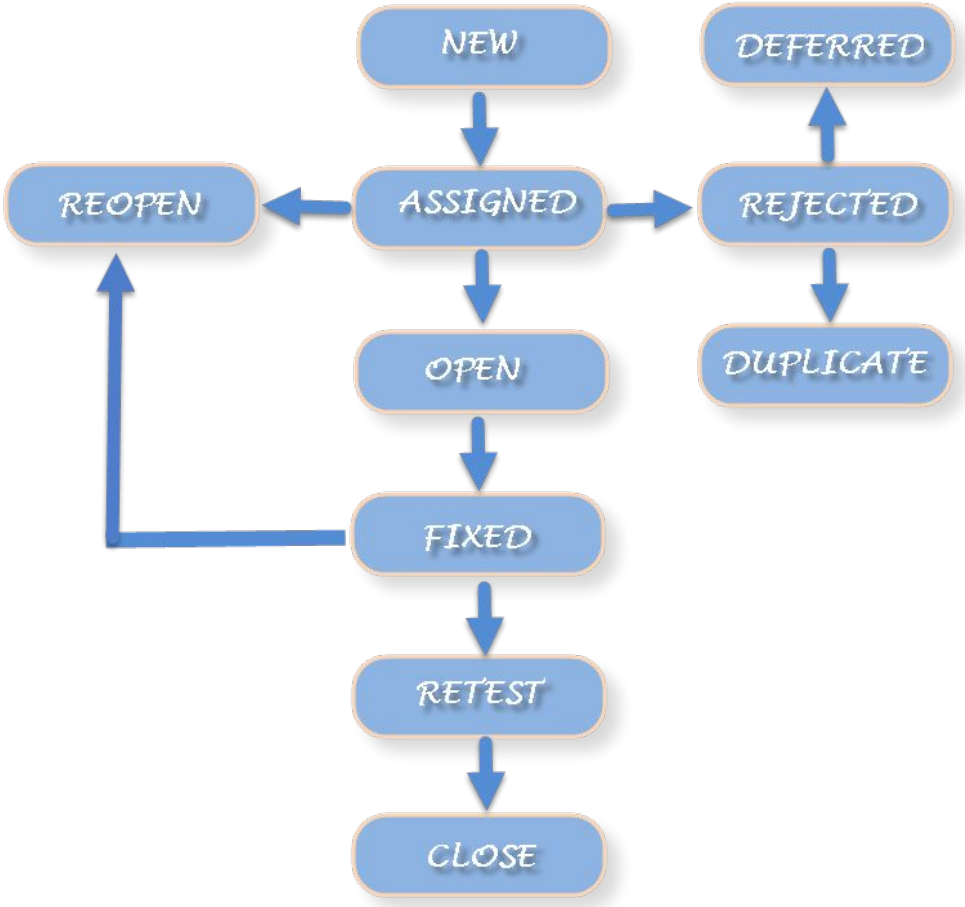
Понятие дефекта (бага) и его жизненный цикл

Дефект в тестировании программного обеспечения — это изменение или отклонение программного приложения от требований конечного пользователя или исходных бизнес-требований.

Дефект программного обеспечения — это ошибка в кодировании, которая приводит к неправильным или неожиданным результатам в программе, которая не соответствует фактическим требованиям. Тестировщики могут столкнуться с такими дефектами при выполнении тест-кейсов.

Каждый дефект проходит определенный жизненный цикл.

Жизненный цикл дефекта, представляет собой цикл этапов, которые он проходит, охватывая различные состояния на протяжении всей своей жизни. Это начинается, как только тестировщик обнаруживает какой-либо новый дефект, и заканчивается, когда тестировщик закрывает этот дефект, гарантируя, что он не будет воспроизведен снова.



Стадии жизненного цикла:

1) Новый
2) Назначен (на данной стадии дефект может быть Отклонен, Отложен либо признан Дубликатом)
3) Открыт
4) Исправлен
5) Повторное тестирование
6) Повторное открытие (если необходимо)
7) Закрыт (проверен)

1) **Новый.** При обнаружении любого нового дефекта он переходит в состояние «Новый».

2) **Назначен:** на этом этапе вновь созданный дефект назначается команде разработчиков для работы над дефектом. Это назначается руководителем проекта или руководителем группы тестирования разработчику.

3) **Открытый:** здесь разработчик начинает процесс анализа дефекта и при необходимости работает над его исправлением. Если разработчик считает, что дефект неуместен, он может быть переведен в любое из следующих четырех состояний, а именно: «Дубликат», «Отложено», «Отклонено» или «Не ошибка», в зависимости от конкретной причины.

4) **Исправлено:** когда разработчик завершает задачу по исправлению дефекта, внося необходимые изменения, он может пометить статус дефекта как «Исправлено». После исправления дефекта разработчик назначает дефект тестировщику для повторного тестирования.

5) **Повторное тестирование:** на этом этапе тестер проводит повторное тестирование дефекта, чтобы проверить, точно ли дефект исправлен разработчиком в соответствии с требованиями или нет.

6) **Повторное открытие**: если какая-либо проблема не устранена в дефекте, она будет снова назначена разработчику для исправления, а статус дефекта изменится на «Повторно открыть».

7) **Закрыт:** если тестер не находит проблем в дефекте после повторного тестирования, и он считает, что дефект был исправлен точно, тогда дефекту присваивается статус «Проверено». Когда дефект больше не существует, тестер меняет статус дефекта на «Закрыт».

Типы дефектов (багов)

Дефекты производительности: программного обеспечения, которые приводят к снижению скорости, стабильности, увеличению времени отклика и повышенному потреблению ресурсов, считаются ошибками производительности. Наиболее важным признаком любой такой ошибки в программном обеспечении является обнаружение более медленной скорости загрузки, чем обычно, или анализ времени отклика.

Дефекты безопасности: Программное обеспечение с плохой безопасностью не только поставит под угрозу данные пользователя, но и нанесет ущерб общему имиджу организации, на восстановление которого могут уйти годы. Из-за своей высокой серьезности ошибки безопасности считаются одними из самых чувствительных ошибок всех типов. Хотя это не требует пояснений, ошибки безопасности могут сделать программное обеспечение уязвимым для потенциальных киберугроз.

Функциональные дефекты: Если какая-либо функциональность программного обеспечения будет скомпрометирована, количество пользователей начнет резко сокращаться, пока оно снова не станет функциональным. Функциональная ошибка — это когда определенная функция или все программное обеспечение не работает должным образом из-за дефекта. Серьезность таких ошибок зависит от функции, которой они мешают. Например, не отвечающая на нажатие кнопка, которая не работает, не так серьезна, как неработающее все программное обеспечение.

Дефекты юзабилити: Дефекты юзабилити делают приложение неудобным в использовании и, таким образом, затрудняют работу пользователя с программным обеспечением. Макет контента, который сложно просматривать или перемещать, а также слишком сложная процедура регистрации являются примерами дефектов юзабилити.

Синтаксические дефекты: Синтаксические ошибки относятся к наиболее распространенным типам ошибок программного обеспечения и не позволяют правильно скомпилировать приложение. Эта ошибка возникает из-за неправильного или отсутствующего символа в исходном коде, из-за которого будет затронута компиляция.

Дефекты совместимости: Всякий раз, когда программное обеспечение или приложение несовместимо с аппаратным обеспечением или операционной системой, это считается несовместимым программным обеспечением или ошибкой совместимости. Приложение с ошибками совместимости не показывает стабильной производительности на определенных типах оборудования, операционных систем, браузеров и устройств или при интеграции с определенным программным обеспечением или при работе в определенных сетевых конфигурациях.

Логические дефекты: приводят к неправильному выводу программного обеспечения или сбою программного обеспечения. В большинстве случаев эти ошибки вызваны ошибками кодирования, которые могут привести к тому, что программное обеспечение застрянет в бесконечном цикле загрузки. В этом случае только внешнее прерывание или программный сбой могут нарушить цикл загрузки.

Дефекты на основе приоритета, они основаны на влиянии этих ошибок на бизнес.

Дефекты с низким приоритетом (Low-priority): Дефекты с низким приоритетом не оказывают большого влияния на работу приложения. Скорее, они больше касаются эстетики программного обеспечения. Например, любая проблема с правописанием или выравниванием кнопки или текста может быть дефектом с низким приоритетом.

Дефекты среднего приоритета(Medium-priority): дефекты со средним приоритетом не оказывают существенного влияния на программное обеспечение, но они должны быть исправлены в любом последующем или предстоящем выпуске.

Дефекты с высоким приоритетом(High-priority): Каждая ошибка, попадающая в эту категорию, может сделать некоторые функции ПО непригодными для использования. Примером дефекта с высоким приоритетом является то, что приложению не удается провести пользователя со страницы входа на домашнюю страницу, даже если пользователь ввел действительные данные для входа.

Срочные дефекты(Urgent): Как следует из названия, в эту категорию попадают все ошибки, которые должны быть устранены в срочном порядке. Срочные дефекты могут оказать длительное влияние на имидж бренда, а также резко повлиять на взаимодействие с пользователем. Установленный срок для исправления этих ошибок — в течение 24 часов с момента сообщения.

Дефекты по степени серьезности. В зависимости от технического воздействия, которое ошибка вызовет в программном обеспечении, ошибки подразделяются на четыре категории.

Дефекты низкой серьезности (Low Severity): не наносят большого ущерба функционированию программного обеспечения, поскольку их основной целью является пользовательский интерфейс. Например, шрифт текста в программе отличается от того, что использовался. Эти ошибки можно легко исправить.

Дефекты средней степени серьезности (Medium Severity): Каждая ошибка, которая может немного повлиять на функциональность программного обеспечения. Все такие ошибки делают работу программного обеспечения отличной от той, для которой она должна функционировать. Хотя они также не являются важными для программы, их следует исправить для лучшего взаимодействия с пользователем.

Дефекты высокой степени серьезности (High Severity): Ошибки высокой степени серьезности влияют на функциональность программного обеспечения, заставляя его вести себя иначе, чем то, для чего оно было запрограммировано. например, поставщик услуг электронной почты не разрешает добавлять более одного адреса электронной почты в поле получателя

Критические дефекты(Critical): Причина, по которой критические ошибки считаются наиболее разрушительными, заключается в том, что дальнейшее тестирование программного обеспечения становится невозможным до тех пор, пока такие ошибки не будут обнаружены в программном обеспечении. Примером критического дефекта является возврат приложением сообщения об ошибке сервера после попытки входа в систему.

Баг-репорт (отчет о дефекте)

Отчет о дефекте в тестировании программного обеспечения — это подробный документ о дефектах, обнаруженных в программном приложении. Отчет о дефекте содержит все необходимые подробности: дата, когда дефект был обнаружена, имя тестировщика, который его нашел, имя разработчика, который его исправил, и т. д. Отчет о дефекте помогает идентифицировать подобные дефекты в будущем, чтобы их можно было избежать.

Одним из наиболее важных навыков, которые вам необходимо иметь в наборе инструментов тестировщика, является умение писать хорошие отчеты об ошибках. Поиск дефектов — это только часть работы, потому что, если разработчики не смогут воспроизвести обнаруженные вами ошибки, им будет очень трудно их исправить.

Итак, рассмотрим, что должен содержать качественный баг-репорт.

Описательное название. Все начинается с заголовка. Он должно быть четким и описательным, чтобы можно было с первого взгляда понять, о чем идет речь. Заголовок должен отвечать на три вопроса: Что? Где?При каких условиях?

Краткое описание. Описание должно быть ясным и кратким. Человек, читающий ваше описание, не видел бага и из описания должен понять в чем суть.(опциональное поле)

Ожидаемые результаты и актуальные результаты. Вы должны прояснить, чего вы ожидали, и как дефект отклонился от этого ожидания. Это поле помогает предотвратить любые недоразумения и дает разработчику четкое представление о том, что пошло не так.

Подробности о проекте и версии. Для тестировщиков нет ничего необычного в том, что они работают над несколькими проектами одновременно, а иногда они используют общую базу данных, поэтому очень важно убедиться, что ваша ошибка указана в правильном проекте и разделе в этом проекте. Вы также должны получить правильную версию программного обеспечения.

Сведения о тестовом окружении. Любая справочная информация, которую вы можете предоставить о своей среде, поможет разработчикам отследить эту ошибку. Какое устройство вы использовали? Какая операционная система работала? Какая это была модель? Какой браузер вы использовали? Каждая деталь, которую вы можете сообщить о тестовой среде, поможет.

Например: тип устройства, модель, версия браузера и версия ОС.

Тип дефекта и серьезность. Разработчики также должны знать, насколько серьезна ошибка. Ни один продукт не поставляется без дефектов, поэтому правильная классификация ошибок с точки зрения типа и серьезности действительно помогает процессу принятия решений относительно того, что исправлять, а что нет. Поэтому всегда лучше указывать приоритет и серьезность ошибок.

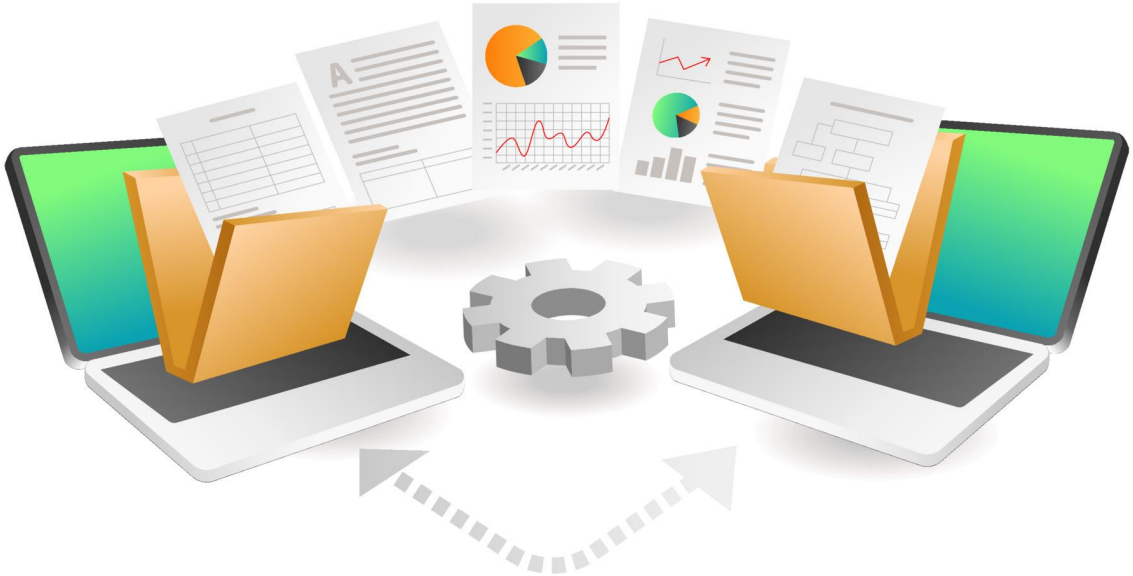
Шаги для воспроизведения (это очень важно!). Необходимо предоставить детальный пошаговый отчет о том, что именно было сделано, чтобы найти дефект. Чтобы не забыть все шаги всегда делайте заметки во время тестирования. Обязательно указывайте все необходимые URL.

Визуальные приложения. Вспомогательный материал всегда с благодарностью принимается теми, кому поручено исправлять дефекты. Обычно это скриншоты, но могут быть и аудио- и видеофайлы. Имейте в виду, что это часто то, на что разработчики смотрят в первую очередь, прежде чем читать предоставленный вами текст. Очень важно правильно подписывать приложения, чтобы разработчикам было легче в них ориентироваться.

Теги и ссылки. Может быть хорошей идеей использовать описательные теги, которые позволяют фильтровать базу данных дефектов и находить группы связанных ошибок.

Дефект обязательно должен быть назначен на разработчика (или н атого человека как принято на проекте), чтобы исключить возможность того, что баг может затеряться.

Завершение жизненного цикла тестирование (анализ и отчетность)



На данном этапе подготавливается отчет о завершении тестирования (отчет о результатах тестирования- Test Result Report).

Отчет о тестировании — это документ, который содержит сводку всех тестовых действий и окончательных результатов тестирования проекта. Отчет о тестировании — это оценка того, насколько хорошо проведено тестирование. На основании отчета о тестировании заинтересованные стороны (заказчик, проджект менеджер, команда разработки) могут оценить качество тестируемого продукта и принять решение о выпуске программного обеспечения.

Например, если в отчете о тестировании сообщается, что в продукте осталось много дефектов, заинтересованные стороны могут отложить выпуск до тех пор, пока все дефекты не будут исправлены.

Основные секции отчета о результатах тестирования:

1.Информация о проекте. Вся информация о проекте, такая как название проекта, название продукта и версия, должна быть описана в отчете.

2. Цель тестирования. Отчет о тестировании должен включать цель каждого раунда тестирования, такого как модульное тестирование, тестирование производительности, системное тестирование и т. д.

3. Краткое содержание тестирования. Этот раздел включает в себя сводку деятельности по тестированию в целом. Подробная информация здесь включает:

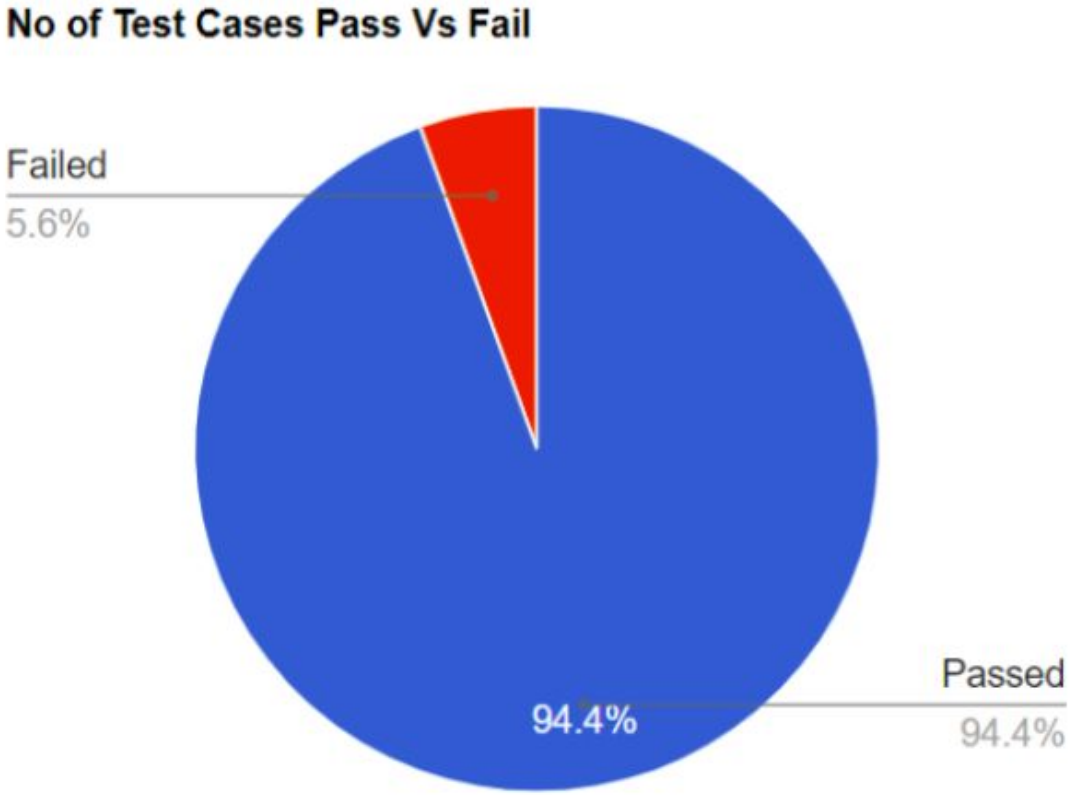
- Количество выполненных тест кейсов
- Количество пройденных тестов
- Количество неудавшихся тест кейсов
- Процент прохождения
- Комментарии

Эта информация должна отображаться визуально с помощью цветового индикатора, графика и выделенной таблицы.

Примеры:

Total No of Test Cases Planned	Total No Of Test Cases Executed	Total No Of Passed Test Cases	Total No Of Failed Test Cases
100	90	85	5

Примеры визуального отображения в виде графиков:



4. Одной из наиболее важных сведений в отчете о тестировании является дефект. Отчет должен содержать следующую информацию:

- Общее количество дефектов
- Статус дефектов (открыто, закрыто)
- Количество открытых, исправленных, закрытых дефектов
- Разбивка по серьезности и приоритету
- Могут быть включены некоторые метрики, такие как % исправленных дефектов.

Информация может быть представлена в виде таблицы либо графика для большей наглядности.

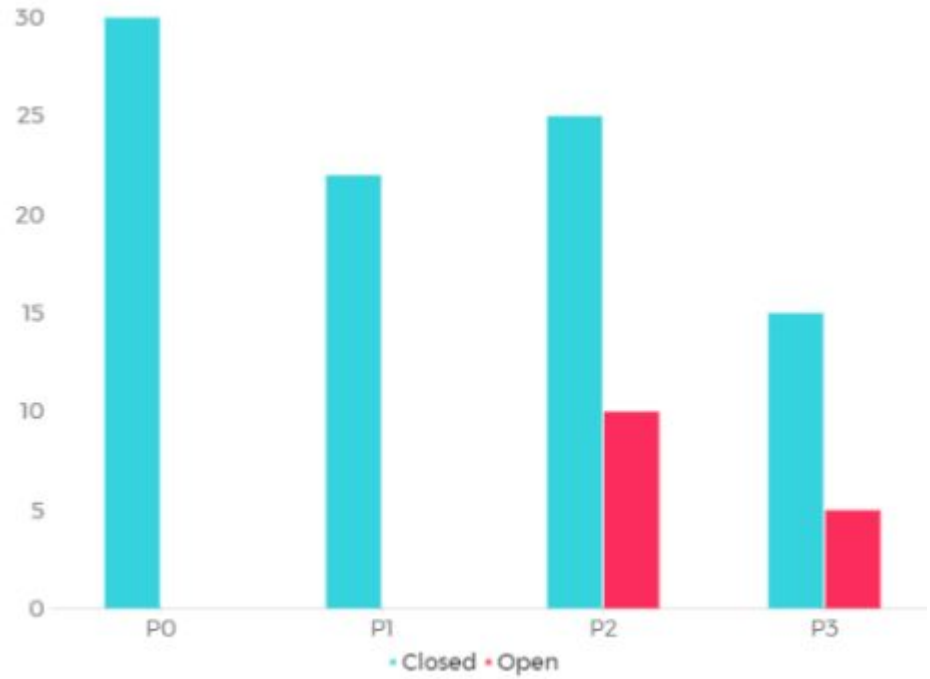
Отчет о результатах тестирования должен:

- подробное описывать деятельность по тестированию, показать, какие тесты выполнялись.
- вся информация в отчете должна быть краткой и понятной.
- отчет должен соответствовать стандартному шаблону, принятому на проекте.
- в отчет могут быть включены какие-то рекомендации по улучшению процесса тестирования, взаимодействия с командой и тд .
- обычно предоставляется после окончания процесса тестирования, то есть в конце спринта, либо могут быть иные сроки.

Примеры отображения дефектов по их статусу и приоритету (серьезности):

	P0	P1	P2	P3	Total
Closed	30	22	25	15	92
Open	0	0	10	5	15

Примеры визуального отображения дефектов по их статусу и приоритету(серьезности):




THE END