# Отчёты о дефектах

Баг(дефект)— расхождение ожидаемого и фактического результата.

Ожидаемый результат — поведение системы, описанное в требованиях.

Фактический результат — поведение системы, наблюдаемое в процессе тестирования.

## Виды ошибок

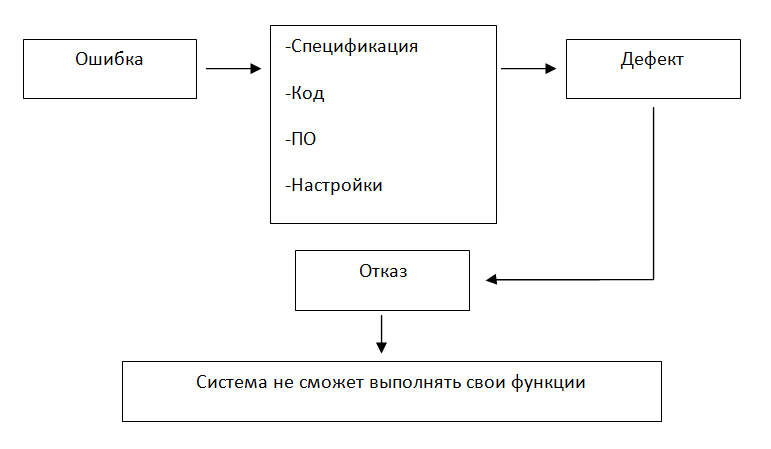


Рис. 3.1. Виды ошибок

Ошибка — действие человека, приводящее к некорректным результатам. Этот термин очень часто используют как наиболее универсальный, описывающий любые проблемы («ошибка человека», «ошибка в коде», «ошибка в документации», «ошибка выполнения операции», «ошибка передачи данных», «ошибочный результат» и т.п.). Ошибки, сделанные программистом, известны как «Ошибка», что ,в свою очередь, приводит к ошибке в программе.

Дефект — это отклонение от требований спецификаций или ожиданий пользователей. Другими словами, дефект является ошибкой в ​​кодировании или логике, что приводит к сбою программы или созданию неправильного / неожиданного результата. Это могут быть аппаратные средства, программное обеспечение, сеть, производительность, формат или функциональность. Недостаток в компоненте или системе, способный привести к ситуации сбоя или отказа.

Сбой — самоустраняющийся отказ или однократный отказ, устраняемый незначительным вмешательством оператора.

Отказ — событие, заключающееся в нарушении работоспособного состояния объекта. Это неспособность системы или компонента выполнять требуемые функции в рамках определенных требований к производительности. Неисправность возникает при выполнении ошибки.

Эти термины скорее относятся к теории надёжности и нечасто встречаются в повседневной работе тестировщика, но именно сбои и отказы являются тем, что тестировщик замечает в процессе тестирования (и отталкиваясь от чего, проводит исследование с целью выявить дефект и его причины).

Отчёт о дефекте - это документ, описывающий ситуацию или последовательность действий приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

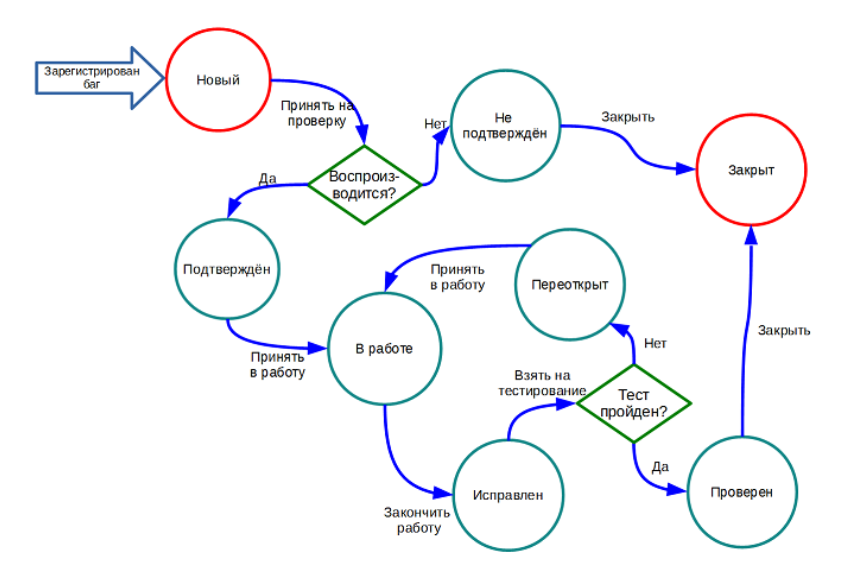
Цели написания отчёта о дефекте

Как следует из самого определения, отчёт о дефекте пишется со следующими основными целями:

* Предоставить информацию о проблеме — уведомить проектную команду и иных заинтересованных лиц о наличии проблемы, описать суть проблемы.
* Приоритизировать проблему — определить степень опасности проблемы для проекта и желаемые сроки её устранения.
* Содействовать устранению проблемы — качественный отчёт о дефекте не только предоставляет все необходимые подробности для понимания сути случившегося, но также может содержать анализ причин возникновения проблемы и рекомендации по исправлению ситуации.

# Жизненный цикл “бага”

#### Отчёт о дефекте (и сам дефект вместе с ним) проходит определённые стадии жизненного цикла:

* Обнаружен (submitted) — начальное состояние отчёта (иногда называется «Новый» (new)), в котором он находится сразу после создания. Некоторые средства также позволяют сначала создавать черновик (draft) и лишь потом публиковать отчёт.
* Назначен (assigned) — в это состояние отчёт переходит с момента, когда кто-то из проектной команды назначается ответственным за исправление дефекта. Назначение ответственного может производится решением лидера команды разработки, коллегиально, по добровольному принципу, иным принятым в команде способом или выполняется автоматически на основе определённых правил.
* Исправлен (fixed) — в это состояние отчёт переводит ответственный за исправление дефекта член команды после выполнения соответствующих действий по исправлению.
* Проверен (verified) — в это состояние отчёт переводит тестировщик, удостоверившись, что дефект на самом деле был устранён. Как правило, такую проверку выполняет тестировщик, изначально написавший отчёт о дефекте.
* Закрыт (closed) — состояние отчёта, означающее, что по данному дефекту не планируется никаких дальнейших действий. Здесь есть некоторые расхождения в жизненном цикле, принятом в разных инструментальных средствах управления отчётами о дефектах.
* Открыт заново (reopened) — в это состояние (как правило, из состояния «Исправлен») отчёт переводит тестировщик, удостоверившись, что дефект по-прежнему воспроизводится на билде, в котором он уже должен быть исправлен.
* Рекомендован к отклонению (to be declined) — в это состояние отчёт о дефекте может быть переведён из множества других состояний, чтобы вынести на рассмотрение вопрос об отклонении отчёта по той или иной причине. Если рекомендация является обоснованной, то отчёт переводится в состояние «Отклонён».
* Отклонён (declined) — в это состояние отчёт переводится в случаях, подробно описанных в пункте «Закрыт»: если средство управления отчётами о дефектах предполагает использование этого состояния вместо состояния «Закрыт» для тех или иных резолюций по отчёту.
* Отложен (deferred) — в это состояние отчёт переводится в случае, если исправление дефекта в ближайшее время является нерациональным или не представляется возможным, однако есть основания полагать, что в обозримом будущем ситуация исправится (выйдет новая версия библиотеки, вернётся из отпуска специалист по необходимой технологии, изменятся требования заказчика и т.д.)Рис. 3.2. Жизненный цикл отчёта о дефекте

#### Использование данных отчета о дефекте

Информация по дефектам – это записи о просчетах в качестве, а значит – список возможностей для улучшения качества на ваших будущих проектах. Если вы собираете некую информацию о дефектах (например, после релиза или только на больших проектах), тогда, возможно, Вы захотите улучшить этот процесс.

Информация о дефектах, которая может быть полезна для улучшения качества, включает следующие вопросы:

* Что было не так? Решать нужно саму проблему, а не ее симптомы. Например, зацикливание - это симптом, а изменение индекса цикла - это проблема.
* Когда была создана эта проблема? Какое именно действие при разработке явилось ее источником? Это была проблема в требованиях? Проектировании системы? Коде? Тестировании?
* Когда проблема была выявлена? Может, она и не была сразу же устранена, но что нас интересует: сколько она существовала до того, как мы ее обнаружили?
* Каким образом была найдена эта проблема? Способ обнаружения можно внедрить в постоянно используемую практику.
* Можно ли было обнаружить ее раньше? Есть ли какой-либо процесс контроля качества, который мог бы ее выявить, будь он эффективнее?
* Сколько стоило устранение этой проблемы? Этот момент очень легко недооценить. Убедитесь, что Вы учли процесс диагностики проблемы и всю работу по ее устранению, которую Вам пришлось сделать, включая ре-дизайн, переписывание кода, ре-компиляцию, переработку тестов, повторное тестирование, повторный релиз, выпуск заплатки, формирование отчета по дефекту, отчет по статусу проекта и т.д. (не забудьте еще возможные затраты на исправление подпорченной репутации на рынке ПО).
* Какого рода была эта проблема? Когда у Вас огромное количество дефектов, то их категоризация облегчает анализ и обучение.

Когда Вы анализируете информацию о дефектах, то ищите те дефекты, которые обнаруживаются регулярно, и те, затраты на устранение которых высоки. Вот как раз таких дефектов и нужно избегать в будущем (или, по крайней мере, устранять их на более ранней стадии разработки), именно такая тактика гарантированно будет способствовать улучшению качества.

#### Атрибуты отчёта о дефекте

В зависимости от инструментального средства управления отчётами о дефектах внешний вид их записи может немного отличаться, могут быть добавлены или убраны отдельные поля, но концепция остаётся неизменной.

* Идентификатор (identifier) представляет собой уникальное значение, позволяющее однозначно отличить один отчёт о дефекте от другого и используемое во всевозможных ссылках. В общем случае идентификатор отчёта о дефекте может представлять собой просто уникальный номер, но (если позволяет инструментальное средство управления отчётами) может быть и куда сложнее: включать префиксы, суффиксы и иные осмысленные компоненты, позволяющие быстро определить суть дефекта и часть приложения (или требований), к которой он относится.
* Краткое описание (summary) должно в предельно лаконичной форме давать исчерпывающий ответ на вопросы «Что произошло?» ,«Где это произошло?», «При каких условиях это произошло?». Например: «Отсутствует логотип на странице приветствия, если пользователь является администратором». Что произошло? Отсутствует логотип. Где это произошло? На странице приветствия. При каких условиях это произошло? Если пользователь является администратором.
* Подробное описание (description) представляет в развёрнутом виде необходимую информацию о дефекте, а также (обязательно!) описание фактического результата, ожидаемого результата и ссылку на требование (если это возможно). В отличие от краткого описания, которое, как правило, является одним предложением, здесь можно и нужно давать подробную информацию. Если одна и та же проблема (вызванная одним источником) проявляется в нескольких местах приложения, можно в подробном описании перечислить эти места.
* Шаги по воспроизведению (steps to reproduce, STR) описывают действия, которые необходимо выполнить для воспроизведения дефекта. Это поле похоже на шаги тест-кейса, за исключением одного важного отличия: здесь действия прописываются максимально подробно, с указанием конкретных вводимых значений и самых мелких деталей, т.к. отсутствие этой информации в сложных случаях может привести к невозможности воспроизведения дефекта.
* Воспроизводимость (reproducibility) показывает, при каждом ли прохождении по шагам воспроизведения дефекта удаётся вызвать его проявление. Это поле принимает всего два значения: всегда (always) или иногда (sometimes). Можно сказать, что воспроизводимость «иногда» означает, что тестировщик не нашёл настоящую причину возникновения дефекта. Тестировщику нужно потратить много времени на то, чтобы удостовериться в наличии дефекта (т.к. однократный сбой в работе приложения мог быть вызван огромным количеством посторонних причин). Разработчику тоже нужно потратить время, чтобы добиться проявления дефекта и убедиться в его наличии. После внесения исправлений в приложение разработчик фактически должен полагаться только на свой профессионализм, т.к. даже многократное прохождение по шагам воспроизведения в таком случае не гарантирует, что дефект был исправлен (возможно, через ещё 10–20 повторений он бы проявился).
* Важность (severity) показывает степень ущерба, который наносится проекту существованием дефекта. В общем случае выделяют следующие градации важности:
  + критическая (critical) — существование дефекта приводит к масштабным последствиям катастрофического характера: потеря данных, раскрытие конфиденциальной информации, нарушение ключевой функциональности приложения и т.д.
  + высокая (major) — существование дефекта приносит ощутимые неудобства многим пользователям в рамках их типичной деятельности: недоступность вставки из буфера обмена, неработоспособность общепринятых клавиатурных комбинаций, необходимость перезапуска приложения при выполнении типичных сценариев работы.
  + средняя (medium) — существование дефекта слабо влияет на типичные сценарии работы пользователей, и/или существует обходной путь достижения цели, например: диалоговое окно не закрывается автоматически после нажатия кнопок «OK»/«Cancel», при распечатке нескольких документов подряд не сохраняется значение поля «Двусторонняя печать», перепутаны направления сортировок по некоему полю таблицы.
  + низкая (minor) — существование дефекта редко обнаруживается незначительным процентом пользователей и почти не влияет на их работу, например: опечатка в глубоко вложенном пункте меню настроек, некое окно сразу при отображении расположено неудобно (нужно перетянуть его в удобное место), неточно отображается время до завершения операции копирования файлов.
* Срочность (priority) показывает, как быстро дефект должен быть устранён. В общем случае выделяют следующие градации срочности:
  + наивысшая (ASAP, as soon as possible) срочность указывает на необходимость устранить дефект настолько быстро, насколько это возможно. В зависимости от контекста «настолько быстро, насколько возможно» может варьироваться от «в ближайшем билде» до единиц минут.
  + высокая (high) срочность означает, что дефект следует исправить вне очереди, т.к. его существование или уже объективно мешает работе, или начнёт создавать такие помехи в самом ближайшем будущем.
  + обычная (normal) срочность означает, что дефект следует исправить в порядке общей очерёдности. Такое значение срочности получает большинство дефектов.
  + низкая (low) срочность означает, что, в обозримом будущем, исправление данного дефекта не окажет существенного влияния на повышение качества продукта.
* Фактический результат (actual result) - результат, полученный после прохождения шагов к воспроизведению.
* Ожидаемый результат (expected result) - описывает ожидаемое поведение ПО после прохождения шагов к воспроизведению.
* Симптом (symptom) — позволяет классифицировать дефекты по их типичному проявлению. Не существует никакого общепринятого списка симптомов. Более того, далеко не в каждом инструментальном средстве управления отчётами о дефектах есть такое поле, а там, где оно есть, его можно настроить. В качестве примера есть следующие значения симптомов дефекта: 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 (localization issue), Нереализованная функциональность (missing feature), Проблема масштабируемости (scalability), Низкая производительность (low performance), Крах системы (system crash), Неожиданное поведение (unexpected behavior), Недружественное поведение (unfriendly behavior), Расхождение с требованиями (variance from specs), Предложение по улучшению (enhancement).
* Комментарий (comments, additional info) — может содержать любые полезные для понимания и исправления дефекта данные. Иными словами, сюда можно писать всё то, что нельзя писать в остальные поля.
* Приложения (attachments) — представляет собой не столько поле, сколько список прикреплённых к отчёту о дефекте приложений (копий экрана, вызывающих сбой файлов и т.д.)

#### Свойства качественных отчётов о дефектах

Отчёт о дефекте может оказаться некачественным (а следовательно, вероятность исправления дефекта понизится), если в нём нарушено одно из следующих свойств:

* Тщательное заполнение всех полей точной и корректной информацией. Нарушение этого свойства происходит по множеству причин: недостаточный опыт тестировщика, невнимательность, лень и т.д.
* Правильный технический язык. Это свойство в равной мере относится и к требованиям, и к тест-кейсам, и к отчётам о дефектах — к любой документации, а потому не будем повторяться.
* Специфичность описания шагов. Говоря о тест-кейсах, мы подчёркивали, что в их шагах стоит придерживаться золотой середины между специфичностью и общностью. В отчётах о дефектах предпочтение, как правило, отдаётся специфичности по очень простой причине: нехватка какой-то мелкой детали может привести к невозможности воспроизведения дефекта. Потому, если у вас есть хоть малейшее сомнение в том, важна ли какая-то деталь, считайте, что она важна.
* Отсутствие лишних действий и/или их длинных описаний. Чаще всего, это свойство подразумевает, что не нужно в шагах по воспроизведению дефекта долго и по пунктам расписывать то, что можно заменить одной фразой.
* Отсутствие дубликатов. Когда в проектной команде работает большое количество тестировщиков, то может возникнуть ситуация, при которой один и тот же дефект будет описан несколько раз разными людьми.
* Очевидность и понятность. Описывайте дефект так, чтобы у читающего Ваш отчёт не возникло ни малейшего сомнения в том, что это действительно дефект. Лучше всего это свойство достигается за счёт тщательного объяснения фактического и ожидаемого результата, а также указания ссылки на требование в поле «Подробное описание».
* Прослеживаемость. Из содержащейся в качественном отчёте о дефекте информации должно быть понятно, какую часть приложения, какие функции и какие требования затрагивает дефект. Лучше всего это свойство достигается правильным использованием возможностей инструментального средства управления отчётами о дефектах: указывайте в отчёте о дефекте компоненты приложения, ссылки на требования, тест-кейсы, смежные отчёты о дефектах (похожих дефектах; зависимых и зависящих от данного дефектах), расставляйте теги и т.д.
* Отдельные отчёты для каждого нового дефекта. Существует два незыблемых правила:
  + в каждом отчёте описывается ровно один дефект (если один и тот же дефект проявляется в нескольких местах, то эти проявления перечисляются в подробном описании).
  + при обнаружении нового дефекта создаётся новый отчёт. Нельзя для описания нового дефекта править старые отчёты, переводя их в состояние «открыт заново».
* Соответствие принятым шаблонам оформления и традициям. Как и в случае с тест-кейсами, с шаблонами оформления отчётов о дефектах проблем не возникает: они определены имеющимся образцом или экранной формой инструментального средства управления жизненным циклом отчётов о дефектах. Но, что касается традиций, которые могут различаться даже в разных командах в рамках одной компании, единственный совет: почитайте уже готовые отчёты о дефектах, перед тем как писать свои. Это может сэкономить вам много времени и сил.

#### Логика создания эффективных отчётов о дефектах

При создании отчёта о дефекте рекомендуется следовать следующему алгоритму:

1. Обнаружить дефект.
2. Понять суть проблемы.
3. Воспроизвести дефект.
4. Проверить наличие описания найденного вами дефекта в системе управления дефектами.
5. Сформулировать суть проблемы в виде «что сделали, что получили, что ожидали получить».
6. Заполнить поля отчёта, начиная с подробного описания.
7. После заполнения всех полей внимательно перечитать отчёт, исправить неточности и добавить подробности.
8. Ещё раз перечитать отчёт, т.к. в пункте 6 вы точно что-то упустили.

#### Типичные ошибки при написании отчётов о дефектах

Ошибки оформления и формулировок:

* Плохие краткие описания (summary). Формально, эта проблема относится к оформлению, но фактически она куда опаснее: чтение отчёта о дефекте и осознание сути проблемы начинается именно с краткого описания. Суть отчёта о дефекте: ответить на вопросы «что?», «где?», «при каких условиях?».
* Идентичные краткие и подробные описания (summary и description). Да, изредка бывают настолько простые дефекты, что для них достаточно одного краткого описания (например, «опечатка в имени пункта главного меню “File” (сейчас “Fille”)»), но, если дефект связан с неким более-менее сложным поведением приложения, стоит продумать как минимум три способа описания проблемы:
  + краткий для поля «краткое описание» (его лучше формулировать в самом конце размышлений);
  + подробный для поля «подробное описание» (поясняющий и расширяющий информацию из «краткого описания»);
  + ещё один краткий для последнего шага в шагах по воспроизведению дефекта.
* Отсутствие в подробном описании явного указания фактического результата, ожидаемого результата и ссылки на требование, если они важны, и их представляется возможным указать.
* Игнорирование кавычек, приводящее к искажению смысла. Как Вы поймёте такое краткое описание, как «запись исчезает при наведении мыши»? Какая-то запись исчезает при наведении мыши? А вот и нет. Оказывается, «поле “Запись” исчезает при наведении мыши». Даже если не дописать слово «поле», кавычки подскажут, что имеется в виду имя собственное, т.е. название некоего элемента.
* Общие проблемы с формулировками фраз на русском и английском языках.
* Лишние пункты в шагах воспроизведения.
* Копии экрана в виде «копий всего экрана целиком». Чаще всего нужно сделать копию какого-то конкретного окна приложения, а не всего экрана, тогда поможет Alt+PrintScreen. Даже если важно захватить больше, чем одно окно, практически любой графический редактор позволяет отрезать ненужную часть картинки.
* Копии экрана, на которых не отмечена проблема. Если обвести проблемную область красной линией, то это в разы повысит скорость и простоту понимания сути проблемы в большинстве случаев.
* Откладывание написания отчёта «на потом». Стремление сначала найти побольше дефектов, а уже потом их описывать приводит к тому, что какие-то важные детали (а иногда и сами дефекты!) забываются. Если «на потом» измеряется не минутами, а часами или даже днями, то проектная команда не получает важную информацию вовремя. Вывод простой: описывайте дефект сразу же, как только обнаружили его.
* Пунктуационные, орфографические, синтаксические и им подобные ошибки.

Логические ошибки:

* Выдуманные дефекты. Одной из наиболее обидных для тестировщика причин отклонения отчёта о дефекте является так называемое «описанное поведение не является дефектом» («not a bug»), когда по какой-то причине корректное поведение приложения описывается как ошибочное.
* Отнесение расширенных возможностей приложения к дефектам. Самым ярким примером этого случая является описание как дефекта того факта, что приложение запускается под операционными системами, не указанными явно в списке поддерживаемых. Лишь в очень редких случаях при разработке неких системных утилит и тому подобного ПО, крайне зависящего от версии ОС и потенциально способного «поломать» неподдерживаемую, это можно считать ошибкой (с точки зрения общего здравого смысла ,такое приложение действительно должно показывать предупреждение или даже сообщение об ошибке и завершать работу на неподдерживаемой ОС).
* Неверно указанные симптомы. Это не смертельно и всегда можно подправить, но, если изначально отчёты будут сгруппированы по симптомам, это создаёт множество раздражающих неудобств.
* Чрезмерно заниженные (или завышенные) важность и срочность. С этой бедой достаточно эффективно борются проведением общих собраний и пересмотром отчётов о дефектах силами всей команды (или хотя бы нескольких человек), но, если эти показатели занижены именно чрезмерно, есть высокая вероятность, что пройдёт очень много времени, прежде чем до такого отчёта просто дойдёт очередь на следующих собраниях по пересмотру.
* Концентрация на мелочах в ущерб главному. Здесь стоит упомянуть хрестоматийный пример, когда тестировщик нашёл проблему, приводящую к краху приложения с потерей пользовательских данных, но записал её как косметический дефект (в сообщении об ошибке, которое «перед смертью» показывало приложение, была опечатка). Всегда думайте о том, как произошедшая с приложением неприятность повлияет на пользователей, какие сложности они могут из-за этого испытать, насколько это для них важно, тогда шанс увидеть реальную проблему резко повышается.
* Техническая безграмотность. Представьте себе такое краткое описание (оно же идентично продублировано в подробном, т.е. это и есть всё описание дефекта): «Количество найденных файлов не соответствует реальной глубине вложенности каталога». А что должно? Это ведь почти то же самое, что «цвет кошки не соответствует её размеру».
* Указание в шагах воспроизведения неважной для воспроизведения ошибки информации. Стремление прописать всё максимально подробно иногда принимает нездоровую форму, когда в отчёт о дефекте начинает попадать чуть ли не информация о погоде за окном и курс национальной валюты.
* Отсутствие в шагах воспроизведения информации, важной для воспроизведения дефекта.
* Игнорирование «последовательных дефектов». Иногда один дефект является следствием другого (допустим, файл повреждается при передаче на сервер, а затем приложение некорректно обрабатывает этот повреждённый файл). Да, если файл будет передан без повреждений, то второй дефект может не проявиться. Но может и проявиться в другой ситуации, т.к. проблема никуда не исчезла: приложение некорректно обрабатывает повреждённые файлы. Потому стоит описать оба дефекта.

# Инструменты управления отчётами о дефектах

Инструментальных средств управления отчётами о дефектах (bug tracking system, defect management tool) очень много, к тому же ,многие компании разрабатывают свои внутренние средства решения этой задачи. Зачастую такие инструментальные средства являются частями инструментальных средств управления тестированием.

Общий набор функций багтрекинговых систем:

* создание отчётов о дефектах, управление их жизненным циклом с учётом контроля версий, прав доступа и разрешённых переходов из состояния в состояние;
* сбор, анализ и предоставление статистики в удобной для восприятия человеком форме;
* рассылка уведомлений, напоминаний и иных артефактов соответствующим сотрудникам;
* организация взаимосвязей между отчётами о дефектах, тест-кейсами, требованиями и анализ таких связей с возможностью формирования рекомендаций;
* подготовка информации для включения в отчёт о результатах тестирования;
* интеграция с системами управления проектами.

#### Наиболее популярные проекты:



#### Поля отчёта о дефекте в Jira

* Project (проект) позволяет указать, к какому проекту относится дефект.
* Issue type (тип записи/артефакта) позволяет указать, что именно представляет собой создаваемый артефакт. JIRA позволяет создавать не только отчёты о дефектах, но и множество других артефактов, типы которых можно настраивать.
* Improvement (предложение по улучшению) — предложение на улучшение компонента, функциональности.
* New feature (новая особенность) — описание новой функциональности, нового свойства, новой особенности продукта.
* Task (задание) — некое задание для выполнения тем или иным участником проектной команды.
* Custom issue (произвольный артефакт) — как правило, это значение при настройке JIRA удаляют, заменяя своими вариантами, или переименовывают в Issue.
* Summary (краткое описание) позволяет указать краткое описание дефекта.
* Priority (срочность) позволяет указать срочность исправления дефекта. По умолчанию JIRA предлагает следующие варианты:
  + Highest (самая высокая срочность).
  + High (высокая срочность).
  + Medium (обычная срочность).
  + Low (низкая срочность).
  + Lowest (самая низкая срочность).

Обратите внимание: по умолчанию поля severity (важность) нет. Но его можно добавить.

* Components (компоненты) содержит перечень компонентов приложения, затронутых дефектом (но иногда здесь перечисляют симптомы дефектов).
* Affected versions (затронутые версии) содержит перечень версий продукта, в которых проявляется дефект.
* Environment (окружение) содержит описание аппаратной и программной конфигурации, в которой проявляется дефект.
* Description (подробное описание) позволяет указать подробное описание дефекта.
* Original estimate (начальная оценка времени исправления) позволяет указать начальную оценку того, сколько времени займёт устранение дефекта.
* Remaining estimate (расчётное остаточное время исправления) показывает, сколько времени осталось от начальной оценки.
* Story points (оценочные единицы) позволяет указать сложность дефекта (или иного артефакта) в специальных оценочных единицах, принятых в гибких методологиях управления проектами.
* Labels (метки) содержит метки (теги, ключевые слова), по которым можно группировать и классифицировать дефекты и иные артефакты.
* Epic/Theme (история/область) содержит перечень высокоуровневых меток, описывающих относящиеся к дефекту крупные области требований, крупные модули приложения, крупные части предметной области, объёмные пользовательские истории и т.д.
* External issue id (идентификатор внешнего артефакта) позволяет связать отчёт о дефекте или иной артефакт с внешним документом.
* Epic link (ссылка на историю/область) содержит ссылку на историю/область, наиболее близко относящуюся к дефекту.
* Has a story/s (истории) содержит ссылки и/или описание пользовательских историй, связанных с дефектом (как правило, здесь приводятся ссылки на внешние документы).
* Tester (тестировщик) содержит имя автора описания дефекта.
* Additional information (дополнительная информация) содержит полезную дополнительную информацию о дефекте.
* Sprint (спринт) содержит номер спринта (2–4-недельной итерации разработки проекта в терминологии гибких методологий управления проектами), во время которого был обнаружен дефект.

Многие дополнительные поля и возможности становятся доступными при других операциях с дефектами (просмотром или редактированием созданного дефекта, просмотре отчётов и т.д.)