ДОКУМЕНТЫ, СОЗДАВАЕМЫЕ ПРИ ТЕСТИРОВАНИИ

В ходе разработки сложной программной системы создается большое количество проектной документации. Ее назначение – координация совместных действий большого количества разработчиков в процессе выполнения работ. Основное назначение тестовой документации, помимо синхронизации действий тестировщиков различных уровней, – обеспечение гарантий того, что тестирование выполняется в соответствии с выбранными критериями качества и, что все аспекты поведения системы протестированы. Кроме того, тестовая документация используется при внесении изменений в систему для проверки того, что старая и новая функциональность работает корректно (Рис.14 [Синицын 2006]).

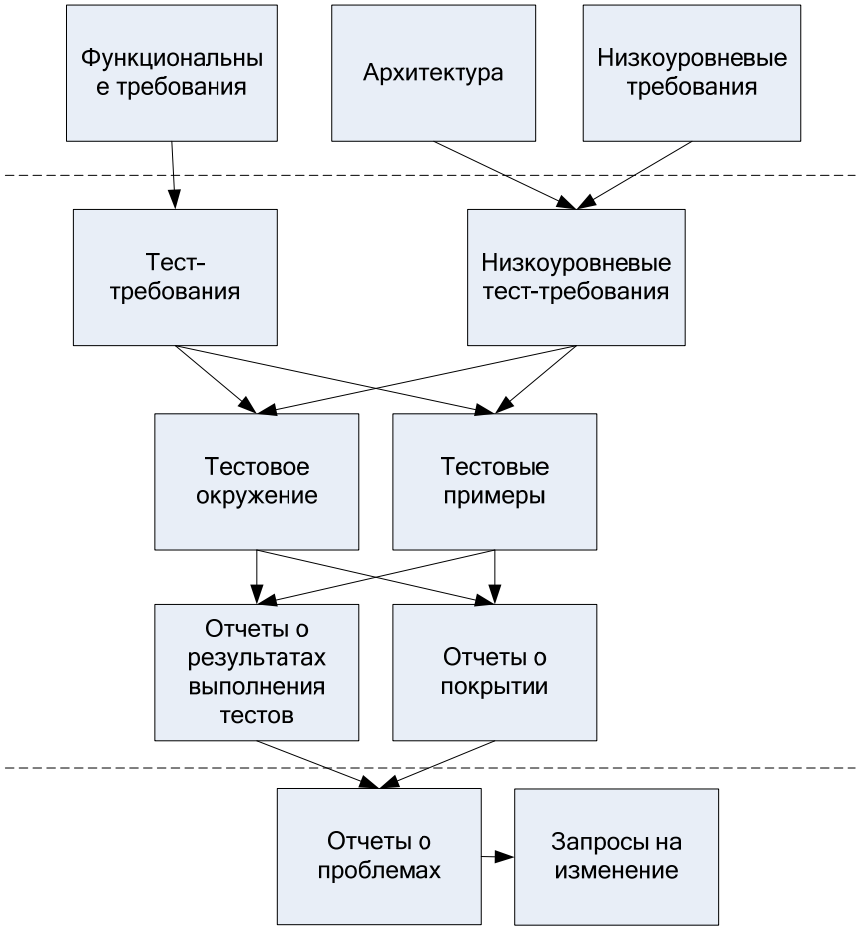

Рис.14 Документация, сопровождающая процесс верификации

ПЛАН ТЕСТИРОВАНИЯ. Перед началом верификации менеджером тестирования (test manager) создается документ, называемый планом тестирования. План тестирования – организационный документ, содержащий требования к тому, как должно выполняться тестирование в данном конкретном проекте.

ОСНОВНАЯ ЗАДАЧА ПЛАНА ТЕСТИРОВАНИЯ, как документа, – определение границ тестирования, подхода к тестированию, требуемых для тестирования ресурсов, плана-графика тестирования.

План тестирования определяет тестируемые элементы и функции системы, задачи, решаемые в ходе тестирования, сотрудников, ответственных за тестирование и риски, связанные с этим планом. В этом документе уделяется много внимания именно тому, как должен быть организован процесс тестирования, а не тому, как тестировать саму систему.

Минимально необходимые элементы, которые рекомендуется включать в каждый план тестирования это:

- ✓ идентификатор плана тестирования и номер его версии;
- ✓ общее описание тест-плана;
- ✓ трассировка на другие документы;
- ✓ определение тестируемых областей системы;
- ✓ определение подходов к тестированию;
- ✓ критерий успешности/неуспешности прохождения тестов (pass/fail criteria);
- ✓ тестовые документы;
- ✓ требования к среде тестирования;
- ✓ людские ресурсы и уровень их подготовки;
- ✓ план-график тестирования;
- ✓ риски.

В данном документе также определяются требования собственно к тестовой документации – *тест-требованиям, тест-планам, отчетам о выполнении тестирования.*

ТЕСТ-ТРЕБОВАНИЯ. Согласно разработанному плану тестирования по системным и функциональным требованиям разработчиками тестов (test procedure developers) создаются тест-требования – документы, в которых описано то, какие аспекты поведения системы должны быть протестированы.

Функциональные требования включают в себя определение моделей поведения системы в штатных и нештатных ситуациях, правила обработки данных и определения интерфейса с пользователем.

Текст функционального требования часто имеет структуру вида:

> *В случае, если значение температуры на датчике ABC достигает 30 и выше градусов Цельсия, система должна прекращать выдачу звукового сигнала.*

На базе функциональных требований пишутся тест-требования – документы, содержащие определение ключевых точек, которые должны быть проверены для того, чтобы убедиться в корректности реализации функциональных требований. Часто тест-требования начинаются словами:

> *Проверить, что*

и содержат ссылки на соответствующие им функциональные требования.

Примером тест-требований для приведенного выше функционального требования могут служить:

> *Проверить, что в случае падения температуры на датчике ABC ниже 30 градусов Цельсия система выдает предупреждающий звуковой сигнал.*
>
> *Проверить, что в случае, когда значение температуры на датчике ABC выше 30 градусов Цельсия, система не выдает звуковой сигнал.*

Одна из проблем, возникающих при написании тест-требований – принципиальная нетестируемость некоторых требований, например требование:

> *Интерфейс пользователя должен быть интуитивно понятным.*

невозможно проверить без четкого определения того, что является интуитивно понятным интерфейсом. Такие неконкретные функциональные требования обычно впоследствии видоизменяют.

На основании описания архитектуры создаются низкоуровневые тест-требования, в которых описываются аспекты поведения конкретной программной реализации системы, которые необходимо протестировать.

Примером такого требования является, например:

> *Проверить, что значение температуры на датчике ABC не выходит за 255*

ТЕСТ-ПЛАНЫ. На основании тест-требований разработчиками тестов (test developers) создаются тест-планы – документы, которые содержат подробное пошаговое описание того, как должны быть протестированы тест-требования.

ТЕСТ-ПЛАН – последовательность тестовых примеров, выполняющих проверку соответствия реализации системы требованиям. Каждый тестовый пример содержит конкретное описание значений, подаваемых на вход системы, значений, которые ожидаются на выходе и описание сценария выполнения теста.

Тест-план готовится либо в виде программы на каком-либо языке программирования, либо в виде входного файла данных для инструментария, выполняющего тестируемую систему и передающего ей значения, указанные в тест-плане, либо в виде инструкций для пользователя системы, описывающей необходимые действия, которые нужно выполнить для проверки различных функций системы.

Дополнительно, на основании тест-требований и проектной документации создается тестовое окружение, необходимое для корректного выполнения тестов на тестовых стендах – драйверы, заглушки, настроечные файлы и т.п.

ОТЧЕТ О ВЫПОЛНЕНИИ ТЕСТИРОВАНИЯ. В результате выполнения всех тестовых примеров собирается статистика об успешности прохождения тестирования – процент тестовых примеров, для которых реальные выходные значения совпали с ожидаемыми, так называемых пройденных тестов. По результатам выполнения тестов тестировщиками (testers) создаются отчеты о выполнении тестирования, которые содержат информацию о том, какие несоответствия требованиям были выявлены в результате тестирования, а также отчеты о покрытии, содержащие информацию о том, какая доля программного кода системы была задействована в результате выполнения тестирования.

ОТЧЕТ О ПРОБЛЕМАХ. Не пройденные тесты являются исходными данными для анализа причин ошибок и последующего их исправления. По найденным несоответствиям создаются отчеты о проблемах – документы, которые направляются на анализ в группу разработчиков с целью определения причины возникновения несоответствия.

ЗАПРОС НА ИЗМЕНЕНИЕ СИСТЕМЫ. Изменения в систему вносятся только после изучения этих отчетов и локализации проблем, вызвавших несоответствие требованиям. Для того, чтобы процесс изменений не вышел из под контроля и любое изменение протоколировалось, создается запрос на изменение системы. После завершения всех работ по запросу на изменение процесс тестирования повторяется до тех пор, пока не будет достигнут приемлемый уровень качества программной системы.

ДОКУМЕНТИРОВАНИЕ ИНДУСТРИАЛЬНОГО ТЕСТИРОВАНИЯ

ДОКУМЕНТАЦИЯ И СОПРОВОЖДЕНИЕ ТЕСТОВ

ТЕСТОВЫЕ ПРОЦЕДУРЫ – это формальный документ, содержащий описание необходимых шагов для выполнения тестового набора. Процедуры должны быть составлены таким образом, чтобы любой инженер, не связанный с данным проектом, был способен адекватно провести цикл тестирования, обладая только самыми базовыми знаниями о применяющемся инструментарии.

ОПИСАНИЕ ТЕСТОВ разрабатывается для облегчения анализа и поддержки тестового набора и выполняет следующие задачи.

- ✓ Анализировать степень покрытия продукта тестами на основании описания тестового набора.
- ✓ Для любой функции тестируемого продукта найти тесты, в которых функция используется.
- ✓ Для любого теста определить все функции и их сочетания, которые данный тест использует (затрагивает).
- ✓ Понять структуру и взаимосвязи тестовых файлов.
- ✓ Понять принцип построения системы автоматизации тестирования.

ДОКУМЕНТИРОВАНИЕ И ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА

Каждый дефект (ошибка), обнаруженный в процессе тестирования, должен быть документирован и отслежен. При обнаружении нового дефекта его заносят в базу дефектов. Для этого лучше всего использовать специализированные базы, поддерживающие хранение и отслеживание дефектов - типа DDTS. При занесении нового дефекта рекомендуется указывать, как минимум, следующую информацию:

- ✓ Наименование подсистемы, в которой обнаружен дефект.
- ✓ Версия продукта (номер build), на котором дефект был найден.
- ✓ Описание дефекта.
- ✓ Описание процедуры (шагов, необходимых для воспроизведения дефекта).
- ✓ Номер теста, на котором дефект был обнаружен.
- ✓ Уровень дефекта, то есть степень его серьезности с точки зрения критериев качества продукта или заказчика.

Занесенный в базу дефектов новый дефект находится в состоянии "New". После того, как команда разработчиков проанализирует дефект, он переводится в состояние "Open" с указанием конкретного разработчика, ответственного за исправление дефекта. После исправления дефект переводится разработчиком в состояние "Resolved".

При этом разработчик должен указать следующую информацию:

- ✓ Причину возникновения дефекта.
- ✓ Место исправления, как минимум, с точностью до исправленного файла.
- ✓ Краткое описание того, что было исправлено.
- ✓ Время, затраченное на исправление.

После этого тестировщик проверяет, действительно ли дефект был исправлен и если это так, переводит его в состояние "Verified". Если тестировщик не подтвердит факт исправления дефекта, то состояние дефекта изменяется снова на "Open".

Если проектная команда принимает решение о том, что некоторый дефект исправляться не будет, то такой дефект переводится в состояние "Postponed" с указанием лиц, ответственных за это решение, и причин его принятия.

ТЕСТОВЫЙ ОТЧЕТ

Тестовый отчет обновляется после каждого цикла тестирования и должен содержать следующую информацию для каждого цикла:

- ✓ Перечень функциональности в соответствии с пунктами требований, запланированный для тестирования на данном цикле, и реальные данные по нему.
- ✓ Количество выполненных тестов – запланированное и реально исполненное.

- ✓ Время, затраченное на тестирование каждой функции, и общее время тестирования.
- ✓ Количество найденных дефектов.
- ✓ Количество повторно открытых дефектов.
- ✓ Отклонения от запланированной последовательности действий, если таковые имели место.
- ✓ Выводы о необходимых корректировках в системе тестов, которые должны быть сделаны до следующего тестового цикла.

ТЕСТОВЫЕ МЕТРИКИ

Существует устоявшийся набор тестовых метрик, который помогает определить эффективность тестирования (качество) и текущее состояние продукта. К таким метрикам относятся следующие.

- ✓ Покрытие функциональных требований.
- ✓ Покрытие кода продукта. Наиболее применимо для модульного уровня тестирования.
- ✓ Покрытие множества сценариев.
- ✓ Количество или плотность найденных дефектов. Текущее количество дефектов сравнивается со средним для данного типа продуктов с целью установить, находится ли оно в пределах допустимого статистического отклонения.
- ✓ Соотношение количества найденных дефектов с количеством тестов на данную функцию продукта.
- ✓ Количество найденных дефектов, соотнесенное по времени, или скорость поиска дефектов.

ОБЗОРЫ ТЕСТОВ И СТРАТЕГИИ

Цели обзора тестовой стратегии:

1. Установить достаточность проверок, обеспечиваемых тестированием.
2. Проанализировать оптимальность покрытия или адекватность распределения количества планируемых тестов по функциональности продукта.
3. Проанализировать оптимальность подхода к разработке кода, генерации кода, автоматизации тестирования.

Цели обзора тестового кода:

1. Установить соответствие тестового набора тестовой стратегии.
2. Проверить правильность кодирования тестов.
3. Оценить степень качества кода, которая достигнута, исходя из требований по стандартам, простоте поддержки, наличию комментариев и т.п.
4. Если необходимо, проанализировать оптимальность тестового кода с целью удовлетворения требований к быстродействию и объему.

РУЧНОЕ ТЕСТИРОВАНИЕ

Ручное тестирование заключается в выполнении документированной процедуры, где описана методика выполнения тестов, задающая порядок тестов и для каждого теста - список значений параметров, который подается на вход, и список результатов, ожидаемых на выходе. Поскольку процедура предназначена для выполнения человеком, в ее описание для краткости могут использоваться некоторые значения по умолчанию, ориентированные на здравый смысл, или ссылки на информацию, хранящуюся в другом документе.

ПРИМЕР ФРАГМЕНТА ПРОЦЕДУРЫ ПРИ РУЧНОМ ТЕСТИРОВАНИИ

1. Подать на вход три разных целых числа.

2. Запустить тестовое исполнение.
3. Проверить, соответствует ли полученный результат таблице [ссылка на документ_1] с учетом поправок [ссылка на документ_2].
4. Убедиться в понятности и корректности выдаваемой сопроводительной информации.

В приведенной процедуре тестировщик использует два дополнительных документа, а также собственное понимание того, какую сопроводительную информацию считать "понятной и корректной". Успех от использования процедурного подхода достигается в случае однозначного понимания тестировщиком всех пунктов процедуры. Например, в п.1 приведенной процедуры не уточняется, из какого диапазона должны быть заданы три целых числа, и не описывается дополнительно, какие числа считаются "разными".

ПРИМЕР СКРИПТА

1. Выдать на консоль имя или номер теста и время его начала.
2. Вызвать продукт с фиксированными параметрами.
3. Перенаправить вывод продукта в файл.
4. Проверить возвращенное продуктом значение. Оно должно быть равно ожидаемому (эталонному) результату, зафиксированному в тесте.
5. Проверить вывод продукта, сохраненный в файле (п.3), на равенство заранее приготовленному эталону.
6. Выдать на консоль результаты теста в виде вердикта PASS/FAIL и в случае FAIL - краткого пояснения, какая именно проверка не прошла.
7. Выдать на консоль время окончания теста.

Сравнение ручного и автоматизированного подхода (Табл.10.1 [Котляров 2006]) показывает тенденцию современного тестирования, ориентирующую на максимальную автоматизацию процесса тестирования и генерацию тестового кода, что позволяет справляться с большими объемами данных и тестов, необходимых для обеспечения качества при производстве программных продуктов.

Таблица 10.1. Сравнение ручного и автоматизированного подхода

	Ручное	**Автоматизированное**
Задание входных значений	Гибкость в задании данных. Позволяет использовать разные значения на разных циклах прогона тестов, расширяя покрытие	Входные значения строго заданы
Проверка результата	Гибкая, позволяет тестировщику оценивать нечетко сформулированные критерии	Строгая. Нечетко сформулированные критерии могут быть проверены только путем сравнения с эталоном
Повторяемость	Низкая. Человеческий фактор и нечеткое определение данных приводят к неповторяемости тестирования	Высокая
Надежность	Низкая. Длительные тестовые циклы приводят к снижению внимания тестировщика	Высокая, не зависит от длины тестового цикла
Чувствительность к незначительным изменениям в продукте	Зависит от детальности описания процедуры. Обычно тестировщик в состоянии выполнить тест, если внешний вид продукта и текст сообщений несколько изменились	Высокая. Незначительные изменения в интерфейсе часто ведут к коррекции эталонов
Скорость выполнения тестового набора	Низкая	Высокая
Возможность генерации тестов	Отсутствует. Низкая скорость выполнения обычно не позволяет исполнить сгенерированный набор тестов	Поддерживается

ЛИТЕРАТУРА

[Синицын 2006] – Синицын С.В., Налютин Н.Ю. Верификация программного обеспечения. Курс лекций. Московский инженерно-физический институт. М. 2006.

[Котляров 2006] – В.П. Котляров, Т.В. Коликова. Основы тестирования программного обеспечения. Учебное пособие. М.: Интернет-Университет Информационных технологий.