Игнорирование т.н. «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, 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.

2.6. Оценка трудозатрат, планирование и отчётность

2.6.1. Планирование и отчётность

В главе «Логика создания эффективных проверок»{147} мы на примере «Конвертера файлов» рассуждали о том, как при минимальных трудозатратах получить максимальный эффект от тестирования. Это было достаточно просто, т.к. наше приложение смехотворно по своим масштабам. Но давайте представим, что тестировать приходится реальный проект, где требования в «страничном эквиваленте» занимают сотни и даже тысячи страниц. Также давайте вспомним главу «Подробная классификация тестирования»{64} с её несколькими десятками видов тестирования (и это без учёта того факта, что их можно достаточно гибко комбинировать, получая новые варианты) и подумаем, как применить все эти знания (и открываемые ими возможности) в крупном проекте.

Даже если допустить, что мы идеально знаем все технические аспекты предстоящей работы, неотвеченными остаются такие вопросы, как:

- Когда и с чего начать?
- Всё ли необходимое для выполнения работы у нас есть? Если нет, где взять недостающее?
- В какой последовательности выполнять разные виды работ?
- Как распределить ответственность между участниками команды?
- Как организовать отчётность перед заинтересованными лицами?
- Как объективно определять прогресс и достигнутые успехи?
- Как заранее увидеть возможные проблемы, чтобы успеть их предотвратить?
- Как организовать нашу работу так, чтобы при минимуме затрат получить максимум результата?

Эти и многие подобные им вопросы уже лежат вне технической области — они относятся к управлению проектом. Эта задача сама по себе огромна, потому мы рассмотрим лишь малую её часть, с которой многим тестировщикам приходится иметь дело, — планирование и отчётность.

Вспомним жизненный цикл тестирования{27}: каждая итерация начинается с планирования и заканчивается отчётностью, которая становится основой для планирования следующей итерации — и так далее (см. рисунок 2.6.a). Таким образом, планирование и отчётность находятся в тесной взаимосвязи, и проблемы с одним из этих видов деятельности неизбежно приводят к проблемам с другим видом, а в конечном итоге и к проблемам с проектом в целом.

Рисунок 2.6.a — Взаимосвязь (взаимозависимость) планирования и отчётности

Если выразить эту мысль чётче и по пунктам, получается:

- Без качественного планирования не ясно, кому и что нужно делать.
- Когда не ясно, кому и что нужно делать, работа выполняется плохо.
- Когда работа выполнена плохо и не ясны точные причины, невозможно сделать правильные выводы о том, как исправить ситуацию.
- Без правильных выводов невозможно создать качественный отчёт о результатах работы.
- Без качественного отчёта о результатах работы невозможно создать качественный план дальнейшей работы.
- Всё. Порочный круг замкнулся. Проект умирает.

Казалось бы, так и в чём же сложность? Давайте будем хорошо планировать и писать качественные отчёты — и всё будет хорошо. Проблема в том, что соответствующие навыки развиты в достаточной мере у крайне небольшого процента людей. Если вы не верите, вспомните, как учили материал в ночь перед экзаменом, как опаздывали на важные встречи и… повторяли это раз за разом, так и не сделав выводов. (Если в вашей жизни такого не было, можно надеяться, что вам повезло оказаться в том небольшом проценте людей, у которых соответствующие навыки развиты хорошо.)

Корень проблемы состоит в том, что планированию и отчётности в школах и университетах учат достаточно поверхностно, при этом (увы) на практике часто выхолащивая эти понятия до пустой формальности (планов, на которые никто не смотрит, и отчётов, которые никто не читает; опять же, кому-то повезло увидеть строго обратную ситуацию, но явно немногим).

Итак, к сути. Сначала рассмотрим классические определения.

Планирование (planning[327]) — непрерывный процесс принятия управленческих решений и методической организации усилий по их реализации с целью обеспечения качества некоторого процесса на протяжении длительного периода времени.

К высокоуровневым задачам планирования относятся:

- снижение неопределённости;
- повышение эффективности;
- улучшение понимания целей;
- создание основы для управления процессами.

Отчётность (reporting[328]) — 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.

[327] **Planning** is a continuous process of making entrepreneurial decisions with an eye to the future, and methodically organizing the effort needed to carry out these decisions. There are four basic reasons for project planning: to eliminate or reduce uncertainty; to improve efficiency of the operation; to obtain a better understanding of the objectives; to provide a basis for monitoring and controlling work. [«Project Management: A Systems Approach to Planning, Scheduling, and Controlling», Harold Kerzner]

[328] **Reporting** — collecting and distributing performance information (including status reporting, progress measurement, and forecasting). [PMBOK, 3rd edition]

Как и было упомянуто ранее, планирование и отчётность относятся к области управления проектом, которая выходит за рамки данной книги.

Если вас интересуют детали, рекомендуется ознакомиться с двумя фундаментальными источниками информации:

- «Project Management: A Systems Approach to Planning, Scheduling, and Controlling», Harold Kerzner.
- PMBOK («Project Management Body of Knowledge»).

Мы же переходим к более конкретным вещам, с которыми приходится работать (пусть и на уровне использования, а не создания) даже начинающему тестировщику.

2.6.2. Тест-план и отчёт о результатах тестирования

Тест-план

Тест-план (test plan[329]) — документ, описывающий и регламентирующий перечень работ по тестированию, а также соответствующие техники и подходы, стратегию, области ответственности, ресурсы, расписание и ключевые даты.

К низкоуровневым задачам планирования в тестировании относятся:

- оценка объёма и сложности работ;
- определение необходимых ресурсов и источников их получения;
- определение расписания, сроков и ключевых точек;
- оценка рисков и подготовка превентивных контрмер;
- распределение обязанностей и ответственности;
- согласование работ по тестированию с деятельностью участников проектной команды, занимающихся другими задачами.

Как и любой другой документ, тест-план может быть качественным или обладать недостатками. Качественный тест-план обладает большинством свойств качественных требований{41}, а также расширяет их набор следующими пунктами:

- Реалистичность (запланированный подход реально выполним).
- Гибкость (качественный тест-план не только является модифицируемым с точки зрения работы с документом, но и построен таким образом, чтобы при возникновении непредвиденных обстоятельств допускать быстрое изменение любой из своих частей без нарушения взаимосвязи с другими частями).
- Согласованность с общим проектным планом и иными отдельными планами (например, планом разработки).

Тест-план создаётся в начале проекта и дорабатывается по мере необходимости на протяжении всего времени жизни проекта при участии наиболее квалифицированных представителей проектной команды, задействованных в обеспечении качества. Ответственным за создание тест-плана, как правило, является ведущий тестировщик («тест-лид»).

В общем случае тест-план включает следующие разделы (примеры их наполнения будут показаны далее, потому здесь — только перечисление).

- **Цель** (purpose). Предельно краткое описание цели разработки приложения (частично это напоминает бизнес-требования{36}, но здесь информация подаётся в ещё более сжатом виде и в контексте того, на что следует обращать первостепенное внимание при организации тестирования и повышения качества).
- **Области, подвергаемые тестированию** (features to be tested). Перечень функций и/или нефункциональных особенностей приложения, которые будут подвергнуты тестированию. В некоторых случаях здесь также приводится приоритет соответствующей области.
- **Области, не подвергаемые тестированию** (features not to be tested). Перечень функций и/или нефункциональных особенностей приложения, которые не будут подвергнуты тестированию. Причины исключения той или иной об-

[329] **Test plan.** A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process. [ISTQB Glossary]

ласти из списка тестируемых могут быть самыми различными — от предельно низкой их важности для заказчика до нехватки времени или иных ресурсов. Этот перечень составляется, чтобы у проектной команды и иных заинтересованных лиц было чёткое единое понимание, что тестирование таких-то особенностей приложения не запланировано — такой подход позволяет исключить появление ложных ожиданий и неприятных сюрпризов.

- **Тестовая стратегия** (test strategy[330]) **и подходы** (test approach[331]). Описание процесса тестирования с точки зрения применяемых методов, подходов, видов тестирования, технологий, инструментальных средств и т.д.
- **Критерии** (criteria). Этот раздел включает следующие подразделы:
  - **Приёмочные критерии, критерии качества** (acceptance criteria[332]) — любые объективные показатели качества, которым разрабатываемый продукт должен соответствовать с точки зрения заказчика или пользователя, чтобы считаться готовым к эксплуатации.
  - **Критерии начала тестирования** (entry criteria[333]) — перечень условий, при выполнении которых команда приступает к тестированию. Наличие этого критерия страхует команду от бессмысленной траты усилий в условиях, когда тестирование не принесёт ожидаемой пользы.
  - **Критерии приостановки тестирования** (suspension criteria[334]) — перечень условий, при выполнении которых тестирование приостанавливается. Наличие этого критерия также страхует команду от бессмысленной траты усилий в условиях, когда тестирование не принесёт ожидаемой пользы.
  - **Критерии возобновления тестирования** (resumption criteria[335]) — перечень условий, при выполнении которых тестирование возобновляется (как правило, после приостановки).
  - **Критерии завершения тестирования** (exit criteria[336]) — перечень условий, при выполнении которых тестирование завершается. Наличие этого критерия страхует команду как от преждевременного прекращения тестирования, так и от продолжения тестирования в условиях, когда оно уже перестаёт приносить ощутимый эффект.
- **Ресурсы** (resources). В данном разделе тест-плана перечисляются все необходимые для успешной реализации стратегии тестирования ресурсы, которые в общем случае можно разделить на:
  - программные ресурсы (какое ПО необходимо команде тестировщиков, сколько копий и с какими лицензиями (если речь идёт о коммерческом ПО));

---

[330] **Test strategy.** A high-level description of the test levels to be performed and the testing within those levels (group of test activities that are organized and managed together, e.g. component test, integration test, system test and acceptance test) for an organization or program (one or more projects). [ISTQB Glossary]

[331] **Test approach.** The implementation of the test strategy for a specific project. It typically includes the decisions made that follow based on the (test) project's goal and the risk assessment carried out, starting points regarding the test process, the test design techniques to be applied, exit criteria and test types to be performed. [ISTQB Glossary]

[332] **Acceptance criteria.** The exit criteria that a component or system must satisfy in order to be accepted by a user, customer, or other authorized entity. [ISTQB Glossary]

[333] **Entry criteria.** The set of generic and specific conditions for permitting a process to go forward with a defined task, e.g. test phase. The purpose of entry criteria is to prevent a task from starting which would entail more (wasted) effort compared to the effort needed to remove the failed entry criteria. [ISTQB Glossary]

[334] **Suspension criteria.** The criteria used to (temporarily) stop all or a portion of the testing activities on the test items. [ISTQB Glossary]

[335] **Resumption criteria.** The criteria used to restart all or a portion of the testing activities that were suspended previously. [ISTQB Glossary]

[336] **Exit criteria.** The set of generic and specific conditions, agreed upon with the stakeholders for permitting a process to be officially completed. The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task which have not been finished. Exit criteria are used to report against and to plan when to stop testing. [ISTQB Glossary]

  - аппаратные ресурсы (какое аппаратное обеспечение, в каком количестве и к какому моменту необходимо команде тестировщиков);
  - человеческие ресурсы (сколько специалистов какого уровня и со знаниями в каких областях должно подключиться к команде тестировщиков в тот или иной момент времени);
  - временные ресурсы (сколько по времени займёт выполнение тех или иных работ);
  - финансовые ресурсы (в какую сумму обойдётся использование имеющихся или получение недостающих ресурсов, перечисленных в предыдущих пунктах этого списка); во многих компаниях финансовые ресурсы могут быть представлены отдельным документом, т.к. являются конфиденциальной информацией.
- **Расписание** (test schedule[337]). Фактически это календарь, в котором указано, что и к какому моменту должно быть сделано. Особое внимание уделяется т.н. «ключевым точкам» (milestones), к моменту наступления которых должен быть получен некий значимый ощутимый результат.
- **Роли и ответственность** (roles and responsibility). Перечень необходимых ролей (например, «ведущий тестировщик», «эксперт по оптимизации производительности») и область ответственности специалистов, выполняющих эти роли.
- **Оценка рисков** (risk evaluation). Перечень рисков, которые с высокой вероятностью могут возникнуть в процессе работы над проектом. По каждому риску даётся оценка представляемой им угрозы и приводятся варианты выхода из ситуации.
- **Документация** (documentation). Перечень используемой тестовой документации с указанием, кто и когда должен её готовить и кому передавать.
- **Метрики** (metrics[338]). Числовые характеристики показателей качества, способы их оценки, формулы и т.д. На этот раздел, как правило, формируется множество ссылок из других разделов тест-плана.

Метрики в тестировании являются настолько важными, что о них мы поговорим отдельно. Итак.

Метрика (metric[338]) — числовая характеристика показателя качества. Может включать описание способов оценки и анализа результата.

Сначала поясним важность метрик на тривиальном примере. Представьте, что заказчик интересуется текущим положением дел и просит вас кратко охарактеризовать ситуацию с тестированием на проекте. Общие слова в стиле «всё хорошо», «всё плохо», «нормально» и тому подобное его, конечно, не устроят, т.к. они предельно субъективны и могут быть крайне далеки от реальности. И совсем иначе выглядит ответ наподобие такого: «Реализовано 79 % требований (в т.ч. 94 % важных), за последние три спринта тестовое покрытие выросло с 63 % до 71 %, а общий показатель прохождения тест-кейсов вырос с 85 % до 89 %. Иными словами, мы полностью укладываемся в план по всем ключевым показателям, а по разработке даже идём с небольшим опережением».

[337] **Test schedule.** A list of activities, tasks or events of the test process, identifying their intended start and finish dates and/or times, and interdependencies. [ISTQB Glossary]

[338] **Metric.** A measurement scale and the method used for measurement. [ISTQB Glossary]

Чтобы оперировать всеми этими числами (а они нужны не только для отчётности, но и для организации работы проектной команды), их нужно как-то вычислить. Именно это и позволяют сделать метрики. Затем вычисленные значения можно использовать для:

- принятия решений о начале, приостановке, возобновлении или прекращении тестирования (см. выше раздел «Критерии» тест-плана);
- определения степени соответствия продукта заявленным критериям качества;
- определения степени отклонения фактического развития проекта от плана;
- выявления «узких мест», потенциальных рисков и иных проблем;
- оценки результативности принятых управленческих решений;
- подготовки объективной информативной отчётности;
- и т.д.

Метрики могут быть как прямыми (не требуют вычислений), так и расчётными (вычисляются по формуле). Типичные примеры прямых метрик — количество разработанных тест-кейсов, количество найденных дефектов и т.д. В расчётных метриках могут использоваться как совершенно тривиальные, так и довольно сложные формулы (см. таблицу 2.6.1).

Таблица 2.6.1 — Примеры расчётных метрик

Простая расчётная метрика	**Сложная расчётная метрика**
$T^{SP} = \frac{T^{Success}}{T^{Total}} \cdot 100\%$, где T^{SP} — процентный показатель успешного прохождения тест-кейсов, $T^{Success}$ — количество успешно выполненных тест-кейсов, T^{Total} — общее количество выполненных тест-кейсов. Минимальные границы значений: • Начальная фаза проекта: 10 %. • Основная фаза проекта: 40 %. • Финальная фаза проекта: 85 %.	$T^{SC} = \sum_{Level}^{MaxLevel} \frac{(T_{Level} \cdot I)^{R_{Level}}}{B_{Level}}$, где T^{SC} — интегральная метрика прохождения тест-кейсов во взаимосвязи с требованиями и дефектами, T_{Level} — степень важности тест-кейса, I — количество выполнений тест-кейса, R_{Level} — степень важности требования, проверяемого тест-кейсом, B_{Level} — количество дефектов, обнаруженных тест-кейсом. Способ анализа: • Идеальным состоянием является непрерывный рост значения T^{SC}. • В случае отрицательной динамики уменьшение значения T^{SC} на 15 % и более за последние три спринта может трактоваться как недопустимое и являться достаточным поводом для приостановки тестирования.

В тестировании существует большое количество общепринятых метрик, многие из которых могут быть собраны автоматически с использованием инструментальных средств управления проектами. Например[339]:

- процентное отношение (не) выполненных тест-кейсов ко всем имеющимся;
- процентный показатель успешного прохождения тест-кейсов (см. «Простая расчётная метрика» в таблице 2.6.1);
- процентный показатель заблокированных тест-кейсов;
- плотность распределения дефектов;
- эффективность устранения дефектов;
- распределение дефектов по важности и срочности;
- и т.д.

[339] «Important Software Test Metrics and Measurements — Explained with Examples and Graphs» [http://www.softwaretestinghelp.com/software-test-metrics-and-measurements/]

Как правило, при формировании отчётности нас будет интересовать не только текущее значение метрики, но и её динамика во времени, которую очень удобно изображать графически (что тоже могут выполнять автоматически многие средства управления проектами).

Некоторые метрики могут вычисляться на основе данных о расписании, например метрика «сдвига расписания»:

$ScheduleSlippage = \frac{DaysToDeadline}{NeededDays} - 1$, где

$ScheduleSlippage$ — значение сдвига расписания,

$DaysToDeadline$ — количество дней до запланированного завершения работы,

$NeededDays$ — количество дней, необходимое для завершения работы.

Значение $ScheduleSlippage$ не должно становиться отрицательным.

Таким образом, мы видим, что метрики являются мощнейшим средством сбора и анализа информации. И вместе с тем здесь кроется опасность: ни при каких условиях нельзя допускать ситуации «метрик ради метрик», когда инструментальное средство собирает уйму данных, вычисляет множество чисел и строит десятки графиков, но… никто не понимает, как их трактовать. Обратите внимание, что к обеим метрикам в таблице 2.6.1 и к только что рассмотренной метрике $ScheduleSlippage$ прилагается краткое руководство по их трактовке. И чем сложнее и уникальнее метрика, тем более подробное руководство необходимо для её эффективного применения.

И, наконец, стоит упомянуть про так называемые «метрики покрытия», т.к. они очень часто упоминаются в различной литературе.

Покрытие (coverage[340]) — процентное выражение степени, в которой исследуемый элемент (coverage item[341]) затронут соответствующим набором тест-кейсов.

Самыми простыми представителями метрик покрытия можно считать:

- Метрику покрытия требований (требование считается «покрытым», если на него ссылается хотя бы один тест-кейс):

  $R^{SimpleCoverage} = \frac{R^{Covered}}{R^{Total}} \cdot 100\%$, где

  $R^{SimpleCoverage}$ — метрика покрытия требований,

  $R^{Covered}$ — количество требований, покрытых хотя бы одним тест-кейсом,

  R^{Total} — общее количество требований.

- Метрику плотности покрытия требований (учитывается, сколько тест-кейсов ссылается на несколько требований):

  $R^{DensityCoverage} = \frac{\sum T_i}{T^{Total}.R^{Total}} \cdot 100\%$, где

  $R^{DensityCoverage}$ — плотность покрытия требований,

  T_i — количество тест-кейсов, покрывающих *i*-е требование,

  T^{Total} — общее количество тест-кейсов,

  R^{Total} — общее количество требований.

[340] **Coverage, Test coverage.** The degree, expressed as a percentage, to which a specified coverage item has been exercised by a test suite. [ISTQB Glossary]

[341] **Coverage item.** An entity or property used as a basis for test coverage, e.g. equivalence partitions or code statements. [ISTQB Glossary]

- Метрику покрытия классов эквивалентности (анализируется, сколько классов эквивалентности затронуто тест-кейсами).
  $E^{Coverage} = \frac{E^{Covered}}{E^{Total}} \cdot 100\%$, где
  $E^{Coverage}$ — метрика покрытия классов эквивалентности,
  $E^{Covered}$ — количество классов эквивалентности, покрытых хотя бы одним тест-кейсом,
  E^{Total} — общее количество классов эквивалентности.

- Метрику покрытия граничных условий (анализируется, сколько значений из группы граничных условий затронуто тест-кейсами).
  $B^{Coverage} = \frac{B^{Covered}}{B^{Total}} \cdot 100\%$, где
  $B^{Coverage}$ — метрика покрытия граничных условий,
  $B^{Covered}$ — количество граничных условий, покрытых хотя бы одним тест-кейсом,
  B^{Total} — общее количество граничных условий.

- Метрики покрытия кода модульными тест-кейсами. Таких метрик очень много, но вся их суть сводится к выявлению некоей характеристики кода (количество строк, ветвей, путей, условий и т.д.) и определению, какой процент представителей этой характеристики покрыт тест-кейсами.

Метрик покрытия настолько много, что даже в ISTQB-глоссарии дано определение полутора десяткам таковых. Вы можете найти эти определения, выполнив в файле ISTQB-глоссария поиск по слову «coverage».

На этом мы завершаем теоретическое рассмотрение планирования и переходим к примеру — учебному тест-плану для нашего приложения «Конвертер файлов[{55}]». Напомним, что приложение является предельно простым, потому и тест-план будет очень маленьким (однако, обратите внимание, сколь значительную его часть будет занимать раздел метрик).

Пример тест-плана

Для того чтобы заполнить некоторые части тест-плана, нам придётся сделать допущения о составе проектной команды и времени, отведённом на разработку проекта. Поскольку данный тест-план находится внутри текста книги, у него нет таких типичных частей, как заглавная страница, содержание и т.п. Итак.

Цель

Корректное автоматическое преобразование содержимого документов к единой кодировке с производительностью, значительно превышающей производительность человека при выполнении аналогичной задачи.

Области, подвергаемые тестированию

(См. соответствующие разделы требований.)

- ПТ-1.*: дымовой тест.
- ПТ-2.*: дымовой тест, тест критического пути.
- ПТ-3.*: тест критического пути.
- БП-1.*: дымовой тест, тест критического пути.
- АК-2.*: дымовой тест, тест критического пути.
- О-4: дымовой тест.
- О-5: дымовой тест.
- ДС-*: дымовой тест, тест критического пути.

Области, не подвергаемые тестированию

- СХ-1: приложение разрабатывается как консольное.
- СХ-2, О-1, О-2: приложение разрабатывается на PHP указанной версии.
- АК-1.1: заявленная характеристика находится вблизи нижней границы производительности операций, характерных для разрабатываемого приложения.
- О-3: не требует реализации.
- О-6: не требует реализации.

Тестовая стратегия и подходы

Общий подход.

Специфика работы приложения состоит в однократном конфигурировании опытным специалистом и дальнейшем использовании конечными пользователями, для которых доступна только одна операция — размещение файла в каталоге-приёмнике. Потому вопросы удобства использования, безопасности и т.п. не исследуются в процессе тестирования.

Уровни функционального тестирования:

- Дымовой тест: автоматизированный с использованием командных файлов ОС Windows и Linux.
- Тест критического пути: выполняется вручную.
- Расширенный тест не производится, т.к. для данного приложения вероятность обнаружения дефектов на этом уровне пренебрежимо мала.

В силу кроссфункциональности команды значительного вклада в повышение качества можно ожидать от аудита кода, совмещённого с ручным тестированием по методу белого ящика. Автоматизация тестирования кода не будет применяться в силу крайне ограниченного времени.

Критерии

- Приёмочные критерии: успешное прохождение 100 % тест-кейсов уровня дымового тестирования и 90 % тест-кейсов уровня критического пути (см. метрику «Успешное прохождение тест-кейсов») при условии устранения 100 % дефектов критической и высокой важности (см. метрику «Общее устранение дефектов»). Итоговое покрытие требований тест-кейсами (см. метрику «Покрытие требований тест-кейсами») должно составлять не менее 80 %.
- Критерии начала тестирования: выход билда.
- Критерии приостановки тестирования: переход к тесту критического пути допустим только при успешном прохождении 100 % тест-кейсов дымового теста (см. метрику «Успешное прохождение тест-кейсов»); тестирование может быть приостановлено в случае, если при выполнении не менее 25 % запланированных тест-кейсов более 50 % из них завершились обнаружением дефекта (см. метрику «Стоп-фактор»).
- Критерии возобновления тестирования: исправление более 50 % обнаруженных на предыдущей итерации дефектов (см. метрику «Текущее устранение дефектов»).
- Критерии завершения тестирования: выполнение более 80 % запланированных на итерацию тест-кейсов (см. метрику «Выполнение тест-кейсов»).

Ресурсы

- Программные ресурсы: четыре виртуальных машины (две с ОС Windows 7 Ent x64, две с ОС Linux Ubuntu 14 LTS x64), две копии PHP Storm 8.
- Аппаратные ресурсы: две стандартных рабочих станции (8GB RAM, i7 3GHz).

- Человеческие ресурсы:
    - Старший разработчик с опытом тестирования (100%-я занятость на всём протяжении проекта). Роли на проекте: лидер команды, старший разработчик.
    - Тестировщик со знанием PHP (100%-я занятость на всём протяжении проекта). Роль на проекте: тестировщик.
- Временные ресурсы: одна рабочая неделя (40 часов).
- Финансовые ресурсы: согласно утверждённому бюджету. Дополнительные финансовые ресурсы не требуются.

Расписание

- 25.05 — формирование требований.
- 26.05 — разработка тест-кейсов и скриптов для автоматизированного тестирования.
- 27.05–28.05 — основная фаза тестирования (выполнение тест-кейсов, написание отчётов о дефектах).
- 29.05 — завершение тестирования и подведение итогов.

Роли и ответственность

- Старший разработчик: участие в формировании требований, участие в аудите кода.
- Тестировщик: формирование тестовой документации, реализация тестирования, участие в аудите кода.

Оценка рисков

- Персонал (вероятность низкая): в случае нетрудоспособности какого-либо из участников команды можно обратиться к представителям проекта «Каталогизатор» для предоставления временной замены (договорённость с менеджером «Каталогизатора» Джоном Смитом достигнута).
- Время (вероятность высокая): заказчиком обозначен крайний срок сдачи 01.06, потому время является критическим ресурсом. Рекомендуется приложить максимум усилий к тому, чтобы фактически завершить проект 28.05 с тем, чтобы один день (29.05) остался в запасе.
- Иные риски: иных специфических рисков не выявлено.

Документация

- Требования. Ответственный — тестировщик, дата готовности 25.05.
- Тест-кейсы и отчёты о дефектах. Ответственный — тестировщик, период создания 26.05–28.05.
- Отчёт о результатах тестирования. Ответственный — тестировщик, дата готовности 29.05.

Метрики

- Успешное прохождение тест-кейсов:
  $T^{SP} = \frac{T^{Success}}{T^{Total}} \cdot 100\%$, где

  T^{SP} — процентный показатель успешного прохождения тест-кейсов,
  $T^{Success}$ — количество успешно выполненных тест-кейсов,
  T^{Total} — общее количество выполненных тест-кейсов.
  Минимальные границы значений:
    - Начальная фаза проекта: 10%.
    - Основная фаза проекта: 40%.
    - Финальная фаза проекта: 80%.

- Общее устранение дефектов:

  $D_{Level}^{FTP} = \frac{D_{Level}^{Closed}}{D_{Level}^{Found}} \cdot 100\%$, где

  D_{Level}^{FTP} — процентный показатель устранения дефектов уровня важности $Level$ за время существования проекта
  D_{Level}^{Closed} — количество устранённых за время существования проекта дефектов уровня важности $Level$,
  D_{Level}^{Found} — количество обнаруженных за время существования проекта дефектов уровня важности $Level$.
  Минимальные границы значений:

		Важность дефекта			
		Низкая	Средняя	Высокая	Критическая
Фаза проекта	Начальная	10%	40%	50%	80%
	Основная	15%	50%	75%	90%
	Финальная	20%	60%	100%	100%

- Текущее устранение дефектов:

  $D_{Level}^{FCP} = \frac{D_{Level}^{Closed}}{D_{Level}^{Found}} \cdot 100\%$, где

  D_{Level}^{FCP} — процентный показатель устранения в текущем билде дефектов уровня важности $Level$, обнаруженных в предыдущем билде,
  D_{Level}^{Closed} — количество устранённых в текущем билде дефектов уровня важности $Level$,
  D_{Level}^{Found} — количество обнаруженных в предыдущем билде дефектов уровня важности $Level$.
  Минимальные границы значений:

		Важность дефекта			
		Низкая	Средняя	Высокая	Критическая
Фаза проекта	Начальная	60%	60%	60%	60%
	Основная	65%	70%	85%	90%
	Финальная	70%	80%	95%	100%

- Стоп-фактор:

  $S = \begin{cases} Yes, T^E \geq 25\% \;\&\&\; T^{SP} < 50\% \\ No, T^E < 25\% \;||\; T^{SP} \geq 50\% \end{cases}$, где

  S — решение о приостановке тестирования,
  T^E — текущее значение метрики T^E,
  T^{SP} — текущее значение метрики T^{SP}.

- Выполнение тест-кейсов:

  $T^E = \frac{T^{Executed}}{T^{Planned}} \cdot 100\%$, где

  T^E — процентный показатель выполнения тест-кейсов,
  $T^{Executed}$ — количество выполненных тест-кейсов,
  $T^{Planned}$ — количество тест-кейсов, запланированных к выполнению.
  Уровни (границы):
  - Минимальный уровень: 80 %.
  - Желаемый уровень: 95–100 %.

- Покрытие требований тест-кейсами:

  $R^C = \frac{R^{Covered}}{R^{Total}} \cdot 100\%$, где

  R^C — процентный показатель покрытия требования тест-кейсами,
  $R^{Covered}$ — количество покрытых тест-кейсами требований,
  R^{Total} — общее количество требований.
  Минимальные границы значений:
  - Начальная фаза проекта: 40 %.
  - Основная фаза проекта: 60 %.
  - Финальная фаза проекта: 80 % (рекомендуется 90 % и более).

Задание 2.6.a: поищите в Интернет более развёрнутые примеры тест-планов. Они периодически появляются, но и столь же быстро удаляются, т.к. настоящие (не учебные) тест-планы, как правило, являются конфиденциальной информацией.

На этом мы завершаем обсуждение планирования и переходим к отчётности, которая завершает цикл тестирования.

Отчёт о результатах тестирования

Отчёт о результатах тестирования (test progress report[342], test summary report[343]) — 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.

Как и любой другой документ, отчёт о результатах тестирования может быть качественным или обладать недостатками. Качественный отчёт о результатах тестирования обладает многими свойствами качественных требований{41}, 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 в первую очередь нужен следующим лицам:

- 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;

[342] **Test progress report.** A document summarizing testing activities and results, produced at regular intervals, to report progress of testing activities against a baseline (such as the original test plan) and to communicate risks and alternatives requiring a decision to management. [ISTQB Glossary]

[343] **Test summary report.** A document summarizing testing activities and results. It also contains an evaluation of the corresponding test items against exit criteria. [ISTQB Glossary]

- 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.

В общем случае отчёт о результатах тестирования включает следующие разделы (примеры их наполнения будут показаны далее, потому здесь — только перечисление).

Важно! Если по поводу тест-плана в сообществе тестировщиков есть более-менее устоявшееся мнение, то формы отчётов о результатах тестирования исчисляются десятками (особенно, если отчёт привязан к некоторому отдельному виду тестирования). Здесь приведён наиболее универсальный вариант, который может быть адаптирован под конкретные нужды.

- **Краткое описание** (summary). 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).

Важно! Различайте краткое описание отчёта о результатах тестирования и краткое описание отчёта о дефекте[170]! При одинаковом названии они создаются по разным принципам и содержат разную информацию!

- **Команда тестировщиков** (test team). Список участников проектной команды, задействованных в обеспечении качества, с указанием их должностей и ролей в подотчётный период.
- **Описание процесса тестирования** (testing process description). Последовательное описание того, какие работы были выполнены за подотчётный период.
- **Расписание** (timetable). Детализированное расписание работы команды тестировщиков и/или личные расписания участников команды.
- **Статистика по новым дефектам** (new defects statistics). Таблица, в которой представлены данные по обнаруженным за подотчётный период дефектам (с классификацией по стадии жизненного цикла и важности).
- **Список новых дефектов** (new defects list). Список обнаруженных за подотчётный период дефектов с их краткими описаниями и важностью.
- **Статистика по всем дефектам** (overall defects statistics). 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** (recommendations). 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** (appendixes). 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 (см. рисунок 2.6.b), особенно актуальной для таких разделов отчёта о результатах тестирования, как краткое описание (summary) и рекомендации (recommendations):

- Выводы строятся на основе целей (которые были отражены в плане).
- Выводы дополняются рекомендациями.
- Как выводы, так и рекомендации строго обосновываются.
- Обоснование опирается на объективные факты.

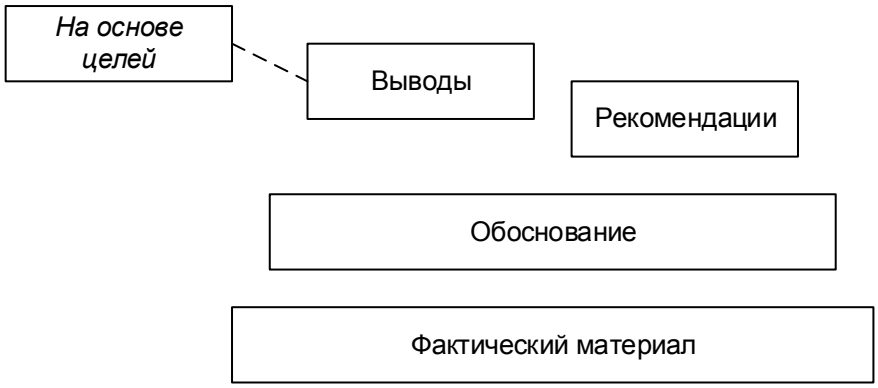


Рисунок 2.6.b — Универсальная логика отчётности

Выводы должны быть:

- Краткими. Сравните:

Плохо	Хорошо
1.17. Как показал глубокий анализ протоколов о выполнении тестирования, можно сделать достаточно уверенные выводы о том, что основная часть функций, отмеченных заказчиком как наиболее важные, функционирует в рамках допустимых отклонений от согласованных на последнем обсуждении с заказчиком метрик качества.	1.11. Базовая функциональность полностью работоспособна (см. 2.1–2.2). 1.23. Существуют некритические проблемы с детализацией сообщений в файле журнала (см. 2.3–2.4). 1.28. Тестирование приложения под ОС Linux не удалось провести из-за недоступности сервера SR-85 (см. 2.5).

- Информативными. Сравните:

Плохо	Хорошо
1.8. Результаты обработки файлов с множественными кодировками, представленными в сопоставимых пропорциях, оставляют желать лучшего. 1.9. Приложение не запускается при некоторых значениях параметров командной строки. 1.10. Непонятно, что происходит с анализом изменения содержимого входного каталога.	1.8. Обнаружены серьёзные проблемы с библиотекой распознавания кодировок (см. BR 834). 1.9. Нарушена функциональность анализа параметров командной строки (см. BR 745, BR 877, BR 878). 1.10. Выявлена нестабильность в работе модуля «Сканер», проводятся дополнительные исследования.

- Полезными для читающего отчёт. Сравните:

Плохо	Хорошо
1.18. Некоторые тесты прошли на удивление хорошо. 1.19. В процессе тестирования мы не испытывали сложности с настройкой среды автоматизации. 1.20. По сравнению с результатами, которые были получены вчера, ситуация немного улучшилась. 1.21. С качеством по-прежнему есть некоторые проблемы. 1.22. Часть команды была в отпуске, но мы всё равно справились.	**Представленного в колонке «Плохо» просто не должно быть в отчёте!**

Рекомендации должны быть:

- Краткими. Да, мы снова говорим о краткости, т.к. её отсутствием страдает слишком большое количество документов. Сравните:

Плохо	Хорошо
2.98. Мы рекомендуем рассмотреть возможные варианты исправления данной ситуации в контексте поиска оптимального решения при условии минимизации усилий разработчиков и максимального повышения соответствия приложения заявленным критериям качества, а именно: исследовать возможность замены некоторых библиотек их более качественными аналогами.	2.98. Необходимо изменить способ определения кодировки текста в документе. Возможные решения: • [сложно, надёжно, но очень долго] написать собственное решение; • [требует дополнительного исследования и согласования] заменить проблемную библиотеку «cflk_n_r_coding» аналогом (возможно, коммерческим).

- Реально выполнимыми. Сравните:

Плохо	Хорошо
2.107. Использовать механизм обработки слов, аналогичный используемому в Google.	2.107. Реализовать алгоритм приведения слов русского языка к именительному падежу (см. описание по ссылке …)
2.304. Не загружать в оперативную память информацию о файлах во входном каталоге.	2.304. Увеличить размер доступной скрипту оперативной памяти на 40-50% (в идеале — до 512 МБ).
2.402. Полностью переписать проект без использования внешних библиотек.	2.402. Заменить собственными решениями функции анализа содержимого каталога и параметров файлов библиотеки «cflk_n_r_flstm».

- Дающими как понимание того, что надо сделать, так и некоторое пространство для принятия собственных решений. Сравните:

Плохо	Хорошо
2.212. Рекомендуем поискать варианты решения этого вопроса. 2.245. Использовать только дисковую сортировку. 2.278. Исключить возможность передачи некорректных имён файла журнала через параметр командной строки.	2.212. Возможные варианты решения: а) … б) [рекомендуем!] … в) … 2.245. Добавить функциональность определения оптимального метода сортировки в зависимости от количества доступной оперативной памяти. 2.278. Добавить фильтрацию имени файла журнала, получаемого через параметр командной строки, с помощью регулярного выражения.

Обоснование выводов и рекомендаций — промежуточное звено между предельно сжатыми результатами анализа и огромным количеством фактических данных. Оно даёт ответы на вопросы наподобие:

- «Почему мы так считаем?»
- «Неужели это так?!»
- «Где взять дополнительные данные?»

Сравните:

Плохо	Хорошо
4.107. Покрытие требований тест-кейсами достаточно. 4.304. Необходимо больше усилий направить на регрессионное тестирование. 4.402. От сокращения сроков разработки стоит отказаться.	4.107. Покрытие требований тест-кейсами вышло на достаточный уровень (значение R^C составило 63 % при заявленном минимуме 60 % для текущей стадии проекта). 4.304. Необходимо больше усилий направить на регрессионное тестирование, т.к. две предыдущих итерации выявили 21 дефект высокой важности (см. список в 5.43) в функциональности, в которой ранее не обнаруживалось проблем. 4.402. От сокращения сроков разработки стоит отказаться, т.к. текущее опережение графика на 30 человеко-часов может быть легко поглощено на стадии реализации требований R84.* и R89.*.

Фактический материал содержит самые разнообразные данные, полученные в процессе тестирования. Сюда могут относиться отчёты о дефектах, журналы работы средств автоматизации, созданные различными приложениями наборы файлов и т.д. Как правило, к отчёту о результатах тестирования прилагаются лишь сокращённые агрегированные выборки подобных данных (если это возможно), а также приводятся ссылки на соответствующие документы, разделы системы управления проектом, пути в хранилище данных и т.д.

На этом мы завершаем теоретическое рассмотрение отчётности и переходим к примеру — учебному отчёту о результатах тестирования нашего приложения «Конвертер файлов»[{55}]. Напомним, что приложение является предельно простым, потому и отчёт о результатах тестирования будет очень маленьким.

Пример отчёта о результатах тестирования

Для того, чтобы заполнить некоторые части отчёта, нам придётся сделать допущения о текущем моменте развития проекта и сложившейся ситуации с качеством. Поскольку данный отчёт находится внутри текста книги, у него нет таких типичных частей, как обложка, содержание и т.п.

Итак.

Краткое описание. За период 26–28 мая было выпущено четыре билда, на последнем из которых успешно прошло 100 % тест-кейсов дымового тестирования и 76 % тест-кейсов тестирования критического пути. 98 % требований высокой важности реализовано корректно. Метрики качества находятся в зелёной зоне, потому есть все основания рассчитывать на завершение проекта в срок (на текущий момент реальный прогресс в точности соответствует плану). На следующую итерацию (29 мая) запланировано выполнение оставшихся низкоприоритетных тест-кейсов.

Команда тестировщиков.

Имя	Должность	Роль
Джо Блэк	Тестировщик	Ответственный за обеспечение качества
Джим Уайт	Старший разработчик	Ответственный за парное тестирование и аудит кода

Описание процесса тестирования. Каждый из четырёх выпущенных за подотчётный период билдов (3–6) был протестирован под ОС Windows 7 Ent x64 и ОС Linux Ubuntu 14 LTS x64 в среде исполнения PHP 5.6.0. Дымовое тестирование (см. http://projects/FC/Testing/SmokeTest) выполнялось с использованием автоматизации на основе командных файлов (см. \\PROJECTS\FC\Testing\Aut\Scripts). Тестирование критического пути (см. http://projects/FC/Testing/CriticalPathTest) выполнялось вручную. Регрессионное тестирование показало высокую стабильность функциональности (обнаружен только один дефект с важностью «средняя»), а повторное тестирование показало ощутимый прирост качества (исправлено 83 % обнаруженных ранее дефектов).

Расписание.

Имя	Дата	Деятельность	Продолжительность, ч
Джо Блэк	27.05.2015	Разработка тест-кейсов	2
Джо Блэк	27.05.2015	Парное тестирование	2
Джо Блэк	27.05.2015	Автоматизация дымового тестирования	1
Джо Блэк	27.05.2015	Написание отчётов о дефектах	2
Джим Уайт	27.05.2015	Аудит кода	1
Джим Уайт	27.05.2015	Парное тестирование	2
Джо Блэк	28.05.2015	Разработка тест-кейсов	3
Джо Блэк	28.05.2015	Парное тестирование	1
Джо Блэк	28.05.2015	Написание отчётов о дефектах	2
Джо Блэк	28.05.2015	Написание отчёта о результатах тестирования	1
Джим Уайт	28.05.2015	Аудит кода	1
Джим Уайт	28.05.2015	Парное тестирование	1

Статистика по новым дефектам.

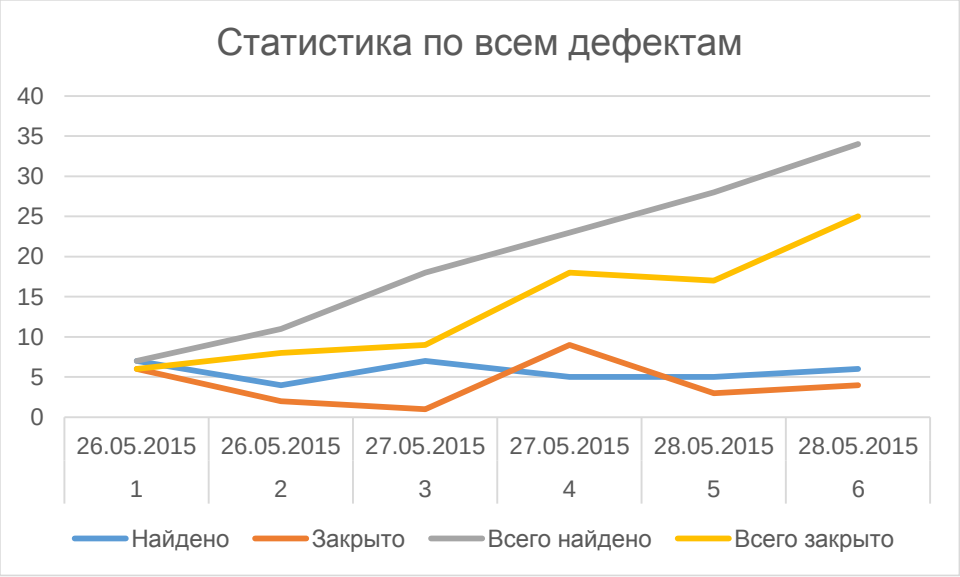
		Важность			
Статус	**Количество**	**Низкая**	**Средняя**	**Высокая**	**Критическая**
Найдено	23	2	12	7	2
Исправлено	17	0	9	6	2
Проверено	13	0	5	6	2
Открыто заново	1	0	0	1	0
Отклонено	3	0	2	1	0

Список новых дефектов.

Идентификатор	Важность	Описание
BR 21	Высокая	Приложение не различает файлы и символические ссылки на файлы.
BR 22	Критическая	Приложение игнорирует файлы .md во входном каталоге.
И так далее — описание всех 23 найденных дефектов.		

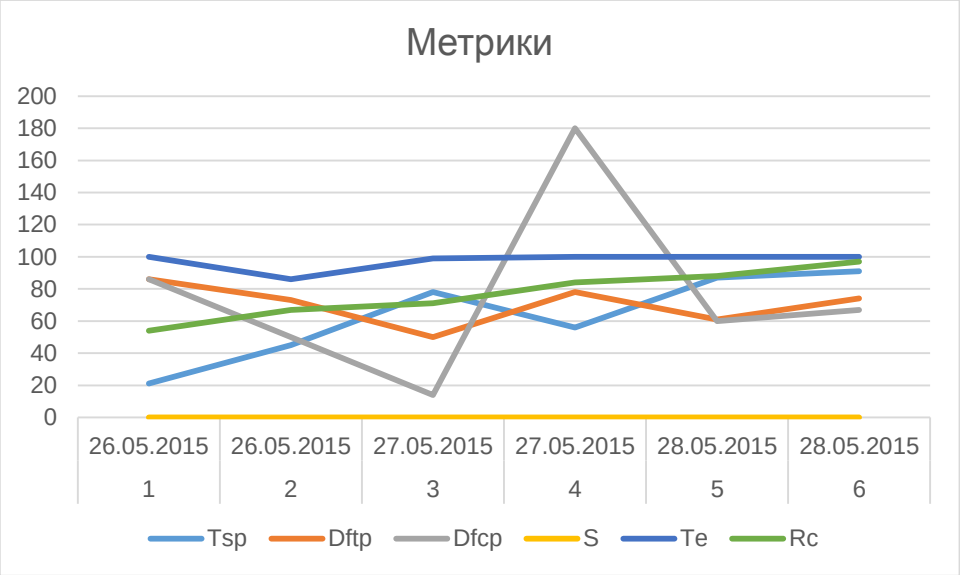
Статистика по всем дефектам.

		Важность			
Статус	**Количество**	**Низкая**	**Средняя**	**Высокая**	**Критическая**
Найдено	34	5	18	8	3
Исправлено	25	3	12	7	3
Проверено	17	0	7	7	3
Открыто заново	1	0	0	1	0
Отклонено	4	0	3	1	0



Рекомендации. В настоящий момент никаких изменений не требуется.

Приложение. График изменения значений метрик.



Задание 2.6.b: поищите в Интернете более развёрнутые примеры отчётов о результатах тестирования. Они периодически появляются, но и столь же быстро удаляются, т.к. настоящие (не учебные) отчёты, как правило, являются конфиденциальной информацией.

2.6.3. Оценка трудозатрат

В завершение данной главы мы снова возвращаемся к планированию, но уже куда более простому — к оценке трудозатрат.

Трудозатраты (man-hours[344]) — количество рабочего времени, необходимого для выполнения работы (выражается в человеко-часах).

Каждый раз, когда вы получаете задание или выдаёте кому-то задание, явно или неявно возникают вопросы наподобие следующих:

- Как много времени понадобится на выполнение работы?
- Когда всё будет готово?
- Можно ли гарантированно выполнить работу к такому-то сроку?
- Каковы наиболее оптимистичный и пессимистичный прогнозы по времени?

Рассмотрим несколько соображений относительно того, как производится оценка трудозатрат.

Любая оценка лучше её отсутствия. Даже если область предстоящей работы для вас совершенно нова, даже если вы ошибётесь в своей оценке на порядок, вы как минимум получите опыт, который сможете использовать в будущем при возникновении подобного рода задач.

Оптимизм губителен. Как правило, люди склонны недооценивать сложность незнакомых задач, что приводит к занижению оценки трудозатрат.

Но даже при достаточно точном определении самих трудозатрат люди без опыта выполнения оценки склонны рассматривать предстоящую работу как некую изолированную деятельность, забывая о том, что на протяжении любого рабочего дня «чистую производительность труда» будут снижать такие факторы, как переписка по почте, участие в собраниях и обсуждениях, решение сопутствующих технических вопросов, изучение документации и обдумывание сложных частей задачи, форс-мажорные обстоятельства (неотложные дела, проблемы с техникой и т.д.).

Таким образом, обязательно стоит учитывать, что в реальности вы сможете заниматься поставленной задачей не 100 % рабочего времени, а меньше (насколько меньше — зависит от конкретной ситуации, в среднем принято считать, что на поставленную задачу из каждых восьми рабочих часов вы сможете потратить не более шести). Учитывая этот факт, стоит сделать соответствующие поправки в оценке общего времени, которое понадобится на выполнение работы (а именно оно чаще всего интересует постановщика задачи).

Оценка должна быть аргументирована. Это не значит, что вы всегда должны пускаться в подробные пояснения, но вы должны быть готовы объяснить, почему вы считаете, что та или иная работа займёт именно столько времени. Во-первых, продумывая эти аргументы, вы получаете дополнительную возможность лучше оценить предстоящую работу и скорректировать оценку. Во-вторых, если ваша оценка не соответствует ожиданиям постановщика задачи, вы сможете отстоять свою точку зрения.

[344] **Man-hour.** A unit for measuring work in industry, equal to the work done by one man in one hour. [http://dictionary.reference.com/browse/man-hour]

Простой способ научиться оценивать — оценивать. В специализированной литературе (см. ниже небольшой список) приводится множество технологий, но первична сама привычка выполнять оценку предстоящей работы. В процессе выработки этой привычки вы естественным образом встретитесь с большинством типичных проблем и через некоторое время научитесь делать соответствующие поправки в оценке, даже не задумываясь.

Что оценивать? Что угодно. Сколько времени у вас уйдёт на прочтение новой книги. За сколько времени вы доедете домой новым маршрутом. За сколько времени вы напишете курсовую или дипломную работу. И так далее. Не важно, что именно вы оцениваете, важно, что вы повторяете это раз за разом, учитывая накапливающийся опыт.

Если вас заинтересовал профессиональный подход к формированию оценки трудозатрат, рекомендуется ознакомиться со следующими источниками:

- «The Mythical Man Month», Frederick Brooks.
- «Controlling Software Projects», Tom De Marco.
- «Software engineering metrics and models», Samuel Conte.

Алгоритм обучения формированию оценки:

- Сформируйте оценку. Ранее уже было отмечено, что нет ничего страшного в том, что полученное значение может оказаться очень далёким от реальности. Для начала оно просто должно быть.
- Запишите полученную оценку. Обязательно именно запишите. Это застрахует вас как минимум от двух рисков: забыть полученное значение (особенно, если работа заняла много времени), соврать себе в стиле «ну, я както примерно так и думал».
- Выполните работу. В отдельных случаях люди склонны подстраиваться под заранее сформированную оценку, ускоряя или замедляя выполнение работы, — это тоже полезный навык, но сейчас такое поведение будет мешать. Однако если вы будете тренироваться на десятках и сотнях различных задач, вы физически не сможете «подстроиться» под каждую из них и начнёте получать реальные результаты.
- Сверьте реальные результаты с ранее сформированной оценкой.
- Учтите ошибки при формировании новых оценок. На этом этапе очень полезно не просто отметить отклонение, а подумать, что привело к его появлению.
- Повторяйте этот алгоритм как можно чаще для самых различных областей жизни. Сейчас цена ваших ошибок крайне мала, а наработанный опыт от этого становится ничуть не менее ценным.

Полезные идеи по формированию оценки трудозатрат:

- Добавляйте небольшой «буфер» (по времени, бюджету или иным критическим ресурсам) на непредвиденные обстоятельства. Чем более дальний прогноз вы строите, тем большим может быть этот «буфер» — от 5–10 % до 30–40 %. Но ни в коем случае не стоит осознанно завышать оценку в разы.
- Выясните свой «коэффициент искажения»: большинство людей в силу особенности своего мышления склонны постоянно или занижать, или завышать оценку. Многократно формируя оценку трудозатрат и сравнивая её впоследствии с реальностью, вы можете заметить определённую закономерность, которую вполне можно выразить числом. Например, может оказаться, что вы склонны занижать оценку в 1.3 раза. Попробуйте в следующий раз внести соответствующую поправку.

- Принимайте во внимание не зависящие от вас обстоятельства. Например, вы точно уверены, что выполните тестирование очередного билда за N человеко-часов, вы учли все отвлекающие факторы и т.д. и решили, что точно закончите к такой-то дате. А потом в реальности выпуск билда задерживается на два дня, и ваш прогноз по моменту завершения работы оказывается нереалистичным.
- Задумывайтесь заранее о необходимых ресурсах. Так, например, необходимую инфраструктуру можно (и нужно!) подготовить (или заказать) заранее, т.к. на подобные вспомогательные задачи может быть потрачено много времени, к тому же основная работа часто не может быть начата, пока не будут завершены все приготовления.
- Ищите способы организовать параллельное выполнение задач. Даже если вы работаете один, всё равно какие-то задачи можно и нужно выполнять параллельно (например, уточнение тест-плана, пока происходит разворачивание виртуальных машин). В случае если работа выполняется несколькими людьми, распараллеливание работы можно считать жизненной необходимостью.
- Периодически сверяйтесь с планом, вносите корректировки в оценку и уведомляйте заинтересованных лиц о внесённых изменениях заблаговременно. Например, вы поняли (как в упомянутом выше примере с задержкой билда), что завершите работу как минимум на два дня позже. Если вы оповестите проектную команду немедленно, у ваших коллег появляется шанс скорректировать свои собственные планы. Если же вы в «час икс» преподнесёте сюрприз о сдвигах срока на два дня, вы создадите коллегам объективную проблему.
- Используйте инструментальные средства — от электронных календарей до возможностей вашей системы управления проектом: это позволит вам как минимум не держать в памяти кучу мелочей, а как максимум — повысит точность формируемой оценки.

Оценка с использованием структурной декомпозиции

С другими техниками формирования оценки вы можете ознакомиться в следующей литературе:

- «Essential Scrum», Kenneth Rubin.
- «Agile Estimating and Planning», Mike Cohn.
- «Extreme programming explained: Embrace change», Kent Beck.
- PMBOK («Project Management Body of Knowledge»).
- Краткий перечень основных техник с пояснениями можно посмотреть в статье «Software Estimation Techniques — Common Test Estimation Techniques used in SDLC[345]».

Структурная декомпозиция (work breakdown structure, WBS[346]) — иерархическая декомпозиция объёмных задач на всё более и более малые подзадачи с целью упрощения оценки, планирования и мониторинга выполнения работы.

345 «Software Estimation Techniques - Common Test Estimation Techniques used in SDLC» [http://www.softwaretestingclass.com/software-estimation-techniques/]

346 The WBS is a deliverable-oriented hierarchical decomposition of the work to be executed by the project team, to accomplish the project objectives and create the required deliverables. The WBS organizes and defines the total scope of the project. The WBS subdivides the project work into smaller, more manageable pieces of work, with each descending level of the WBS representing an increasingly detailed definition of the project work. The planned work contained within the lowest-level WBS components, which are called work packages, can be scheduled, cost estimated, monitored, and controlled. [PMBOK, 3rd edition]

В процессе выполнения структурной декомпозиции большие задачи делятся на всё более и более мелкие подзадачи, что позволяет нам:

- описать весь объём работ с точностью, достаточной для чёткого понимания сути задач, формирования достаточно точной оценки трудозатрат и выработки показателей достижения результатов;
- определить весь объём трудозатрат как сумму трудозатрат по отдельным задачам (с учётом необходимых поправок);
- от интуитивного представления перейти к конкретному перечню отдельных действий, что упрощает построение плана, принятие решений о распараллеливании работ и т.д.

Сейчас мы рассмотрим применение структурной декомпозиции в сочетании с упрощённым взглядом на оценку трудозатрат на основе требований и тест-кейсов.

С подробной теорией по данному вопросу можно ознакомиться в следующих статьях:

- «Test Effort Estimation Using Use Case Points[347]», Suresh Nageswaran.
- «Test Case Point Analysis[348]», Nirav Patel.

Если абстрагироваться от научного подхода и формул, то суть такой оценки сводится к следующим шагам:

- декомпозиции требований до уровня, на котором появляется возможность создания качественных чек-листов;
- декомпозиции задач по тестированию каждого пункта чек-листа до уровня «тестировщицких действий» (создание тест-кейсов, выполнение тест-кейсов, создание отчётов о дефектах и т.д.);
- выполнению оценки с учётом собственной производительности.

Рассмотрим этот подход на примере тестирования требования ДС-2.4{57}: «При указании неверного значения любого из параметров командной строки приложение должно завершить работу, выдав сообщение об использовании (ДС-3.1), а также сообщив имя неверно указанного параметра, его значение и суть ошибки (см. ДС-3.2)».

Это требование само по себе является низкоуровневым и почти не требует декомпозиции, но чтобы проиллюстрировать суть подхода, проведём разделение требования на составляющие:

- Если все три параметра командной строки указаны верно, сообщение об ошибке не выдаётся.
- Если указано неверно от одного до трёх параметров, то выдаётся сообщение об использовании, имя (или имена) неверно указанного параметра и неверное значение, а также сообщение об ошибке:
  - Если неверно указан SOURCE_DIR или DESTINATION_DIR: «Directory not exists or inaccessible».
  - Если DESTINATION_DIR находится в SOURCE_DIR: «Destination dir may not reside within source dir tree».
  - Если неверно указан LOG_FILE_NAME): «Wrong file name or inaccessible path».

[347] «Test Effort Estimation Using Use Case Points», Suresh Nageswaran [http://www.bfpug.com.br/Artigos/UCP/Nageswaran-Test_Effort_Estimation_Using_UCP.pdf]

[348] «Test Case Point Analysis», Nirav Patel [http://www.stickyminds.com/sites/default/files/article/file/2013/XUS373692file1_0.pdf]

Создадим чек-лист и здесь же пропишем **примерное** количество тест-кейсов на каждый пункт из предположения, что мы будем проводить достаточно глубокое тестирование этого требования:

- Все параметры корректные {1 тест-кейс}.
- Несуществующий/некорректный путь для:
  - SOURCE_DIR {3 тест-кейса};
  - DESTINATION_DIR {3 тест-кейса}.
- Недопустимое имя файла LOG_FILE_NAME {3 тест-кейса}.
- Значения SOURCE_DIR и DESTINATION_DIR являются корректными именами существующих каталогов, но DESTINATION_DIR находится внутри SOURCE_DIR {3 тест-кейса}.
- Недопустимые/несуществующие имена объектов ФС указаны в более чем одном параметре {5 тест-кейсов}.
- Значения SOURCE_DIR и DESTINATION_DIR не являются корректными/существующими именами каталогов, и при этом DESTINATION_DIR находится внутри SOURCE_DIR {3 тест-кейса}.

У нас получилось примерно 22 тест-кейса. Также давайте для большей показательности примера предположим, что часть тест-кейсов (например, 5) уже была создана ранее.

Теперь сведём полученные данные в таблицу 2.6.a, где также отразим количество проходов. Этот показатель появляется из соображения, что некоторые тест-кейсы будут находить дефекты, что потребует повторного выполнения тест-кейса для верификации исправления дефекта; в некоторых случаях дефекты будут открыты заново, что потребует повторной верификации. Это относится лишь к части тест-кейсов, потому количество проходов может быть дробным, чтобы оценка была более точной.

Количество проходов для тестирования новой функциональности в общем случае можно грубо оценивать так:

- Простая функциональность: 1–1.5 (не все тесты повторяются).
- Функциональность средней сложности: 2.
- Сложная функциональность: 3–5.

Таблица 2.6.a — Оценка количества создаваемых и выполняемых тест-кейсов

	Создание	**Выполнение**
Количество	12	22
Повторения (проходы)	1	1.2
Общее количество	12	26.4
Время на один тест-кейс		
Итоговое время		

Осталось заполнить ячейки со значениями времени, необходимого на разработку и выполнение одного тест-кейса. К сожалению, не существует никаких магических способов выяснения этих параметров — только накопленный опыт о вашей собственной производительности, на которую среди прочего влияют следующие факторы (по каждому из них можно вводить коэффициенты, уточняющие оценку):

- ваш профессионализм и опыт;
- сложность и объёмность тест-кейсов;
- производительность тестируемого приложения и тестового окружения;
- вид тестирования;
- наличие и удобство средств автоматизации;
- стадия разработки проекта.

Тем не менее существует простой способ получения интегральной оценки вашей собственной производительности, при котором влиянием этих факторов можно пренебречь: нужно измерять свою производительность на протяжении длительного периода времени и фиксировать, сколько создать и выполнить тест-кейсов вы можете в час, день, неделю, месяц и т.д. Чем больший промежуток времени будет рассмотрен, тем меньше на результаты измерений будут влиять кратковременные отвлекающие факторы, появление которых сложно предсказывать.

Допустим, что для некоторого выдуманного тестировщика эти значения оказались следующими — за месяц (28 рабочих дней) ему удаётся:

- Создать 300 тест-кейсов (примерно 11 тест-кейсов в день, или 1.4 в час).
- Выполнить 1000 тест-кейсов (примерно 36 тест-кейсов в день, или 4.5 в час).

Подставим полученные значения в таблицу 2.6.a и получим таблицу 2.6.b.

Таблица 2.6.b — Оценка трудозатрат

	Создание	**Выполнение**
Количество	12	22
Повторения (проходы)	1	1.2
Общее количество	12	26.4
Время на один тест-кейс, ч	0.7	0.2
Итоговое время, ч	8.4	5.2
ИТОГО	**13.6 часа**	

Если бы оценка производительности нашего выдуманного тестировщика производилась на коротких отрезках времени, полученное значение нельзя было бы использовать напрямую, т.к. в нём не было бы учтено время на написание отчётов о дефектах, участие в различных собраниях, переписку и прочие виды деятельности. Но мы потому и ориентировались на итоги измерений за месяц, что за 28 типичных рабочих дней все эти факторы многократно проявляли себя, и их влияние уже учтено в оценке производительности.

Если бы мы всё же опирались на краткосрочные исследования, можно было бы ввести дополнительный коэффициент или использовать допущение о том, что работе с тест-кейсами за один день удаётся посвятить не 8 часов, а меньше (например, 6).

Итого у нас получилось 13.6 часа, или 1.7 рабочих дня. Помня идею о закладке небольшого «буфера», можем считать, что за два полных рабочих дня наш выдуманный тестировщик точно справится с поставленной задачей.

В заключение этой главы ещё раз отметим, что для уточнения собственной производительности и улучшения своих навыков оценки трудозатрат стоит формировать оценку, после чего выполнять работу и сравнивать фактический результат с оценкой. И повторять эту последовательность шагов раз за разом.

Задание 2.6.c: разработайте на основе итогового чек-листа{154}, представленного в разделе 2.4, тест-кейсы и оцените свою производительность в процессе выполнения этой задачи.