Лекция №14 Тест план

### **Введение**

Документацию часто разрабатывают в рамках мероприятий по управлению тестированием. Конкретные виды документов по управлению тестированием и их рамки, как правило, весьма различны. Ниже представлены наиболее распространенные виды документов по управлению тестированием, найденные в организациях и на проектах:

* Политика тестирования – описывает назначение и корпоративные цели тестирования.
* Стратегия тестирования – описывает корпоративные методы тестирования, не зависящие от проекта.
* Главный план тестирования (или план тестирования проекта) – описывает применение стратегии тестирования в конкретном проекте.
* Уровневый план тестирования – описывает конкретные активности, которые необходимо выполнить в рамках конкретного уровня тестирования.

Физическая организация этих типов документов может отличаться в зависимости от специфики. В некоторых организациях и на некоторых проектах они могут быть объединены в один документ; в других – они могут находиться в отдельных документах; в некоторых их содержание может проявиться в качестве неписаных, интуитивно понятных или традиционных методологий тестирования. Большие и формальные организации и проекты имеют тенденцию реализовывать эти типы документов как формализованные рабочие продукты, в то время как небольшие и менее формальные организации и проекты имеют тенденцию формализовывать рабочие продукты в меньшем количестве. Эта программа обучения описывает каждый из этих типов документов отдельно, хотя на практике контекст организации и проекта определяет правильное использование каждого типа.

### **Политика тестирования**

Политика тестирования описывает, почему организация занимается тестированием. Она определяет общие цели тестирования, которые организация хочет достигнуть. Эта политика может быть разработана старшим руководителем тестирования организации в сотрудничестве со старшими руководителями тестирования из групп заинтересованных сторон.

В некоторых случаях политика тестирования будет дополнять или будет компонентом более широкой политики качества. Эта политика качества описывает общую ценность и цели, связанные с управлением качеством.

Если формализованная политика тестирования существует, она может быть коротким высокоуровневым документом, который:

* Обобщает ценности, которую организация получает от тестирования.
* Определяет цели тестирования, такие как укрепление уверенности в ПО, обнаружение дефектов в ПО и снижение уровня рисков качества.
* Описывает, как оценивается эффективность и действенность тестирования при выполнении этих целей.
* Обрисовывает в общих чертах стандартный процесс тестирования, используя в качестве основы базовый процесс тестирования ISTQB.
* Определяет, как организация может улучшить свой процесс тестирования.

Политика тестирования должна обращаться к активностям по тестированию для новых разработок, а также для поддержки. Она может также ссылаться на внутренние или внешние стандарты тестирования рабочих продуктов и корпоративную терминологию.

Начало формы

Стратегия тестирования

Конец формы

### **Стратегия тестирования**

Стратегия тестирования описывает корпоративную методологию тестирования. Она включает способы, которые используют при тестировании для управления рисками продукта и проекта, разделяющие тестирование на уровни и высокоуровневые активности, связанные с тестированием. (Одна и та же организация может использовать различные стратегии для различных ситуаций, таких как различные жизненные циклы ПО, различные уровни риска или различные нормативные требования.) Стратегия тестирования и описанные в ней процессы и активности должны быть связаны с политикой тестирования. Это должно обеспечивать универсальные критерии входа и выхода для организации одной или нескольких программ.

Как упоминалось выше, стратегии тестирования описывают общие методологии тестирования, которые, как правило, включают:

* Стратегии, основанные на требованиях, такие как тестирование на основе рисков, где команда тестировщиков анализирует базис тестирования для определения покрываемых тестовых условий. Например, при тестировании на основе требований анализ тестирования выделяет тестовые условия из требований, после этого тесты проектируются и реализуются для покрытия этих условий. Результаты тестирования сообщаются в терминах статуса требований (например, требование протестировано и тесты пройдены, требование протестировано и тесты не пройдены, требование еще не полностью протестировано, тестирование требования заблокировано и т.д.).
* Стратегии, основанные на модели, такие как эксплуатационное профилирование, где команда тестировщиков разрабатывает модель (основанную на актуальных или предполагаемых ситуациях) в окружении, в котором система работает, входные данные и условия, которым система подвергается, и то, как система должна вести себя. Например, при основанном на моделях тестировании производительности быстро развивающегося приложения для мобильных устройств можно разрабатывать модели входящего и исходящего сетевого трафика, активных и неактивных пользователей и получать обработку нагрузки, основанной на текущем использовании и развитии проекта в течение времени. Кроме того, модели могли бы быть разработаны с учетом текущей производительности аппаратного обеспечения, ПО, объема данных, сети и инфраструктуры. Модели также могут быть разработаны для идеального, ожидаемого и минимального уровня пропускной способности, времени ответа, распределения ресурсов.
* Методические стратегии, основанные на характеристиках качества, где команда тестировщиков руководствуется предопределенным набором тестовых условий, таких как стандарт качества (например, ISO 25000 [ISO2500], который сменил ISO 9126 [ISO9126]), перечень или набор обобщенных логических тестовых условий, которые могут относиться к определенной области, приложению или типу тестирования (например, [тестирование безопасности](https://dist.belstu.by/mod/assign/view.php?id=57440)), и использует этот набор тестовых условий от одной итерации к другой или от одного релиза к другому. Например, в эксплуатационном тестировании простого и стабильного веб-сайта по электронной торговле тестировщики могут использовать список, который определяет ключевые функции, атрибуты и связи для каждой страницы. Тестировщики будут проверять соответствующие элементы этого списка каждый раз при внесении изменений в сайт.
* Стратегии, совместимые с процессом или стандартом, такие как стандарт управления по санитарному надзору за качеством пищевых продуктов и медикаментов США, где команда тестировщиков следует набору процессов, которые определены комитетом по стандартам или иным экспертным советом, где процессы обращаются к документации, прошедшей идентификацию и использующую базис тестирования, тестовые предсказатели и организацию команды по тестированию. Например, в проектах, следующих технологии менеджмента Scrum Agile, в каждой итерации тестировщики анализируют пользовательские истории, которые описывают определенные функции, оценивают объем тестирования каждой функции как часть процесса планирования итерации, определяют тестовые условия для каждой пользовательской истории (их часто называют критериями приемки), выполняют тесты, которые покрывают эти условия, и отчитываются по статусу каждой пользовательской истории (не протестирована, не пройдена или пройдена) во время прохождения тестов.
* Реактивные стратегии, такие как использование атак на основе ошибок, в которой команда тестировщиков ожидает проектирование и реализацию тестов до получения ПО, реагируя на поведение тестируемой системы. Например, при исследовательском тестировании приложения на основе меню можно разработать концепции тестирования, соответствующие функциям, пунктам меню, интерфейсам. Каждому тестировщику дают концепцию тестирования, которую они затем используют для структурирования сессий по исследовательскому тестированию. Тестировщики регулярно отчитываются о результатах сессий тестирования перед руководителем тестирования, который может пересмотреть концепции на основании полученных данных.
* Консультативные стратегии, такие как тестирование, управляемое пользователями, в котором команда тестировщиков основывается на входных данных от одной или нескольких ключевых заинтересованных сторон для определения покрытия тестовых условий. Например, при тестировании совместимости веб-приложения, выполняемого сторонними организациями, компания может предоставлять стороннему поставщику услуг по тестированию список приоритетов версий браузера, антивредоносного ПО, операционных систем, типов соединения и другие конфигурационные параметры, Поставщик услуг по тестированию может затем использовать такие методы, как попарное тестирование (для опций высокого приоритета) и классы эквивалентности (для опций низкого приоритета) для создания тестов.
* Стратегии анти-регрессионного тестирования, такие как широкая автоматизация, где команда тестировщиков использует различные методы для управления риском регрессии, особенно функциональной и/или нефункциональной регрессии, автоматизацией регрессионного тестирования на одном или нескольких уровнях. Например, в регрессионном тестировании веб-приложения тестировщики могут использовать инструмент автоматизации тестирования, основанный на графическом интерфейсе, для автоматизации стандартных и исключительных сценариев использования приложения. Эти тесты затем выполняются каждый раз при модификации приложения.

Различные стратегии могут комбинироваться. Определенные стратегии должны соответствовать потребностям и возможностям организации, и организации могут адаптировать стратегии для соответствия особенностям операций и проектов.

Стратегии тестирования могут описывать выполняемые [уровни тестирования](https://dist.belstu.by/mod/assign/view.php?id=57433). В таких сценариях это может давать управление критериями входа и выхода для каждого уровня и взаимоотношением между уровнями (например, разделение целей тестового покрытия).

Стратегия тестирования может также описывать следующее:

* процедуры интеграции;
* методы разработки спецификации тестов;
* независимость тестирования (которое может отличаться для разных уровней);
* обязательные и необязательные стандарты;
* тестовое окружение;
* автоматизацию тестирования;
* инструменты тестирования;
* повторное использование ПО и тестов рабочих продуктов;
* подтверждающее и регрессионное тестирование;
* контроль тестирования и отчетность;
* тестовые измерения и метрики;
* управление дефектами;
* подход к управлению конфигурацией тестового обеспечения;
* роли и ответственности.

Различные стратегии тестирования могут быть нужны для краткосрочных и длительных проектов. Различные стратегии тестирования подходят для различных организаций и проектов. Например, в случае, когда затрагиваемая безопасность или надежность приложения критична, очень интенсивная стратегия может быть более подходящей, чем другие сценарии. Кроме того, стратегия тестирования отличается в зависимости от различных моделей разработки.

### **Главный план тестирования**

Главный план тестирования охватывает все работы по тестированию, выполняемые в конкретном проекте, включающие отдельные уровни, которые будут выполнены, связи между уровнями, и соответствие уровней тестирования и активностей по разработке. Главный план тестирования должен решать, как тестировщики могут реализовать стратегию тестирования в этом проекте (т.е. подход к тестированию). Главный план тестирования должен соответствовать политике и стратегии тестирования, и, в конкретных областях, где их нет, он должен объяснять те отклонения и исключения, включая любое потенциальное влияние, возникающее вследствие отклонений. Например, если в некой корпоративной стратегии тестирования заложено проведение одного полного регрессионного тестирования системы до выпуска релиза, но в текущем проекте не будет регрессионного тестирования, то план тестирования должен объяснить, почему оно не было запланировано и что должно быть сделано для снижения ожидаемых рисков в отличие от обычной стратегии. План тестирования должен также включать многие другие эффекты, которые ожидаются от этого различия. Например, игнорирование регрессионного тестирования может потребовать планирования релиза обновленной эксплуатационной версии через месяц после первоначального выпуска проекта. Главный план тестирования должен дополнять план проекта или руководство по работе с продуктом, в котором могут быть описаны трудозатраты по тестированию, которое является частью большего проекта или эксплуатации.

Хотя особенность содержания и структуры главного плана тестирования различаются в зависимости от организации, стандартов документации, формальной стороны проекта, типичные разделы главного плана тестирования содержат:

* Тестируемые и нетестируемые элементы.
* Тестируемые и нетестируемые характеристики качества.
* Программу тестирования и бюджет (которые должна быть увязаны с бюджетом проекта или эксплуатации).
* Циклы выполнения тестов и их связь с планом выпуска ПО.
* Взаимоотношение и результаты тестирования других специалистов или отделов.
* Определение, какие элементы тестирования входят и не входят в состав каждого описанного уровня.
* Конкретные критерии входа, продолжения (приостановки/возобновления) и выхода для каждого уровня и взаимоотношений между уровнями.
* Риски проекта по тестированию.
* Общее управление тестированием.
* Ответственность за выполнение каждого уровня тестирования.
* Входные и выходные данные для каждого уровня тестирования.

На меньших проектах или операциях, где формализован только один уровень тестирования, главный план тестирования и тестовый план этого формализованного уровня тестирования часто объединяются в одном документе. Например, если тесты системы формализованы только для одного уровня с обычным компонентным и интеграционным тестированием, выполняемым разработчиками, и обычным приемочным тестированием, выполняемым клиентами, как часть процесса бета-тестирования, тогда план тестирования системы может включать элементы, упоминаемые в этом разделе.

Кроме того, тестирование, как правило, зависит от других действий в рамках проекта. Если эти мероприятия недостаточно документированы, особенно в части влияния и связей с тестированием, то те темы, которые связанны с активностями, могут быть охвачены в главном плане тестирования (или в соответствующем плане тестирования уровня). Например, если процесс управления конфигурацией не документирован, то план тестирования должен определить, как объекты тестирования будут переданы команде тестировщиков.

### **Уровневый план тестирования**

Уровневые планы тестирования описывают определенные активности, выполняемые в рамках каждого уровня тестирования, или, в некоторых случаях, типа тестирования. Уровневый план тестирования распространяется, если необходимо, на главный план тестирования для конкретного уровня или типа тестирования, которые документированы. Они содержат график, задачи и вехи, не обязательно включенные в главный план тестирования. Кроме того, степень различия стандартов и применяемых шаблонов к спецификации тестов на различных уровнях могут быть покрыты в уровневом плане тестирования.

На менее формальных проектах или операциях индивидуальный план тестирования является единственным зафиксированным документом по управлению тестированием. В таких ситуациях некоторые из информационных элементов, упомянутых выше в этом разделе, могут быть включены в этот план тестирования.

Для гибких (Agile) проектов планы тестирования спринта или итерации могут занять место уровневых планов тестирования.

### **Управление рисками проекта**

Важная часть, присущая планированию, включает работу с рисками проекта. Когда риски проекта определены, они должны быть переданы менеджеру проекта для принятия мер. Снижение таких рисков не всегда находится в компетенции организации по тестированию. Однако руководитель тестирования может и должен минимизировать такие риски проекта, как:

* тестовое окружение и готовность оборудования;
* готовность и уровень подготовки тестировщиков;
* недостаток стандартов, регламентов и методов при тестировании.

Подходы к управлению рисками проекта включают предыдущую подготовку тестового обеспечения, предварительное тестирование тестового окружения, предварительное тестирование ранних версий продукта, применение более строгих критериев входа к тестированию, обеспечение тестируемости требований, участие в рецензировании ранних рабочих продуктов проекта, участие в управлении изменениями и мониторинг хода выполнения проекта и качества.

Когда риск проекта был выявлен и проанализирован, существуют четыре главные функции управления этим риском:

1. Снизить риск при помощи предупредительных мер для того, чтобы снизить вероятность и/или влияние.
2. Сделать планы непредвиденных обстоятельств снижения влияния, если риск становится реальным.
3. Передать риск другому участнику для контроля.
4. Игнорировать или принять риск.

Выбор лучшего решения зависит от выгод и преимуществ созданного решения, а также от стоимости, и возможно, от дополнительных рисков, связанных с решением. Когда план непредвиденных обстоятельств определен для проектного риска, лучшие практики определяют триггер (который может решить, когда и как должен быть вызван план непредвиденных обстоятельств), и владельца (кто должен выполнять план непредвиденных обстоятельств).

## Лекция №15 Отчеты об ошибках

### **Введение**

**Отчет об ошибке** - это документ, описывающий и приоритизирующий обнаруженный дефект, а также содействующий его устранению.

**Дефект**- это отклонение фактического результата от ожиданий наблюдателя, сформированных на основе требований, спецификаций, иной документации или опыта и здравого смысла.

**Отчеты от ошибках оформляются с целью:**

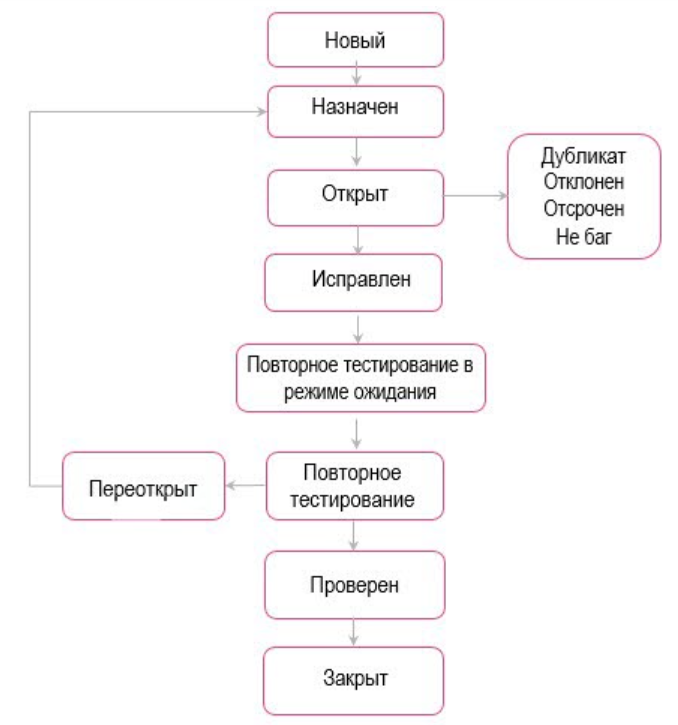
* Предоставить информацию о проблеме:
  + уведомить проектную команду и иных заинтересованных лиц о наличии проблемы, описать суть проблемы;
* Приоритизировать проблему:
  + определить степень опасности и желаемые сроки её устранения;
* Содействовать устранению проблемы:
  + предоставить необходимые подробности для понимания сути случившегося, а также анализ причин возникновения проблемы и рекомендации по исправлению ситуации.

### **Жизненнный цикл дефекта**

**Жизненный цикл дефекта** (Defect Lifecycle) – это последовательность этапов, которые проходит дефект на своём пути с момента его создания до окончательного закрытия. Для простоты восприятия изображается в виде схемы с возможными статусами и действиями, которые приводят к смене этих статусов.

Любой этап работы с дефектом обозначается атрибутом «Статус». Статус дефекта показывает, кто на данный момент работает с дефектом и что следует делать с ним. После того, как работу с багом закончил один из участников проекта, он меняет статус дефекта и «перенаправляет» дефект на того, кто должен продолжить работу.

Зачастую жизненный цикл бага связан с выбранной системой баг-трекинга, в которой по умолчанию могут быть настроены статусы и переходы. Но также некоторые из систем управления дефектами позволяют скорректировать жизненный цикл бага под конкретный проект в случае такой необходимости. Система может предоставлять администратору возможность настроить, какие пользователи могут просматривать и редактировать ошибки в зависимости от их состояния, переводить их в другое состояние или удалять.

**овый (New)**‒ отчет о дефекте заводится в баг-трекинговую систему в первый раз.

**Назначен (Assigned) ‒** отчет о дефекте назначается на соответствующего разработчика.

**Открыт (Open) ‒**разработчик берет отчет о дефекте в работу для анализа и исправления.

**Исправлен (Fixed) ‒**разработчик сделал необходимые изменения в коде и проверил эти изменения сам. Отчет о дефекте с этим статусом возвращается обратно тестировщику.

**Повторное тестирование в режиме ожидания ‒** после исправления дефекта разработчик предоставил конкретный код для повторного тестирования тестировщиком. Тестирование находится на рассмотрении у тестировщика.

**Повторное тестирование (Re-testing) ‒** на этой стадии тестировщик выполняет повторное тестирование измененного кода, который был предоставлен разработчиком, для проверки, исправлен ли дефект или нет.

**Проверен (Verified) ‒**если дефект не воспроизводится, тестировщик подтверждает, что этот дефект исправлен.

**Переоткрыт (Reopened) ‒** если дефект все же воспроизводится, даже после его исправления разработчиком, тестировщик переоткрывает его и назначает на разработчика. Этот дефект проходит через жизненный цикл дефекта еще раз.

**Закрыт (Closed) ‒** если тестировщик уверен, что дефект больше не воспроизводится, то он его закрывает. Этот статус означает, что дефект исправлен, протестирован и одобрен.

**Дубликат (Duplicate) ‒** если дефект повторяется дважды или есть два бага, которые являются следствием одной причины, то одному из них присваивается данный статус.

**Отклонен (Rejected) ‒** если разработчик считает, что этот дефект не является обоснованным или веским, и дефект не будет рассматриваться для исправления или реализации, он его отклоняет.

**Отсрочен (Deferred) ‒** ожидается, что дефект, которому присвоили такой статус, будет исправлен в следующих версиях. Причин для присвоения этого статуса может быть несколько: приоритет дефекта низкий, нехватка времени, данный дефект не повлечет больших сбоев в программном продукте.

**Не баг (Not a bug) ‒**этот статус присваивается, если в функционал приложения не будет внесено никаких изменений. Например, если заказчик просит изменить цвет или размер кнопок, или текста – это не дефект, а просто изменения в дизайне приложения.

Давайте рассмотрим жизненный цикл дефекта более подробно.

1. Тестировщик находит дефект.
2. Тестировщик оформляет отчет о дефекте в баг-трекинговую систему (статус «Новый» (New)) и назначает на разработчика (статус «Назначен» (Assigned)).
3. Разработчик проверяет дефект, воспроизводится он или нет, и присваивает ему один из статусов:
   * «Дубликат» (Duplicate) – похожий дефект уже есть в баг-трекинговой системе;
   * «Отклонен» (Rejected) – дефект не является веским;
   * «Отсрочен» (Deferred) – исправление дефекта можно перенести в следующие версии программного продукта;
   * «Не баг» (Not a bug) – в функционал программного продукта не будет внесено никаких изменений;
   * «Открыт» (Open) – разработчик взял дефект в работу;
   * «Исправлен» (Fixed) – разработчик внес изменения в код и проверил их.
4. Тестировщик проводит повторное тестирование дефекта (статус «Повторное тестирование» (Re-testing)).
5. Если дефект не воспроизводится, тестировщик закрывает его (статусы «Проверен» (Verified), «Закрыт» (Closed)).
6. Если дефект воспроизводится, тестировщик возвращает его разработчику на исправление (статусы «Переоткрыт» (Reopened), «Назначен» (Assigned)) и такой дефект проходит этот жизненный цикл еще раз.

Исходя из информации о статусах дефектов в баг-трекинговой системе можно строить отчеты, которые позволяют оценить как работу разработчика, так и тестировщика. Эта информация может быть полезна для сбора и анализа ряда метрик, позволяющих оценить точки схождения дефектов и в целом качество выпускаемых продукто

### **Атрибуты отчета об ошибке**

* **Идентификатор (ID)** - обязательный атрибут, должен быть уникальным среди всей документации на проекте;
* **Краткое описание (Summary, Title)**- обязательное поле, которое заполняется по правилу WWW (Where, What, When):
  + Where - где обнаружен дефект;
  + What - что происходит с системой;
  + When - при каких условиях это происходит.
  + Пример: *Поиск: 500 ошибка сервера отображется вместо результатов поиска, когда пользователь использует спецсимволы в запросе.*
* **Подробное описание (Description)** - обязательное поле, состоит из трех частей:
  + Шаги воспроизведения (Steps to reproduce);
  + Актуальный результат (Actual result);
  + Ожидаемый результат (Expected result).
* **Воспроизводимость (Repeatability)**- не обязательное поле, как правило, может быть "постоянной", "периодической" или "редкой";
* **Важность (Severity)**- обязательное поле, показывает степень воздействия дефекта на работу тестируемого модуля или приложения. Бывает нескольких видов:
  + **Blocker**– баг блокирует тестирование компонента.
  + **Critical**- баг, когда не выполняется существенная для компонента операция, и нет обходных путей, чтобы эту операцию выполнить.
  + **Major**– баг, когда не выполняется (или выполняется неправильно) операция, но есть обходной путь, чтобы выполнить эту операцию.
  + **Minor**– несущественный функциональный баг, который не влияет на выполнение основных операций компонента (например, дефект в сортировке элементов в списке).
  + **Trivial**– UI дефекты в приложении.
* **Срочность (Priority)** - обязательное поле, это порядок, в котором дефект должен быть исправлен. Заполняет, как правило, проектный менеджер. Можно выделить 4 основных классов:
  + **Critical**(исправить немедленно);
  + **High**(исправить как можно быстрее);
  + **Medium**(с исправлением можно подождать);
  + **Low**(будет исправлен, когда все более приоритетные исправлены).
* **Возможность обойти (Workaround)** - необязательное поле, используется для описания обходного пути, если он имеется;
* **Комментарий (Comment, Note)** - необязательное поле, используется для различного рода заметок и дополнительных материалов, которые помогут исправить дефект;
* **Среда для воспроизведения бага (Environment)** - браузер, или платформа (мобильная или десктопная), где воспроизводится баг. Информация должна быть максимально подробной, с указанием версий, и другой необходимой информации;
* **Компонент (Component)** - необязательное поле, сюда записывают название компонента/модуля, в котором был найден дефект. Используется для сбора статистики по количеству дефектов в том или ином компоненте;
* **Тег (Tag, Label)** - необязательное поле, но может быть использовано с целью быстрого поиска дефектов в багтрэкинговой системе;
* **Ответственный (Assignee)** - обязательное поле, сотрудник, который будет ответственным за исправление дефекта. Выбирается из списка в багтрэкинговой системе;
* **Приложение к багу (Attachment)**- необязательное поле, но очень помогает при исследовании дефекта. Сюда прикладывают снимки с экрана, видеозаписи с экрана, логи сервера и другие необходимые данные, которые помогут локализовать проблему;
* **Срок исправления бага (Due Date)** - необязательно поле, дата, к которой ожидается исправление дефекта. Выставляется сотрудником, ответственным за исправление;
* **Версия билда, где был обнаружен баг (Affects Version)**- обязательное поле;
* **Версия, в которой должен быть исправлен баг (Fix Version)** - необязательное поле;
* **Создатель отчета об ошибке (Reporter)** - обязательное поле, выставляется автоматически багтрэкинговой системой;
* **Оценка трудозатрат на исправление бага (Original Estimate)** - необзательное поле.