МОДУЛЬ 6: «ТЕСТОВЫЕ ПЛАНЫ (TEST PLAN)»

ТЕСТОВЫЕ ПЛАНЫ

Тестовые примеры не существуют сами по себе – каждый тестовый пример проверяет одну ситуацию в работе системы, но вся совокупность тестовых примеров должна полностью проверять всю функциональность системы. Поэтому описания всех тестовых примеров объединяются в документы, которые называются **ТЕСТ-ПЛАНАМИ.**

Тест-план представляет собой документ, в котором перечислены все тестовые примеры, необходимые для тестирования системы, либо часть тестовых примеров, объединенных по определенному признаку.

Существует несколько причин для объединения описаний тестовых примеров в единый документ или несколько документов (**ТЕСТ-ПЛАН**).

1. Единая схема идентификации и трассировки тестовых примеров.
2. Объединение тестовых примеров в смысловые группы.
3. Внесение изменений в тестовые примеры.
4. Определение последовательности тестирования.

Тест-планы составляются на основании тест-требований. В отличие от тест-требований в тест-плане описываются конкретные способы проверки функциональности системы, т.е. **КАК ДОЛЖНА ПРОВЕРЯТЬСЯ ФУНКЦИОНАЛЬНОСТЬ**. Тест-план состоит из отдельных тестовых примеров, каждый из которых проверяет некоторую функцию или набор функций системы. Для каждого тестового примера однозначно определяется критерий успешного прохождения (pass/fail criteria), при помощи которого можно судить о том – соответствует ли поведение системы заданному в требованиях поведению или нет (Рис.16 [Синицын 2006]).

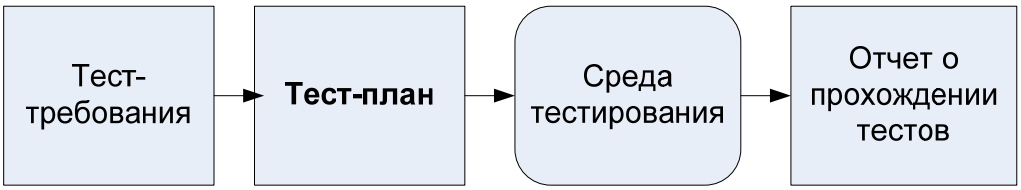

Рис.16 Место тест-планов среди проектной документации

Критерием качества тест-плана является покрытие (выполнение) всех требований к проверке правильности функционирования программы. Желательной характеристикой тест-плана является проверка исполнения всех веток схемы программной реализации.

Структура тест-плана может соответствовать структуре тест-требований или следовать логике внешнего поведения системы. Каждый пункт тест-плана описывает, **КАК ПРОИЗВОДИТСЯ ПРОВЕРКА ПРАВИЛЬНОСТИ ФУНКЦИОНИРОВАНИЯ** программной реализации, и содержит:

- ✓ ссылку на требование(я), которое проверяется этим пунктом;
- ✓ конкретное входное воздействие на программу (значения входных данных);
- ✓ ожидаемую реакцию программы (тексты сообщений, значения результатов);
- ✓ описание последовательности действий, необходимых для выполнения пунктов тест-плана.

В состав тест-плана рекомендуется включать пункты, служащие для проверки ветвей программы, не выполнявшихся при проверке удовлетворения функциональных требований. Такие пункты тест-плана могут иметь указание “Для полноты покрытия” в поле ссылки.

Тест-план может быть написан на естественном или формальном языке, в последнем случае возможна передача тест-плана на вход тестового окружения для автоматического выполнения определенных в тест-плане тестовых примеров. При приготовлении тест-план в виде текстового документа возможно только ручное тестирование системы по написанному тест-плану.

ТИПОВАЯ СТРУКТУРА ТЕСТОВОГО ПЛАНА

Рассмотрим типовую структуру тест-плана, написанного на естественном языке и содержащего тестовые примеры для проверки работы модуля расчета контрольных сумм [Синицын 2006].

> *Тест-план*
>
> *Тестовый пример 1*
>
> *Номер тест-требования: 2a, 2b*
>
> *Описание теста: В данном тесте проверяется правильность вычисления значения контрольной суммы (поля CRC) при непустом значении поля CRC и нулевых значениях элементов записи.*
>
> *Входные данные: CRC = 12345, A=0, B=0, C=0, D=0*
>
> *Ожидаемые выходные данные: CRC = 0, A=0, B=0, C=0, D=0, Empty = TRUE*
>
> *Сценарий теста:*
>
> *Установка значения поля CRC в 12345*
>
> *Установка значений полей A-F в 0*
>
> *Вызов функции Set_CRC*
>
> *Проверка значений CRC на 0 и Empty на TRUE*
>
> *Тестовый пример 2*
>
> *Номер тест-требования: 2a*
>
> *Описание теста: В данном тесте проверяется соответствие алгоритма вычисления поля CRC, заданного в спецификации требований.*
>
> *Входные данные: CRC = 0, A-D заполнены байтами 01010101b*
>
> *Ожидаемые выходные данные: CRC = 0111100b, Empty = FALSE*
>
> *Сценарий теста:*
>
> *Установка значения поля CRC в 0*
>
> *Заполнение байт полей A-D байтами 01010101b*
>
> *Вызов функции Set_CRC*
>
> *Проверка значений CRC на 0111100b и Empty на FALSE*

Тестовый пример 3

Номер тест-требования: 2a

Описание теста: В данном тесте проверяется неизменность полей A-F записи при вычислении поля CRC (подсчете контрольной суммы).

Входные данные: CRC = 0, A-D заполнены байтами 01010101b

Ожидаемые выходные данные: A-D заполнены байтами 01010101b

Сценарий теста:

Установка значения поля CRC в 0

Заполнение байт полей A-D байтами 01010101b

Вызов функции Set_CRC

Проверка значений байт полей A-D на 01010101b

Каждый тестовый пример в этом тест-плане имеет уникальный номер и ссылку на тест-требование, на основе которого он написан. Общее описание теста помогает при сопровождении тест-планов – внесении изменений при изменении системы, инспекциях тест-планов, выявляющих несогласованность и т.п.

В каждом тестовом примере перечислены все входные значения и ожидаемые выходные значения, а также сценарий, описывающий последовательность действий, которые необходимо выполнить тестовому окружению для выполнения тестового примера.

Такая структура тест-плана позволяет описывать тестовые примеры с совершенно различными наборами входных и выходных данных и сценариями, однако при большом количестве тестовых примеров эта схема слишком громоздка.

ВОЗМОЖНЫЕ ФОРМЫ ПОДГОТОВКИ ТЕСТ-ПЛАНОВ

Форма представления тест-плана в первую очередь зависит от того, каким образом тест-план будет использоваться в процессе тестирования.

При ручном тестировании удобно представление тест-планов в виде текстовых документов, в которых отдельные разделы представляют собой описания тестовых примеров. Каждый тестовый пример в таком случае включает в себя перечисление последовательности действий, которые необходимо выполнить тестировщику для проведения тестирования – сценария теста, а также ожидаемые отклики системы на эти действия.

Для автоматизированного тестирования сценарий теста должен быть записан на каком-либо формальном языке, в этом случае возможно непосредственное использование тест-планов как входных данных для среды тестирования.

Другой формой представления тест-планов является таблица. Эта форма часто используется при четко и формально определенных входных потоках данных системы. Например, каждый столбец таблицы может представлять собой тестовый пример, каждая строка – описание входного потока данных, а в ячейке таблицы записывается передаваемое в данном тестовом примере в данный поток значение. Ожидаемые значения для данного теста записываются в аналогичной таблице, в которой в строках перечисляются выходные потоки данных.

Третьей формой представления тест-плана является определение примеров в виде конечного автомата. Эта форма представления используется при тестировании протоколов связи или при тестировании программных модулей, взаимодействие которых с внешним миром производится при помощи обмена сообщениями по заданному интерфейсу. Модуль при этом может быть представлен как конечный автомат с набором состояний, а тест-план будет состоять из двух частей – описания переходов между состояниями и их параметров и тестовых примеров, в которых задается маршрут перехода между состояниями, параметры переходов и ожидаемые значения. Такое представление тест-плана может быть пригодно как для ручного, так и для автоматизированного тестирования.

ЛИТЕРАТУРА

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

ТЕСТОВЫЙ СЦЕНАРИЙ

Представление сценариев, удобное для ручного тестирования, – тест-план в виде текстового документа, в котором каждый тестовый пример – это один раздел.

Для тестового примера в этот документ записывается следующая информация:

- ✓ идентификатор;
- ✓ описание теста и его цель;
- ✓ ссылки на тестируемую часть системы;
- ✓ ссылки на используемую проектную документацию, в частности тест-требования;
- ✓ перечисление действий сценария;
- ✓ ожидаемая реакция системы на каждый пункт сценария.

Действия сценария должны быть описаны так, чтобы их мог воспроизвести человек с практически любым уровнем подготовки. Описание ожидаемой реакции системы должно также быть записано таким образом, чтобы можно было однозначно судить о том – соответствует реакция ожидаемой или нет.

Неудачной ожидаемой реакцией при ручном тестировании была бы запись:

> *Сообщение «Загрузка» пропадает через приемлемое время.*

Степень приемлемости здесь будет зависеть от терпеливости тестировщика, и обеспечить повторяемость тестирования будет затруднительно. Более удачной формой описания той же самой ожидаемой реакции будет:

> *Сообщение «Загрузка» исчезает с экрана не более, чем через 10 секунд после появления.*

Ниже приведен пример описания тестового примера в виде сценария, предназначенного для ручного тестирования [Синицын 2006]:

> *Группа тестов: Работа с учетными записями.*
>
> *Тестовый пример: 1289-15*

Назначение: *Проверка того, что учетная запись пользователя проверяется перед началом передачи данных и в случае ввода записи по умолчанию при максимальной защите системы передачи не происходит.*

Тест-требования: *8.5.8.1, 8.5.8.2*

Предусловия для теста: *Система должна быть приведена в состояние «Максимальная защита» и сброшена в настройки по умолчанию.*

Критерий прохождения теста: *Все ожидаемые значения совпадают с реальными значениями.*

Сценарий тестирования:

№	Шаг сценария	Ожидаемый результат
1	Запустить терминальный клиент и соединиться с системой по адресу 127.0.0.1	Должно появиться приглашение терминала TRANSFER>
2	Запустить процесс передачи данных при помощи ввода команды SEND DATA	Должно появиться приглашение DATA TRANSFER INITIATED и следующими двумя строками Enter your credentials… Login:
3	Ввести имя учетной записи default	Должна появиться строка Password:
4	Ввести пароль default	Должно появиться сообщение Default user blocked – system set to High security и соединение с терминалом должно быть прервано

Такие тест-планы совмещают с отчетами о проведении тестирования, добавляя в таблицу описания сценария третью и четвертые колонки – “Реальный результат” и “Соответствует”, в который заносятся реальная реакция системы и указание на совпадение/несовпадение результатов соответственно. В конце описания каждого тестового примера добавляется графа “Пройден/не пройден”, в которую заносится информация о том, пройден ли тестовый пример в целом. В конце всего тест-плана совмещенного с отчетом помещается графа “Тестовых примеров пройдено/всего”, в которую заносится число пройденных тестовых примеров и общее их число.

Сценарии тестирования для автоматического тестирования часто описывают на том или ином языке программирования. Например, методы в тестирующих классах Microsoft Visual Studio Team Edition представляют собой именно пошаговые описания действий, которые необходимо выполнить тестовому окружению для проведения тестирования.

ЛИТЕРАТУРА

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

ИСПОЛЬЗОВАНИЕ ТАБЛИЦ

ПРИМЕР ПРИЛОЖЕНИЯ

Для систем с большим количеством точек входа и для приложений с большим пространством входов таблицы являются эффективным средством при определении тестовых примеров [Тампе 2003].

Приложение с графическим пользовательским интерфейсом имеют много возможностей, с помощью которых пользователь может заполнить поле ввода.

Внешний вид рассматриваемого диалогового окна представлен на рис.4.1. Пользователь заполняет поля username и password, а затем нажимает либо кнопку OK для подтверждения ввода, либо кнопку CANCEL, чтобы прервать диалог.

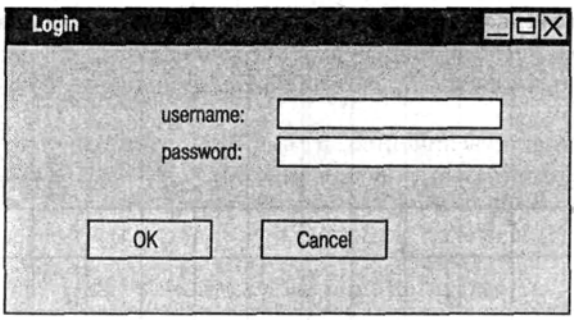

Рис. 4.1. Пример диалогового окна

Фокус (текущая позиция курсора) может находиться в четырех местах, каждое из которых должно правильно отвечать на любое нажатие клавиши, исходя из своего контекста.

Рассмотрим клавиатуру с 88 различными клавишами. Такое количество клавиш дает более 600 различных комбинаций нажатия клавиш для каждого поля ввода GUI. Эта величина образуется с помощью следующих комбинаций.

- ✓ 88 различных клавиш – дают 26 символов нижнего регистра, 10 цифр, 12 символов пунктуации, 12 функциональных клавиш и 28 прочих клавиш (ENTER, INSERT, HOME, PAGE UP, UP ARROW, SHIFT, CONTROL).
- ✓ 87 комбинаций с клавишей SHIFT – нажатие клавиши SHIFT можно сочетать с нажатием одной из остальных 87 клавиш, некоторые из них дают символы верхнего регистра и другие знаки пунктуации.
- ✓ 87 комбинаций с клавишей CONTROL – нажатие клавиши CONTROL можно сочетать с нажатием одной из 87 оставшихся клавиш.
- ✓ 87 комбинаций с клавишей ALT – клавишу ALT можно использовать совместно с одной из 87 оставшихся клавиш.
- ✓ 86 комбинаций CONTROL+SHIFT – существует 86 возможных трехклавишных комбинаций, в которых задействуют одновременно CONTROL+SHIFT и любая третья клавиша.
- ✓ 86 комбинаций ALT+SHIFT – еще одна распространенная комбинация нажатия трех клавиш, ALT+SHIFT+<любая третья клавиша.
- ✓ 86 комбинаций CONTROL+ALT – комбинация включает в одновременное нажатие клавиш CONTROL, ALT и любой третьей клавиши.

Некоторые комбинации дают различный эффект в зависимости от основного состояния системы. В качестве примера можно привести символ пробела, который игнорируется, если он следует после другого пробела, но важен, если следует за другим символом.

Таблица свободно отражает поля ввода на список возможных клавишных комбинаций. Часть этой таблицы показана в табл.4.1. Ограничивая расчеты лишь наиболее распространенными двух- и трехклавишными комбинациями, полная таблица будет содержать более 600 столбцов.

Таблица 4.1. Перечень возможных клавишных комбинаций для диалогового окна

Текущая позиция курсора	a	b	...	z	0	...	9	SHIFT + a	SHIFT + b	...	SHIFT + z	SHIFT + 0	...	SHIFT + 9	CONTROL + a	CONTROL + b
поле Username																
поле Password																
кнопка OK																
кнопка CANCEL																

...	CONTROL + z	CONTROL + 0	...	CONTROL+9	CONTROL+SHIFT+a	CONTROL+SHIFT+b	...	CONTROL+SHIFT+z	CONTROL+SHIFT+0	...	CONTROL+SHIFT+9	ENTER	TAB	BACKSPACE	DELETE	HOME	Left arrow	Right arrow	Down arrow	Up arrow	...

Перечень всех возможных комбинаций дает в итоге огромную таблицу. Несмотря на это, данный подход имеет свои преимущества.

1. Упражнение по созданию большой таблицы помогает начинающим тестерам понять важность тестирования.
2. Большая таблица откроет глаза тем наивным менеджерам, которые не предоставили тестерам достаточно времени на тестирование и недооценили этот важный этап создания качественного продукта.
3. Если тестеры в замешательстве и не знают с чего начать, сведение подробного перечня всех возможных комбинаций ввода в таблицу вызывает в жизни идеи, которые затем служат основой поиска более продуктивных путей тестирования.
4. Тестеры явственнее ощущают преимущества, предоставляемые инструментальными средствами.

ДОКУМЕНТИРОВАНИЕ ТЕСТОВЫХ ПРИМЕРОВ

В простейшем случае таблица представляет из себя матрицу, где в крайнем левом столбце собраны все возможные варианты начальных условий, а в самой верхней строке перечислены действия, которые необходимо выполнить. Каждая из ячеек матрицы, образуемая пересечением строк и столбцов, обозначает отдельный тестовый пример.

Представленные далее варианты таблиц иллюстрируют различные методы документирования информации тестирования диалогового окна.

Формат таблицы № 1	Дает очень короткое описание ожидаемых результатов.
Формат таблицы № 2	Использует анализ граничных значений для описания различных условий ввода.
Формат таблицы № 3	Полное описание теста собирается в отдельный документ, а контрольные данные — в тестовой таблице.
Формат таблицы № 4	Используется деление на классы эквивалентности для группировки вариантов ввода, дающих одинаковые результаты.
Формат таблицы № 5	Начальные условия, входные данные и ожидаемые результаты определяются в отдельных строках тестовой таблицы.

ФОРМАТ ТЕСТОВЫХ ТАБЛИЦ №1

МЕТОД. Дается краткое описание ожидаемых результатов (см. табл.4.2).

Таблица 4.2. Формат тестовой таблицы №1: фиксация краткого описания

Поле ввода	<a>	<b>	<Shift> + <a>	<Shift> + <b>	<1>	<#>	<ENTER>	<TAB>	<BACKSPACE>	<DELETE>	<HOME>	<LEFT ARROW>	...
Поле Username	плюс 'a'	плюс 'b'	плюс 'A'	плюс 'B'	плюс '1'	ошибка: неверный символ	ввести имя пользователя/пароль	переход в поле Password	стирание символа слева	стирание символа справа	переход в начало поля Username	переход на один символ влево	
Поле Password	плюс '*'	плюс '*'	плюс '*'	плюс '*'	плюс '*'	ошибка: неверный символ	ввести имя пользователя/пароль	переход на кнопку OK	стирание символа слева	стирание символа справа	переход в начало поля Password	переход на один символ вправо	
Кнопка OK	без изменений	без изменений	без изменений	без изменений	без изменений	без изменений	ввести имя пользователя/пароль	переход на кнопку Cancel	без изменений	без изменений	без изменений	без изменений	
Кнопка Cancel	без изменений	без изменений	без изменений	без изменений	без изменений	без изменений	ввести имя пользователя/пароль	переход на поле Username	без изменений	без изменений	без изменений	без изменений	

ПРЕИМУЩЕСТВА. В итоге получается контрольная таблица для всех возможных вариантов ввода. Каждой ячейке соответствует отдельный тестовый пример.

НЕДОСТАТКИ. Не определены начальные условия, поэтому отсутствует критерий однозначности вводимых данных. В большинстве ситуаций ожидаемый результат зависит от состояния GUI. Рассмотрим тест, в котором говорится: “Введите символ k”. В данном случае не ясно, будет ли символ “k” первым символом в пустом поле ввода или он прибавляется к уже введенным символам? Как отреагирует система, если тестер при вводе превысит максимальное количество символов?

ФОРМАТ ТЕСТОВЫХ ТАБЛИЦ №2

МЕТОД. Используется анализ граничных значений для описания условий ввода.

Типичные тесты, заданные на границе, обычно содержат следующие три условия.

- ✓ Граничное значение.
- ✓ Граничное значение – 1.
- ✓ Граничное значение + 1.

Применение анализа граничных значений к ограниченным диапазонам дает в результате следующие условия теста.

- ✓ Min.
- ✓ Min-1.
- ✓ Min+1.
- ✓ Max.
- ✓ Max-1.
- ✓ Max+1.

Тестеры часто добавляют еще один тестовый пример.

- ✓ “промежуточное” или “типичное” значение для улучшения общей оценки.

В табл.4.3 показано, как применяется анализ граничных значений.

Таблица 4.3. Применение анализа граничных значений в примере с диалоговым окном

Условие для граничного значения	Исходное состояние полей Username или Password	Ожидаемые результаты после ввода дополнительного символа
min	0 символов, “пустое поле”	1 символ в поле
min –1	нельзя задать размер поля в –1 символ	*невозможно*
min +1	в поле 1 символ	2 символа в поле
max	“максимальное число символов”, даже если неизвестно, каким должно быть это значение	Приложение игнорирует лишние символы. Сообщение об ошибке не появляется. Примечание: в этой ситуации в действительности тестируется условие “max –1”
max –1	“число символов, равное max –1”	максимальное число символов в поле
max +1	В примере с диалоговым окном нет возможности ввести такое число символов, которое ревышает максимальное значение	*невозможно*
типичное значение	Это число символов в диапазоне “min +2..max –2”. Если неизвестно, какое значение будет “типичным”, то в тесте задается такое значение числа символов, которое не было задействовано в предыдущих примерах	Приложение принимает новый символ

ПРЕИМУЩЕСТВА. Существуют отдельные условия для задания пустого, частично и полностью заполненного поля; верной и неверной пары значений имя пользователя и пароль. Эта таблица удобна для контрольных проверок различных тестов.

НЕДОСТАТКИ. Большинство тестов допускают двоякое толкование и, следовательно, не могут быть воспроизведены вследствие неполноты описания ожидаемых и реальных результатов. Например, если в поле username уже есть один символ, неизвестно каким был предыдущий. Неизвестно также, какая пара значений имя пользователя и пароль обрабатываются. При нажатии клавиш BACKSPACE, DELETE или клавиш со стрелками неизвестно точное положение курсора в непустом поле.

ФОРМАТ ТЕСТОВЫХ ТАБЛИЦ №3

МЕТОД. В отдельном документе фиксируется исчерпывающее описание теста, а в тестовой таблице на это писание делается ссылка.

Ожидаемые результаты могут быть точными только тогда, когда заданы недвусмысленные входные условия. На рис.4.3 показано одно из возможных описаний теста, в котором указывается содержимое диалогового окна, предшествующее впечатыванию символа “a”, описываются ожидаемые результаты.

Идентификация

ID тестового примера: *тп-402* Версия программы: _____

Подсистема: *тесты для поля Username* Операционная система: _____

Тестер: _____ Дата проведения теста: _____

Начальные установки

1. *Вызвать диалоговое окно входа в систему*
2. *В поле Username ввести символ "г"*
3. *Убедиться, что все остальные области ввода пустые*

Ввод данных

1. *Поместить курсор в поле Username*
2. *Ввести символ "а"*

Ожидаемые результаты

В поле Username появляются символы "га"

Действительные результаты ☐ Пройден ☐ Провален

Рис. 4.3. Образец тестового примера

В исправленном тестовом примере на рис.4.4 не только указывается положение курсора, а также тестируется возможность вставки символа, при этом сочетается два метода выполнения теста.

Идентификация

ID тестового примера: *тп-402* Версия программы: _____

Подсистема: *тесты для поля Username* Операционная система: _____

Тестер: _____ Дата проведения теста: _____

Начальные установки

1. *Вызвать диалоговое окно входа в систему*
2. *В поле Username ввести символ "г"*
3. *Убедиться, что все остальные области ввода пустые*

Ввод данных

Часть А

1. *Поместить курсор в поле Username после "г"*
2. *Ввести символ "а"*

Часть Б

1. *Возврат к начальной установке*
2. *Поместить курсор в поле Username перед "г"*

Ожидаемые результаты

Часть А

В поле Username появляются символы "га"

Часть Б

В поле Username появляются символы "аг"

Действительные результаты ☐ Пройден ☐ Провален

Рис. 4.4. Образец тестового примера, состоящего из нескольких частей

ПРЕИМУЩЕСТВА. В этом формате тестовых таблиц дается указатель на отдельные документы тестовых примеров. Такой интуитивно понятный указатель помогает передать то, как размещены все тестовые примеры, на которые были сделаны ссылки, в общей схеме тестирования.

НЕДОСТАТКИ. Нет гарантии завершенности тестового примера. То, будут ли тесты, на которые сделаны ссылки, недвусмысленными и повторяемыми, зависит от способности тестера задавать полные тестовые примеры.

ФОРМАТ ТЕСТОВЫХ ТАБЛИЦ №4

МЕТОД. Для сочетания входных данных, дающий аналогичный результат используется разбиение на классы эквивалентности. Например, если поле username пустое, то разве имеет значение, какой символ будет введен первым “а”, “б” или “ю”? Поведение приложения будет одинаковым.

В табл.4.6 были введены классы эквивалентности путем объединения столбцов и строк, в которых проверяются аналогичные условия; в результате размер таблицы уменьшился и число тестовых примеров снизилось.

Таблица 4.6. Использование классов эквивалентности для комбинирования аналогичных столбцов и строк

Следующая клавиша		Клавиши нижнего регистра	Клавиши верхнего регистра	Числа	Верные символы	Невер-ные символы	<ENTER>	<TAB>	<BACK-SPACE>	<DELETE>	<HOME>	<LEFT ARROW>	...
Текущее входное состояние													
поле Username	пустое поле												
	от одного до max – 1 символов												
	максимальное число символов в поле												
поле Password	пустое поле												
	от одного до max – 1 символов												
	максимальное число символов в поле												
кнопка OK	отсутствует имя пользователя или пароль												
	неверное имя пользователя/ пароль												
	верное имя пользователя/ пароль												
Кнопка Cancel													

В этом конкретном примере используется три различных типа классов эквивалентности.

- ✓ Аналогичное нажатие клавиш.
- ✓ Текущее состояние поле username и password.
- ✓ Функция CANCEL.

В первом типе разбиения комбинируется нажатие клавиш, приводящее к аналогичным результатам. В одном классе эквивалентности содержатся все буквы нижнего регистра, в то время как в другом – все буквы верхнего регистра.

При втором разбиении на классы эквивалентности обнаруживается, для примера с диалоговым окном слияние некоторых строк в таблице дополнительно сокращает пространство входов. При добавлении новых буквенно-числовых значений в поле username приложение диалогового окна ведет себя одинаково и тогда, когда в поле один, два или три символа. Поведение изменяется, когда превышается максимальное число

разрешенных символов. Это приводит к созданию трех классов эквивалентности: одного – для пустого поля, другого – для максимального числа символов, а третьего – для любого другого числа символов, которое лежит между этими предельными значениями. Для категории кнопки OK строки, соответствующие пустому полю имени пользователя и пароля, объединяются в один класс эквивалентности под названием отсутствует имя пользователя или пароль. Этот класс отличается от класса **неверное имя пользователя/пароль** тогда, когда и имя пользователя, и пароль содержат какой-то набор символов, но либо имя пользователя, либо пароль неверны.

Третье, и последнее разбиение на классы эквивалентности влияет на кнопку CANCEL. В связанных с этим разбиением тестах больше не учитывается содержимое полей username и password, поскольку здесь рассматривается выход из диалогового окна. При задании команды CANCEL тестер должен убедиться, что значения имени пользователя и пароля не обновлялись или не использовались каким-либо способом.

ПРЕИМУЩЕСТВА. Предоставляя более детальную информацию о содержимом поле ввода диалогового окна, описание становится более полным и такие тесты можно будет точно воспроизвести.

НЕДОСТАТКИ. При одностроковом подходе сложнее определить, какие условия тестов отсутствуют. Создание таблицы такого формата не гарантирует, что тесты будут заданы хорошо. Для сравнения, матричный формат дает возможность быстро установить тип тестируемых условий. Тестер отвечает за предоставление адекватной информации для создания всестороннего описания тестового примера.

ЛИТЕРАТУРА

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

КОНЕЧНЫЙ АВТОМАТ

КОНЕЧНЫЙ АВТОМАТ

Конечный автомат – это поведенческая модель, состояние которой зависит как от предыдущих, так и от текущих входных данных. Диаграмма переходов – это графическое представление конечного автомата. Ее задача – отобразить состояние, которое может принять система или компонент. Диаграмма переходов также показывает события или обстоятельства, результатом которых стал переход системы компонент из одного состояния в другое.

В качестве примера рассмотрим тест для секундомера [Тампе 2003]. Диаграмма переходов состоит из трех состояний, а интерфейс содержит три типа входных данных и один – выходных. Каждый из трех типов входных данных представляет собой событие, влияющее на систему. Интерфейс имеет следующее описание.

Вход		
	НАЧАЛО	истекшее время продолжает увеличиваться, начиная с текущего отображаемого значения времени. Значение времени увеличивается каждую секунду
	КОНЕЦ	увеличение значения истекшего времени прекращается и отображается последнее значение времени
	СБРОС	текущее время сбрасывается на нулевое значение
Выход		
	Время	текущее значение времени

На диаграмме перехода (рис.5.1 [Тампе 2003]) показано поведение секундомера и связь между входными данными и состояниями. Овалы – состояния секундомера, а стрелки – переходы из одного состояния в

другое. Каждая стрелка предоставляет информацию из двух частей: 1) событие, вызвавшее переход; 2) действие, произошедшее в результате. В этом случае некоторые события влияют на время.

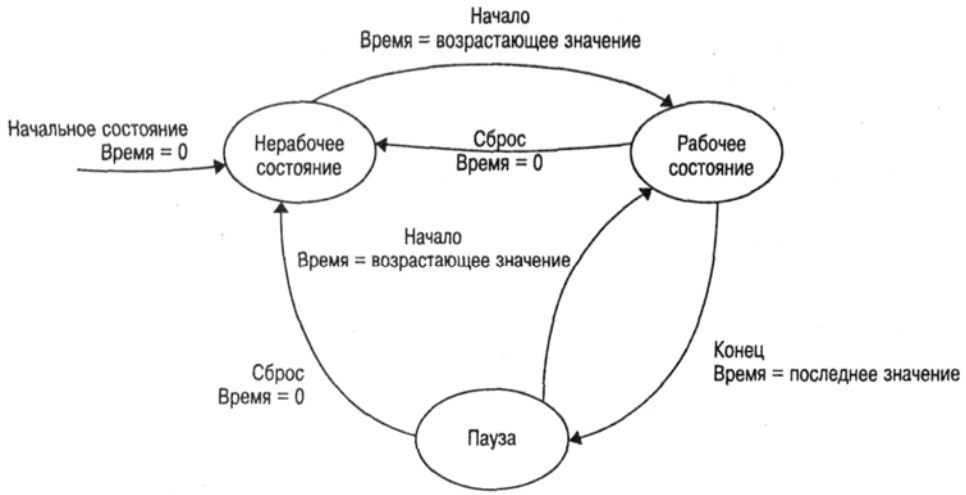

Рис. 5.1. Диаграмма переходов, иллюстрирующая поведение секундомера

В таблице состояний (табл.5.1 [Тампе 2003]) показано другое представление информации (рис.5.1). В первом столбце перечислены всевозможные события, заголовки остальных столбцов содержат перечень всех возможных состояний. В ячейках содержится информация о том, как каждое событие влияет на состояние. Номера в нижнем правом углу каждой ячейки – это метки для идентификации ячейки.

Таблица 5.1. Таблица состояний, описывающая диаграмму переходов

Событие	Нерабочее состояние	Состояние выполнения	Состояние паузы
НАЧАЛО	Выполнение		Выполнение
	//Время = возрастающее значение	?	//Время = возрастающее значение
	ячейка 1	ячейка 4	ячейка 7
КОНЕЦ		Пауза	
	?	//Время = последнее значение	?
	ячейка 2	ячейка 5	ячейка 8
СБРОС		Неактивен	Неактивен
	?	//Время = 0	//Время = 0
	ячейка 3	ячейка 6	ячейка 9

На рис.5.1 не дано определение четырем событиям – на рисунке отсутствуют дуги. Чтобы определить отсутствующие потоки на диаграмме секундомера, для переходов в неактивное состояние необходимо прибегнуть к кнопке СБРОС без учета предыдущего состояния. Аналогично кнопка НАЧАЛО будет переводить секундомер в состояние выполнения. Это отражено в ячейках 3 и 4 в табл.5.2 [Тампе 2003]. Кнопка КОНЕЦ будет переводить все состояния в состояние паузы, что записано в ячейке 8. Эти дополнительные переходы содержатся в таблице состояний табл.5.2 и в обновленной диаграмме переходов на рис.5.2 [Тампе 2003].

Таблица 5.2. Таблица состояний, в которой заданы все переходы

Событие	Нерабочее состояние	Состояние выполнения	Состояние паузы
НАЧАЛО	Выполнение	Выполнение	Выполнение
	//Время = возрастающее значение	//Время = возрастающее значение	//Время = возрастающее значение
	ячейка 1	ячейка 4	ячейка 7
КОНЕЦ	Неактивен	Пауза	Пауза
	//Время = 0	//Время = последнее значение	//Время = последнее значение
	ячейка 2	ячейка 5	ячейка 8
СБРОС	Неактивен	Неактивен	Неактивен
	//Время = 0	//Время = 0	//Время = 0
	ячейка 3	ячейка 6	ячейка 9

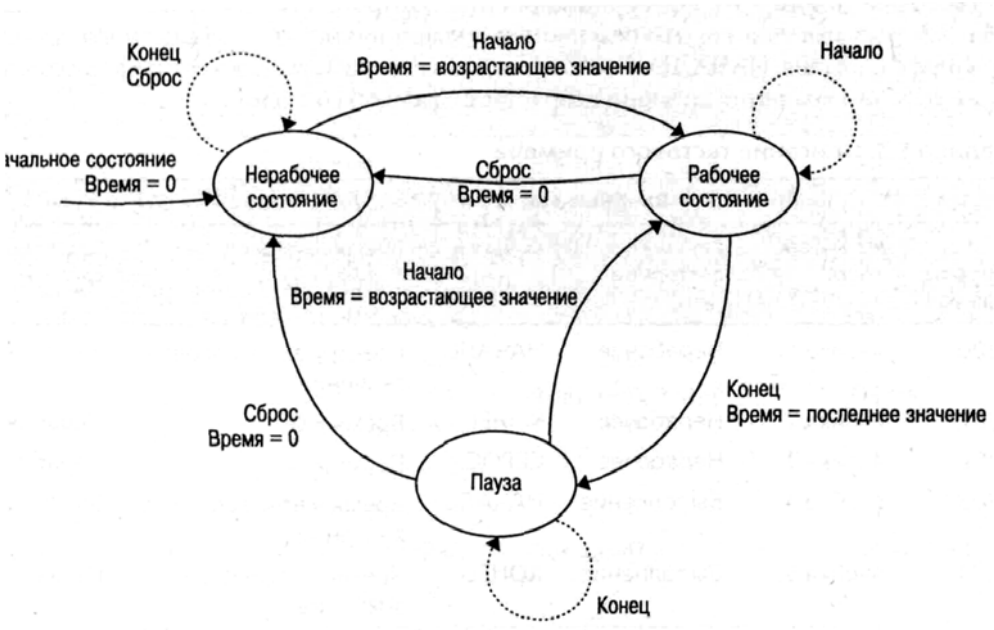

Рис. 5.2. Диаграмма переходов, содержащая все переходы

При тестировании конечного автомата тесты, которые провалились, можно обозначить с помощью любого из следующих условий.

- ✓ Состояние, показанное на диаграмме переходов, отсутствует в коде.
- ✓ Операция, показанная на диаграмме переходов, отсутствует в коде.
- ✓ Операция приводит к неверному состоянию.
- ✓ Состояние выполняет неверные функции.

СОЗДАНИЕ ТЕСТОВЫХ ПРИМЕРОВ НА ОСНОВЕ ТАБЛИЦЫ СОСТОЯНИЙ

Информация, содержащаяся в таблице состояний, легко преобразуется в тесты благодаря табличной форме записи. Табл.5.3 [Тампе 2003] включает список тестов, полученных на основе информации табл.5.2. Эта таблица тестов состоит из шести столбцов.

ID тестового примера	уникальный идентификатор для каждого тестового примера.
Источник теста	указывает на соответствующую ячейку в таблице состояний.
Текущее состояние	начальные условия, необходимые для запуска теста.
Событие	ввод, выполненный пользователем.
Время на выходе	текущее значение времени, возвращаемое секундомером.
Следующее состояние	новое состояние секундомера.

Таблица 5.3. Описание тестового примера

		Входные данные		Ожидаемые результаты	
ID тестового примера	**Источник теста**	**Текущее состояние**	**Событие**	**Время на выходе**	**Следующее состояние**
T-200	ячейка 1	Нерабочее	НАЧАЛО	Время = возрастающее значение	Выполнение
T-201	ячейка 2	Нерабочее	КОНЕЦ	Время = 0	Нерабочее
T-202	ячейка 3	Нерабочее	СБРОС	Время = 0	Нерабочее
T-203	ячейка 4	Выполнение	НАЧАЛО	Время = возрастающее значение	Выполнение
T-204	ячейка 5	Выполнение	КОНЕЦ	Время = последнее значение	Пауза
T-205	ячейка 6	Выполнение	СБРОС	Время = 0	Нерабочее
T-206	ячейка 7	Пауза	НАЧАЛО	Время = возрастающее значение	Выполнение
T-207	ячейка 8	Пауза	КОНЕЦ	Время = последнее значение	Пауза
T-208	ячейка 9	Пауза	СБРОС	Время = 0	Нерабочее

Тестовый пример T-200, информация для которого была получена из ячейки 1 и табл.5.2., указывает на то, что секундомер должен начинать с неактивного состояния и ожидать события НАЧАЛО. Это переводит секундомер в состояние выполнения и таким образом значение времени возрастает с каждой секундой.

В табл.5.4 [Тампе 2003] дано определение дополнительным тестовым примерам, которые обеспечивают более доскональные входные условия, задавая фиксированный интервал для входной последовательности. В этой обновленной таблице содержится один столбец с входными данными, который помечен как “входная последовательность” и описывает последовательность вводимых команд.

Таблица 5.4. Описание тестового примера с более подробными входными условиями

		Входные данные	Ожидаемые результаты	
ID тестового примера	**Источник теста**	**Входная последовательность**	**Время на выходе**	**Следующее состояние**
T-210	ячейка 1	Нерабочее	Время = 0, 1, 2, 3, ...	Выполнение
		Событие НАЧАЛО		
T-211	ячейка 1	Нерабочее	Время = 0, 1, 2, 3, 4, 5	Пауза
	ячейка 5	Событие НАЧАЛО		
		Ожидание 5 секунд		
		Событие КОНЕЦ		
T-212	ячейка 7	Запуск теста T-211	Время = 6, 7, 8, 9	Пауза
	ячейка 5	Событие НАЧАЛО		
		Ожидание 3 секунды		
		Событие КОНЕЦ		
T-213	ячейка 9	Запуск теста T-212	Время = 0	Нерабочее
		Событие СБРОС		

ПРОВЕДЕНИЕ ТЕСТА И УРОВНИ ТЕСТИРОВАНИЯ

На проведение теста влияет множество факторов. Прежде всего, необходимо создать механизмы для управления тестами: тестовые драйвера, а также способы, с помощью которых тестер может вводить входные данные в приложение и переводить секундомер в новое состояние. Также потребуется система для фиксации текущего времени и определения текущего состояния секундомера. Еще одна компонента выполнения теста – это регистрация реальных результатов. Можно расширить таблицу тестовых примеров, создав столбец, в котором тестер будет отмечать результаты прохождения теста (табл.5.5 [Тампе 2003]).

Таблица 5.5. Более совершенная таблица тестового примера

		Входные данные		Ожидаемые результаты		Реальные результаты
ID тестового примера	Источник теста	Текущее состояние	Событие	Время на выходе	Следующее состояние	прошел/ провалился
Т-200	ячейка 1	Нерабочее	НАЧАЛО	Время = возраста-ющее зна-чение	Выполнение	
Т-201	ячейка 2	Нерабочее	КОНЕЦ	Время = 0	Нерабочее	
Т-202	ячейка 3	Нерабочее	СБРОС	Время = 0	Нерабочее	
Т-203	ячейка 4	Выполнение	НАЧАЛО	Время = возраста-ющее зна-чение	Выполнение	
Т-204	ячейка 5	Выполнение	КОНЕЦ	Время = по-следнее значение	Пауза	
Т-205	ячейка 6	Выполнение	СБРОС	Время = 0	Нерабочее	
Т-206	ячейка 7	Пауза	НАЧАЛО	Время = возраста-ющее зна-чение	Выполнение	
Т-207	ячейка 8	Пауза	КОНЕЦ	Время = по-следнее значение	Пауза	
Т-208	ячейка 9	Пауза	СБРОС	Время = 0	Нерабочее	

Остается ответить еще на один вопрос: как будет проводиться тестирование секундомера – на уровне системы или на уровне элементов. Основные различия заключаются в рамках и взаимодействиях, требуемых для выполнения тестов.

1. Поэлементное тестирование затрагивает наименьшие единицы компиляции какого-либо приложения. Цель такого тестирования заключается в демонстрации того, что каждый отдельный элемент функционирует так, как было задумано, перед интеграцией его с другими элементами. Тестеры, как правило, запускают элемент с помощью тестового драйвера, который передает информацию о выполнении теста тестируемому элементу, вводят данные.
2. Тестирование уровня интеграции заключается в комбинировании отдельных элементов и подтверждении того, что они функционируют корректно. Для проведения тестирования интеграции существует несколько подходов, которые перечислены дальше.
3. Тестирование уровней системы применяется для целостных приложений. Это чаще всего касается видения системы пользователем. Тестеры обычно взаимодействуют с приложением, вводя данные тем же способом, который использует конечный пользователь.

Если таймер реализуется как самодостаточная единица компиляции, тестирование можно проводить на уровне элементов. В этом примере более крупное приложение структурируется таким образом, чтобы таймер был его модулем.

ЛИТЕРАТУРА

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