**Отчётность**

**Отчётность** - сбор и распространение информации о результатах работы (включая текущий статус, оценку прогресса и прогноз развития ситуации).

**К высокоуровневым задачам отчётности относятся:**

* Сбор, агрегация и предоставление в удобной для восприятия форме объективной информации о результатах работы.
* Формирование оценки текущего статуса и прогресса (в сравнении с планом).
* Обозначение существующих и возможных проблем (если такие есть).
* Формирование прогноза развития ситуации и фиксация рекомендаций по устранению проблем и повышению эффективности работы.

**Отчёт о результатах тестирования** - документ, обобщающий результаты работ по тестированию и содержащий информацию, достаточную для соотнесения текущей ситуации с тест-планом и принятия необходимых управленческих решений.

**К низкоуровневым задачам отчётности в тестировании относятся:**

* Оценка объёма и качества выполненных работ.
* Сравнение текущего прогресса с тест-планом (в том числе с помощью анализа значений метрик).
* Описание имеющихся сложностей и формирование рекомендаций по их устранению.
* Предоставление лицам, заинтересованным в проекте, полной и объективной информации о текущем состоянии качества проекта, выраженной в конкретных фактах и числах.

**Качественный отчёт** о результатах тестирования обладает многими свойствами качественных требований, а также расширяет их набор следующими пунктами:

* информативность (в идеале, после прочтения отчёта не должно оставаться никаких открытых вопросов о том, что происходит с проектом в контексте качества);
* точность и объективность (ни при каких условиях в отчёте не допускается искажение фактов, а личные мнения должны быть подкреплены твёрдыми обоснованиями).

Отчёт о результатах тестирования создаётся по заранее оговорённому расписанию (зависящему от модели управления проектом) при участии большинства представителей проектной команды, задействованных в обеспечении качества.  
Большое количество фактических данных для отчёта может быть легко извлечено в удобной форме из системы управления проектом. Ответственным за создание отчёта, как правило, является ведущий тестировщик («тест-лид»). При необходимости, отчёт может обсуждаться на небольших собраниях.

**Отчёт о результатах тестирования предоставляется:**

* менеджеру проекта — как источник информации о текущей ситуации и основа для принятия управленческих решений;
* руководителю команды разработчиков («дев-лиду») — как дополнительный объективный взгляд на происходящее на проекте;
* руководителю команды тестировщиков («тест-лиду») — как способ структурировать собственные мысли и собрать необходимый материал для обращения к менеджеру проекта по насущным вопросам, если в этом есть необходимость;
* заказчику — как наиболее объективный источник информации о том, что происходит на проекте, за который он платит свои деньги.

**Отчёт о результатах тестирования включает следующие разделы:**

* **Краткое описание.** В предельно краткой форме отражает основные достижения, проблемы, выводы и рекомендации. В идеальном случае, прочтения краткого описания может быть достаточно для формирования полноценного представления о происходящем, что избавит от необходимости читать весь отчёт (это важно, т.к. отчёт о результатах тестирования может попадать в руки очень занятым людям).
* **Команда тестировщиков.** Список участников проектной команды, задействованных в обеспечении качества, с указанием их должностей и ролей в подотчётный период.
* **Описание процесса тестирования.**Последовательное описание того, какие работы были выполнены за подотчётный период.
* **Расписание.** Детализированное расписание работы команды тестировщиков и/или личные расписания участников команды.
* **Статистика по новым дефектам.** Таблица, в которой представлены данные по обнаруженным за подотчётный период дефектам (с классификацией по стадии жизненного цикла и важности).
* **Список новых дефектов.** Список обнаруженных за подотчётный период дефектов с их краткими описаниями и важностью.
* **Статистика по всем дефектам.** Таблица, в которой представлены данные по обнаруженным за всё время существования проекта дефектам (с классификацией по стадии жизненного цикла и важности). Как правило, в этот же раздел добавляется график, отражающий такие статистические данные.
* **Рекомендации.**Обоснованные выводы и рекомендации по принятию тех или иных управленческих решений (изменению тест-плана, запросу или освобождению ресурсов и т.д.) Здесь этой информации можно отвести больше места, чем в кратком описании (summary), сделав акцент именно на том, что и почему рекомендуется сделать в имеющейся ситуации.
* **Приложения.** Фактические данные (как правило, значения метрик и графическое представление их изменения во времени).

**Логика построения отчёта о результатах тестирования**

Для того, чтобы отчёт о результатах тестирования был действительно полезным, при его создании следует постоянно помнить об универсальной логике отчётности:

1. Выводы строятся на основе целей (которые были отражены в плане).
2. Выводы дополняются рекомендациями.
3. Выводы, так же, как и рекомендации, строго обосновываются.
4. Обоснование опирается на объективные факты.

**Выводы должны быть:**

* Краткими.
* Информативными.
* Полезными для читающего отчёт.

**Рекомендации должны быть:**

* Краткими.
* Реально выполнимыми.
* Дающими как понимание того, что надо сделать.

**Definition of Done**

**Вы это уже сделали?**

Что отвечать на такой вопрос рядовому сотруднику? Варианта два, но и два пути, которые данному работнику могут и не понравиться. Если он отвечает «да», то это означает, что он может взять еще работу на исполнение, а потом окажется, что ему нужно доделать старую. Также если он отвечает «нет», то его могут посчитать медлительным и он тормозит процесс.

Такой вопрос обычно задается бесчисленное количество раз при разработке какого-либо программного продукта, да и, в целом, во время рабочего процесса. В этих моментах, как и в любых других, должна быть оптимизация способная минимизировать или исключить полностью негативные стороны этого процесса. Вообще, понимание выражения "**Definition of Done"**в полной мере понятна людям, знакомыми с философией Scrum. Определенно сделанная задача – это та задача, которая не нуждается в доработках, но тут встает законный вопрос: "А как оценить, что задача действительно выполнена?"

Как может показаться изначально, вопрос так себе. Ну что значит "как оценить?" Выполнена - значит, выполнена, не выполнена - значит ,не выполнена.

Давайте посмотрим банальный пример про разработку чего-нибудь, например, части программы. Скажем, мы написали какой-то функционал, и вроде бы всё, однако, мы понимаем что наш функционал может иметь баги (ошибки), которые, например, сейчас не могут быть проверены, так как не готовы другие модули, позволяющие протестировать. Получается, ставя напротив этой задачи «Выполнено», мы немного лукавим, так как нам к ней придется возвращаться. Даже если есть модули для тестирования, необходимо ли тестировать? Может, следует поставить «Выполнено» только после результатов code review? Как мы видим, у нас очень много вопросов, которые мешают нам правильно построить работу. Из-за нашей неопределенности вытекают очень большие проблемы. Мало того, что мы сами можем не понять ,сделали ли мы до конца, а уж другие члены группы точно не смогут разобраться. Как Вы уже, наверное, догадались, **Definition of Done**и призвана исправить это и не дать нам повода беспокоиться!

**Definition of Done на страже нашего спокойствия**

На самом деле, на страже общего спокойствия. Мы действительно можем не понять до конца степень законченности нашей задачи, однако оповестить команду, что же мы все-таки сделали, обязаны. **Definition of Done,** как и всё в Scrum, должно быть лаконично, поэтому зачастую отводится для этого одно предложение, однако это не единственный вариант.

**Для примера приведем несколько выполненных задач с использованием Definition of Done:**

* Done = функционал оплаты реализован, проведено тестирование с тестировщиком Алексеем
* Done = разработан документ по спецификации, проведено обсуждение с клиентами
* Done = модуль авторизации разработан полностью, протестирован, продемонстрирован на Sprint Review Meeting
* Done = модуль полностью реализован и выгружен для использования

Как видно, любое описание начинается с “done=”, что дает понять, на что обращать внимание. Вообще, в листе принято писать такие результаты, которые можно проверить. Нет смысла описывать мысли, ведь, скажем, «done = придуман внешний вид интерфейса корзины» звучит странно и никак это проверить нельзя.

Желательно, изначально разработать список правил, которые будут описывать Definition of Done, чтобы все в команде писали в одном стиле. Это приведет к более быстрому пониманию того, что хотел передать коллега.

Еще одним из знаменитых способов записи **Definition of Done**— является простой список.

В заключении, хочется сказать, что не стоит пренебрегать Definition of Done, ведь это приводит не только к осознанию того, что было сделано, но и к тому, как это нужно сделать в будущем. Благодаря использованию Definition of Done все будут стараться делать задачи более четкими и конкретными, чтобы потом гордо сказать: «Точно сделано!». Помимо этого, набор рейтинга в Velocity будет гораздо выше.

В большинстве своем, мы привыкли к графикам идущими вверх, что означает положительную динамику, однако они могут идти и вниз и также показывать положительную динамику. Одним из таких ярких примеров является Диаграмма сгорания задач (Burndown chart). Само сочетание "Burn Down" дословно переводится как «гореть вниз» и это действительно так. Данный график является основным средством для отслеживания выполненных задач в спринте или во всем проекте. Хотя, по сути, он может использоваться как угодно, но мы его рассматриваем внутри методологии Scrum.

**Пример Диаграммы сгорания задач:**

Синим цветом на диаграмме сгорания отмечена идеальная линия выполнения задач, на которую и следует опираться.

Красным цветом отмечена реальная история выполнения задач.

**По шкале Y**отмечают количество запланированных баллов (в данном случае), идеальные часы, количество задач и так далее.

**По шкале X**отмечают количество дней до окончания Sprint.

Как может показаться на первый взгляд, данная диаграмма сгорания задач / Burndown chart служит всего лишь для самоконтроля и самоотчета, однако ее использование может рассказать об очень многом.

**Читаем Диаграмму сгорания задач / Burndown chart**

Начнем с примеров негативных результатов как ведения графика, так и самой работы команды и закончим более качественными.

**1. Burndown Chart: Слишком рано**

е

По диаграмме сгорания задач/Burndown chart отчетливо видно, что команда все задачи выполнила раньше срока. Такая ситуация тоже не является позитивной, так как это означает ряд совершенных проблем:

1. Команда сделала неправильную оценку предстоящей работы.
2. В случае быстрого выполнения задач, разработчики не добавляли задачи из следующего спринта.
3. Команда сильно перестраховалась, включив изначально дополнительный срок.

В случае такой проблемы, чаще всего Scrum Master спрашивает команду о возможности добавления дополнительных задач из Product Backlog.

**2. Burndown Chart: Опоздали**

Также один из видов негативных диаграмм сгорания задач.

Одной из возможных причин здесь может быть постоянное добавление новых задач во время спринта, что увеличило нагрузку.

Второй частой проблемой является недоделанность задач, когда задачи сделаны наполовину. Такие задачи, как выразился Джефф Сазерленд, являются хламом.

В такой ситуации, на Daily Scrum Meeting обязательно нужно говорить о проблемах, мешающих идти к цели ровной дорогой. Как только линия реальных задач пошла выше, сразу надо решать проблему – это также один из постулатов методологии Scrum.

**3. Burndown Chart: Без оценок**

Может быть даже команда и работала, только забыла или не захотела использовать диаграмму сжигания задач, что является прямо сказать дурным тоном и противоречит эффективной работе. Команда не может контролировать себя, не может совершенствоваться и так далее.

**4. Burndown Chart: Конечная оценка**

Собственно, ситуация равна предыдущей. Не смотря на законченный Sprint, все итоговые оценки были внесены в диаграмму сгорания в самый последний день после завершения работы. Это равносильно тому, когда вообще законченные задачи не вносятся. По данному графику невозможно сделать вывода о правильности работы команды и, даже более того, можно предположить, что команда не стремится к развитию.

**5. Burndown Chart: Zero**

Отсутствие показателя реальных задач в диаграмме не является поводом считать, что работа не производилась, ведь она могла быть просто не оценена. Как и в предыдущих пунктах, такая позиция не позволяет контролировать работу собственной команды и совершенствоваться.

**6. Burndown Chart: Релаксирующая команда**

Этот пример диаграммы сгорания задач уже значительно лучше, нежели другие, ведь в нем можно увидеть, как усовершенствовать команду. Возможные проблемы здесь такие же, как и в пункте «Слишком рано», но Scrum Team решили не заканчивать Sprint раньше, а более расслаблено продолжить работу, что также является ошибкой.

**7. Burndown Chart: Совершенствование**

Scrum Team на текущих показателях выглядит достаточно хорошо. По линиям видно, что в самом начале были трудности, но во время Daily Scrum Meeting все вопросы вскрывались и Scrum Master исправлял работу, ведя команду к цели.

Также, возможно, группа делала принципиальное ускорение для достижения цели.

Еще одной причиной, к примеру, может быть то, что команда брала дополнительные задачи.

**8. Burndown Chart: Опыт**

На лицо опытная группа, которая, после начала работы, сразу исправляет все возникающие трудности и совершенствуется так, что резко переходит к активному сжиганию.

**9. Burndown Chart: A++**

Бесконечно можно смотреть на три вещи: как горит огонь, как течет вода и как строится идеальный график =).

**Матрица соответствия требований (Requirements Traceability Matrix)**

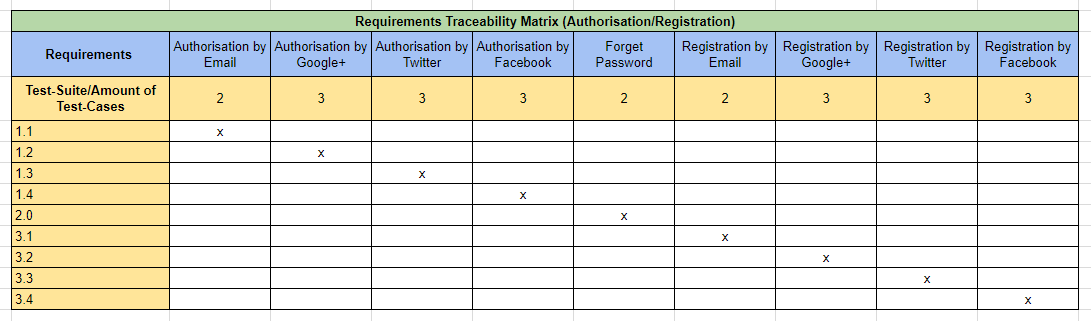
Это двумерная таблица, содержащая соответствие функциональных требований (functional requirements) продукта и подготовленных тестовых сценариев (test cases). В заголовках колонок таблицы расположены требования, а в заголовках строк — тестовые сценарии. На пересечении — отметка, означающая, что требование текущей колонки покрыто тестовым сценарием текущей строки. Матрица обычно хранится в виде электронной таблицы.

Матрица соответствия требований используется QA-инженерами для валидации покрытия требований по продукту тестами.

**Цель «Traceability Matrix»:**

* какие требования «покрыты» тестами, а какие нет.
* избыточность тестов (одно функциональное требование покрыто большим количеством тестов).

Данный тестовый артефакт является неотъемлемой частью тестирования.

Табл. "Requirements Traceability Matrix"