МОДУЛЬ 4: «ТЕСТОВЫЕ ТРЕБОВАНИЯ»

ТЕСТ-ТРЕБОВАНИЯ – основной документ для тестировщика, который определяет функциональность системы с точки зрения того, что должно быть проверено для того, чтобы удостовериться в ее корректном функционировании, а также – на основании какого внешнего эффекта можно убедиться, что проверяемая функция реализована правильно. Существует два подхода к написанию тест-требований – **ФУНКЦИОНАЛЬНЫЙ И СТРУКТУРНЫЙ.**

Тест-требования, написанные в рамках *функционального подхода* основываются на системных требованиях и требованиях к программному обеспечению системы.

Тест-требования, написанные в рамках *структурного подхода* пишутся на основании описания архитектуры системы и принимают в расчет строение исходных текстов системы. Структурные тест-требования важны в том случае, когда к надежности системы предъявляются повышенные требования. Т.е. когда важно проверить не только насколько корректно система в целом отрабатывает сценарии своей работы, но и как в различных нестандартных ситуациях будут вести себя отдельные ее компоненты.

Из-за такого различия функциональный и структурный подходы часто называют *подходами черного и белого ящиков.*

На практике почти всегда применяются оба подхода к разработке тест-требований, в результате в состав документации проекта включаются тест-требования верхнего уровня и тест-требования нижнего уровня, по которым составляются тест-планы (Рис.15 [Синицын 2006]).

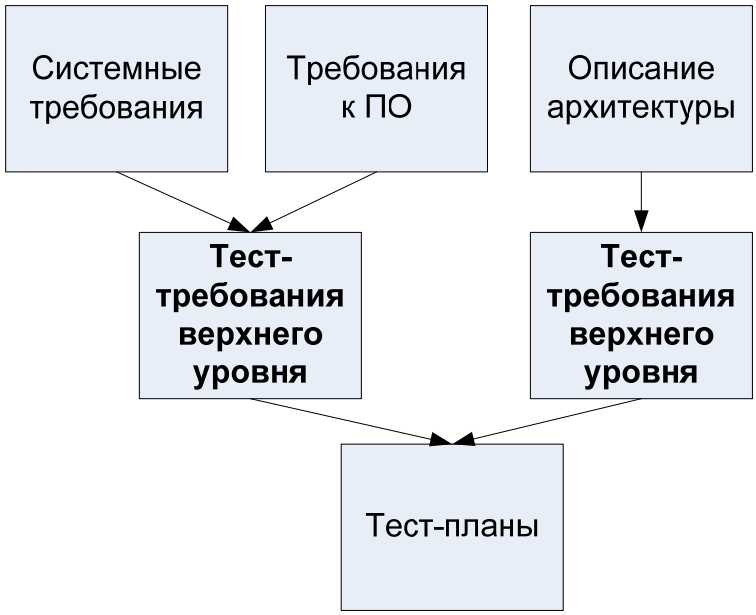

Рис.15 Место тест-требований среди проектной документации

СВОЙСТВА ТЕСТ-ТРЕБОВАНИЙ

Тест-требования должны быть достаточными для построения тест-плана проверки реализации задачи без знакомства с ее программными текстами, т.е. тест-требования должны обладать свойством изоляции от внутренней структуры системы.

Структура тест-требований следует структуре раздела функциональных требований на систему. Задача каждого требования - определение того, что надо проверить. Техника исполнения каждой такой проверки – задача тест-плана. Обычный формат описания отдельного требования следующий:

> *Проверить, что при <описание внешнего воздействия> [происходит] <описание реакции программы>.*

Тест-требования, написанные в рамках функционального подхода обычно разделяют на следующие группы:

- ✓ функции контроля входных данных;
- ✓ функции обработки ошибок (ввода, вычислений);
- ✓ функции получения основного результата;
- ✓ функции обработки особых ситуаций;
- ✓ функции оформления и вывода результатов.

Совокупность тест-требований должна обладать некоторыми важными свойствами: **ПОЛНОТА, ВЕРИФИЦИРУЕМОСТЬ И НЕПРОТИВОРЕЧИВОСТЬ.**

Как правило, одному системному или функциональному требованию соответствует минимум одно тест-требование. Если совокупность проверок, задаваемых тест-требованиями, покрывает всю функциональность системы, определенную в системных требования и требованиях к программному обеспечению, то говорят о полноте тест-требований. При изменениях требований к системе для поддержания полноты должны меняться и тест-требования.

Как системные требования и требования к программному обеспечению, так и тест-требования должны обладать свойством верифицируемости. Т.е. для каждого требования должна существовать возможность определить четкий критерий проверки – выполняется это требование в реализованной системе или нет.

Примером не верифицируемого требования может служить следующее “тест-требование”:

> *Система должна иметь интуитивно понятный пользовательский интерфейс.*

Очевидное, что “тест-требование” будет выглядеть как:

> *Проверить, что система имеет интуитивно понятный пользовательский интерфейс.*

Без четкого определения критериев интуитивной понятности, проверить такое требование при помощи написания тестовых примеров не представляется возможным. Однако, если сопроводить такое требование количественными или качественными характеристиками интуитивно понятного интерфейса – написание тестовых примеров по требованиям становится возможным. Так, среди критериев интуитивной понятность могут быть следующие: глубина вложенности меню не более трех, наличие всплывающих подсказок на каждом элементе управления каждой экранной формы и т.п.

При большом количестве тест-требований и частых их изменениях может возникнуть ситуация, в которой различные требования перестают быть согласованными. В этом случае такие требования имеют взаимоисключающие друг друга критерии проверки.

Например, одно тест-требование на пользовательский интерфейс может декларировать необходимость проверки того, что введенный пользователем пароль имеет длину не более 16 символов, а тест-требование к базе данных системы – что допустимый размер пароля, сохраняемого в БД – от 4 до 12 символов. В этом случае эти два требования являются противоречивыми. Для того, чтобы устранить это противоречие нужно проводить анализ системных и функциональных требований с последующей модификацией тест-требований.

Тест-требования, по которым составляются тест-планы для тестирования системы, обычно обладают свойством непротиворечивости, поскольку противоречия обычно устраняются на уровне верификации проектной документации. Однако противоречия могут быть выявлены и позже, в результате попытки создать адекватные тестовые примеры.

ПРИМЕР ИСПОЛЬЗОВАНИЯ СПЕЦИФИКАЦИИ ТРЕБОВАНИЙ ДЛЯ РАЗРАБОТКИ ТЕСТОВ

Пусть задан следующий фрагмент набора требований для модели обмена транзакциями:

1. Функция *DoTransaction* должна принимать адрес и данные в соответствии с параметрами, создавать в очереди новый элемент, заполнять его адресную часть и часть полей данных переданной информацией и инициировать транзакцию.
2. Функция *DoAddressTenure* должна принимать адрес в соответствии с параметрами, создавать в очереди новый элемент и заполнять его адресную часть.
3. Функция *DoDataTenure* должна принимать данные в соответствии с параметрами, находить в очереди первый элемент с частично незаполненными полями данных, дополнять его переданной информацией и инициировать транзакцию.

Концептуальное описание набора тестов, проверяющего спецификацию, может выглядеть следующим образом:

1. Вызвать *DoTransaction* с адресом и данными. Проверить появление в очереди еще одного элемента. Проверить появление на шине транзакции с правильными адресом и данными.
2. Вызвать *DoAddressTenure* с адресом. Проверить появление в очереди еще одного элемента. Проверить отсутствие новой транзакции на шине.
3. Вызвать *DoDataTenure* с данными. Проверить заполнение полей данных. Проверить появление на шине транзакции с правильными адресом и данными.

ЛИТЕРАТУРА

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

АНАЛИЗ ТЕСТОВЫХ ТРЕБОВАНИЙ НА ДУХОВОЙ ШКАФ

ВВЕДЕНИЕ

Работа тестера состоит в том, чтобы:

- ✓ узнать о новом приложении;
- ✓ перечислить возможности тестового примера;
- ✓ определить тестовый пример;
- ✓ выбрать, какие будут проводиться тесты;
- ✓ создать среду тестирования;
- ✓ провести тесты.

ПРИМЕР ПРИЛОЖЕНИЯ

Примером приложения служит контроллер духового шкафа, его функции кратко описаны на рис.2.1 [Тампе 2006].

Краткое описание приложения

Спроектировать контроллер для электрического духового шкафа, который позволит пользователю выбирать нужную температуру приготовления и задавать время. Духовой шкаф будет разогреваться и выключаться согласно введенной пользователем информации.

Рис. 2.1. Описание приложения

На рис.2.2 [Тампе 2006] представлена диаграмма контекста, на которой показаны входные и выходные данные для контроллера духового шкафа.

Рис. 2.2. Диаграмма контекста контроллера духового шкафа

Задача тестера – проанализировать требования с последующим определением набора тестов.

В дополнение к перечисленной выше информации о продукте предоставлен набор так называемых “требований”, поскольку их содержимое не описывает реальных системных требований (рис.2.3 [Тампе 2006]).

Контроллер духового шкафа будет работать при частоте 30 МГц (или выше) и контролировать температуру в диапазоне от 200° Ф до 500° Ф, по умолчанию задается 350° Ф и значения температуры будут кратны 5, что определяется пользователем на клавишной панели. Управление автоматическое или ручное, а введенная пользователем температура поддерживается с точностью до 2° Ф. В конце выпечки в духовом шкафу выключается нагревательный элемент и в автоматическом режиме подается звуковой сигнал. Программа духового шкафа начинает работать, когда пользователь дает команду старта; в будущих выпусках она будет управлять приготовлением в режиме гриль и в микроволновом режиме. В автоматическом режиме пользователь задает время начала и окончания работы. Нагрев начинается тогда, когда наступает время начала работы и заканчивается, когда время истекает или когда пользователь дает команду окончить работу; позже это будет применимо как для автоматического, так и для ручного режимов. Часы отображают текущее время и отслеживают истекшее время. Ручной режим включается тогда, когда пользователь вводит команду начала работы, и заканчивается при введении команды ее окончания.

Рис. 2.3. Предъявленные требования

ВЫДЕЛЕНИЕ ТРЕБОВАНИЙ

Начинать процесс тестирования следует с преобразования обременительного, сбивающего с толку абзаца в список из отдельных предложений или фрагментов, как это показано на рис.2.4 [Тампе 2006].

Т-1	Духовой шкаф будет регулировать температуру в диапазоне от 200° Ф до 500° Ф с шагом 5° Ф.
Т-2	Контроллер духового шкафа будет поддерживать заданную температуру с точностью до ±2° Ф.
Т-3	Духовой шкаф будет работать в ручном или автоматическом режиме.
Т-4	Пользователь будет определять температуру приготовления.
Т-5	По умолчанию температура приготовления будет составлять 350° Ф.
Т-6	В автоматическом режиме пользователь будет задавать время начала и время окончания работы.
Т-7	Автоматический режим начинает работать, когда пользователь выдаст команду начинать.
Т-8	Нагрев в ручном режиме будет начинаться, когда пользователь введет команду начала работы.
Т-9	Нагрев в автоматическом режиме будет начинаться в указанное время начала.
Т-10	Ручной режим будет выключаться, когда пользователь выполнит команду окончания.
Т-11	Автоматический режим будет выключаться, когда истечет время или когда пользователь выполнит команду окончания.
Т-12	При нагреве в духовом шкафу будет выключаться нагревательный элемент.
Т-13	При автоматическом выключении духовой шкаф будет подавать звуковой сигнал.
Т-14	Контроллер духового шкафа будет в состоянии управлять приготовлением в режиме гриль и микроволновом режиме в будущей версии.
Т-15	Быстродействие процессора будет составлять по меньшей мере, 30 МГц.
Т-16	У пользовательского интерфейса будет буквенно-цифровой дисплей.
Т-17	Интерфейс ввода пользователем будет иметь вид клавишной панели.
Т-18	Температура духового шкафа и сообщения отсылаются на буквенно-цифровой дисплей.
Т-19	Часы будут показывать текущее время и будут в состоянии контролировать истекшее время.

Рис. 2.4. Описание приложения с разбиением на пункты

Следующий этап заключается в разбиении требований на три типа: входные состояния, ожидаемые результаты и проектные ограничения. Входная информация используется при разработке теста, выходная информация будет использоваться позже при определении тестовых примеров, а проектные ограничения в тесты не включаются.

ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ

Это общий термин, описывающий результирующее поведение системы, включая внутреннее состояние или изменение памяти, которые пользователю наблюдать сложно, в дополнение к образованию отдельных значений или набора результирующих данных. Помимо значений выходных данных при выполнении тестов часто необходимо оценивать и другие трансформации. На рис.2.4 [Тампе 2006] перечислены пять требований, описывающие результаты; эти положения будут исключены из списка входных состояний.

Т-2	Контроллер духового шкафа будет поддерживать заданную температуру с точностью до ±2° Ф.
Т-9	Нагрев в автоматическом режиме будет начинаться в указанное время.
Т-12	При нагреве в духовом шкафу будет выключаться нагревательный элемент.
Т-13	При автоматическом выключении духовой шкаф будет подавать звуковой сигнал.
Т-18	Температура духового шкафа и сообщения отсылаются на буквенно-цифровой дисплей.

ПРОЕКТНЫЕ ОГРАНИЧЕНИЯ

Хотя проектные ограничения иногда и маскируются под требования, тестеры за них не отвечают. Проектные ограничения накладываются проектом системы, а не предполагаемым ее поведением. Часто требования могут предусматривать некоторые будущие возможности или свойства системы, играя роль связующего звена с функциями, которые еще не разработаны. Цель таких указаний – позволить инженерам, соответствующим образом адаптировать свои проекты.

Пять элементов из перечисленных на рис.2.4 [Тампе 2006] – это проектные ограничения, которые лучше всего проверять в ходе пересмотра проекта системы. В противном случае в продукте могут быть построены неверные компоненты. Таким образом, в списке входных состояний будут пропущены следующие компоненты.

T-14	В будущей версии контроллер духового шкафа сможет управлять приготовлением в режиме гриль и микроволновом режиме.
T-15	Быстродействие процессора будет составлять по меньшей мере, 30 МГц.
T-16	У пользовательского интерфейса будет буквенно-цифровой дисплей.
T-17	Интерфейс ввода пользователем будет иметь вид клавишной панели.
T-19	Часы будут показывать текущее время и контролировать истекшее время.

СХЕМАТИЧЕСКИЙ ПОДХОД

Схематический подход – это метод, который используется для рассмотрения требований. Требования преобразуются в схемы для того, чтобы перечислить различные условия теста. Схематический метод тестирования управляется процессами и не завит от содержания. Реально он поможет обнаружить отсутствующую в требованиях информацию. Каждый элемент в схеме, в конечном счете, будет определять входные состояния для тестового примера.

Схема, как показано на рис.2.5 [Тампе 2006], представляет собой древовидную структуру, в которой между корнем и каждым листком существует однозначно определенный путь. Этот путь устанавливает специфический набор входных состояний, которые затем будут использованы для определения тестового примера. Схема содержит узловые точки, или точки ветвления, обозначающие подсказки, тип значения или входной вариант. Нумерация узла обозначает специфическое входное состояние соответствующего ему предка. Число листьев на дереве (или путей в схеме) дает приблизительное число тестовых примеров, необходимых для тестирования всех функций. В зависимости от структуры схемы можно иметь точное соответствие между каждым путем в схеме и связанным с ним тестовым примером.

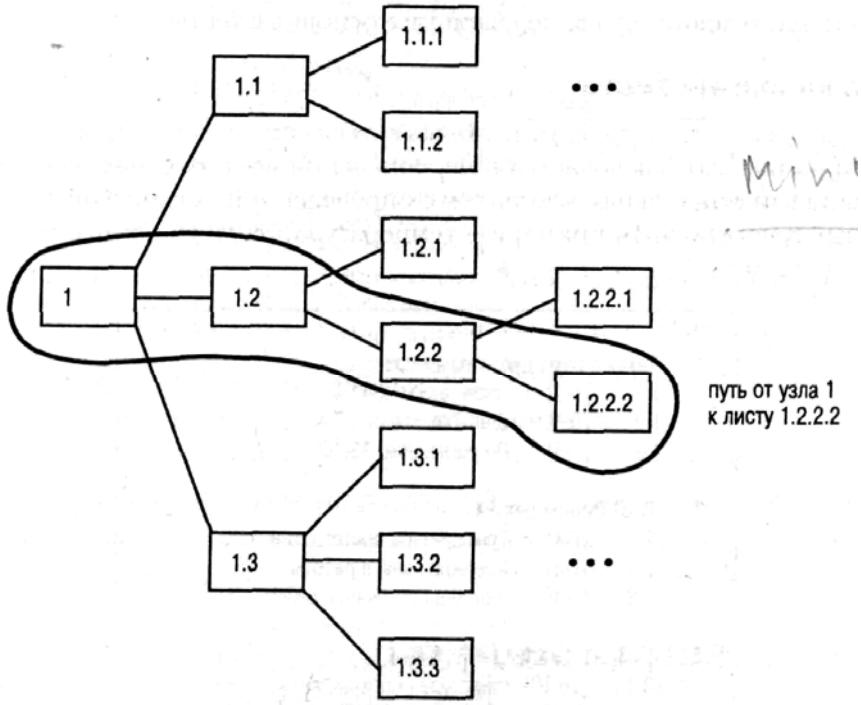

Рис. 2.5. Путь в схеме

Создание схемы – это итерационный процесс. Начальный список образуется на основе требований, которые были выделены в категорию входных состояний. Номер требования включен для того, чтобы облегчить оперативный контроль. Каждая последующая схема является усовершенствованием предыдущей. Число итераций, необходимых для создания окончательной версии, зависит от опыта и от проекта. Каждый специалист продолжает делать свою работу. Последнюю итерацию тестеры выполняют совместно с остальными разработчиками, поскольку именно из этой последней версии будет выводиться тестовый пример.

Некоторым тестерам схематический подход может оказаться интуитивным, а другим – утомительным. Преимущества схематического подхода заключается в том, что он служит инструментом для дискуссий и помогает получить необходимую информацию от команды разработчиков. Это инструмент организации мышления. Схема выделяет основную часть в документации с требованиями, отфильтровывая лишние словесные выражения.

РАЗРАБОТКА СХЕМЫ ТЕСТА

При создании схемы теста будем использовать список с отступами. Первый этап, показанный на рис.2.6 [Тампе 2006], заключается в перечислении всех входных описаний из списка требований как отдельных элементов с определенной логической классификацией. Исходные группы в этом примере – температура духового шкафа, ручной и автоматический режимы.

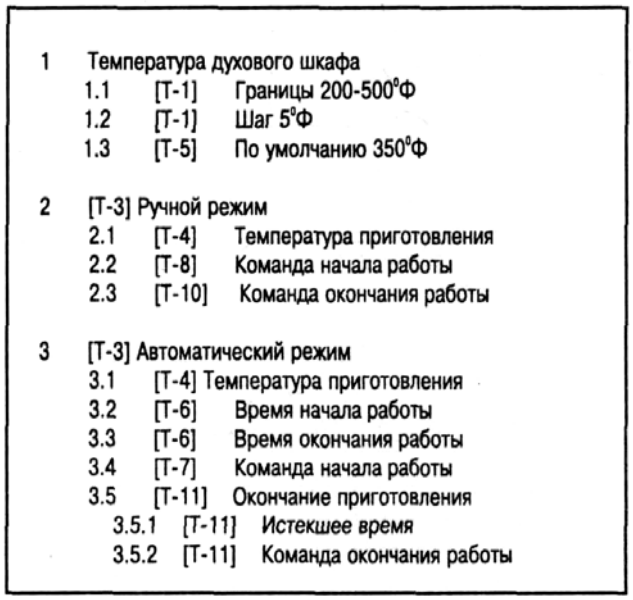

Рис. 2.6. Итерация схемы № 1

Для создания итерации схемы №2 элемент температура духового шкафа (с п.1.1 по п.1.3 на рис.2.7 [Тампе 2006]) будет перемещен в область дополнительного описания температуры приготовления в ручном и автоматическом режимах.

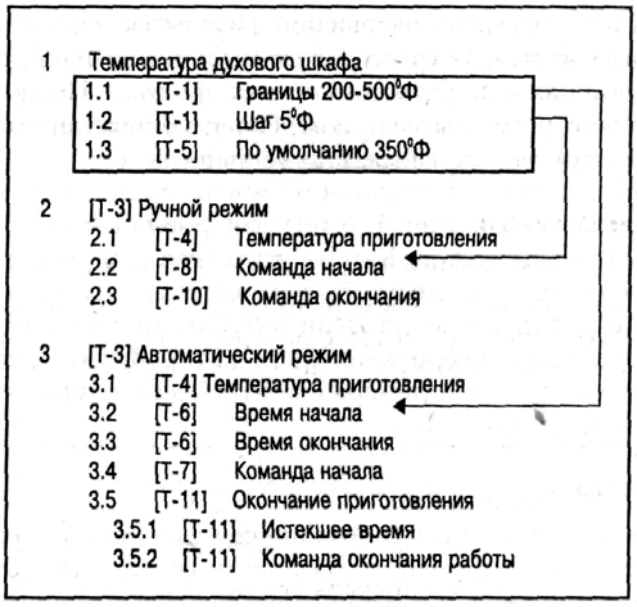

Рис. 2.7. Переход от итерации схемы № 1 к итерации схемы № 2

Зачем это делается? Пользователь работает с духовым шкафом в одном из двух режимов: автоматическом или ручном. Температура может вести себя по-разному в зависимости от того, какой используется режим. Поэтому в схеме пути должны разделяться, чтобы можно было различать приготовления в ручном и в автоматическом режимах. На рис.2.8 [Тампе 2006] показан модифицированная схема с выдвинутым на первый план влиянием элементов.

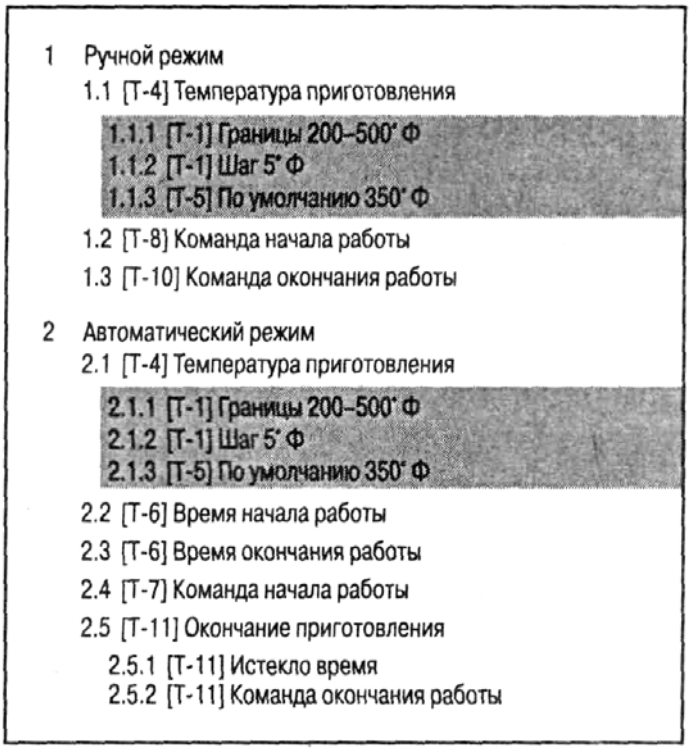

Рис. 2.8. Итерация схемы № 2

Итерация схемы №2 по-прежнему содержит информацию из исходных требований. Преимущества схематического подхода заключается в предоставлении иерархической структуры, в которой аналогичные функции сгруппированы вместе. Разные тестеры могут создавать совершенно различные схемы, однако конечным критерием будет согласие между структурой схемы и поведением приложения так, как это описано в требованиях или других описаниях продукта. Несмотря на высокую вероятность утраты некоторого смысла при переходе в описании, главная задача состоит в определении основных функций, подлежащих тестированию.

ЛИТЕРАТУРА

[Тампе 2003] – Тампе Л. Введение в тестирование программного обеспечения.: Пер. с англ. – М.: Издательский дом “Вильямс”, 2003.