МОДУЛЬ 7: «ОТЧЕТЫ О ПРОХОЖДЕНИИ ТЕСТОВ»

ОТЧЕТЫ О ПРОХОЖДЕНИИ ТЕСТОВ

СВЯЗЬ ОТЧЕТА О ПРОХОЖДЕНИИ ТЕСТА С ДРУГИМИ ДОКУМЕНТАМИ

ОТЧЕТЫ О ПРОХОЖДЕНИИ ТЕСТОВ – основной (а иногда единственный) источник для заключения о соответствии протестированной системы требованиям. После выполнения всех тестов, описанных в тест-планах, среда тестирования создает отчет о том, насколько успешно система выполнила эти тесты. Такой отчет содержит информацию о каждом выполненном тестовом примере (его идентификатор) и результат его выполнения – успех или неудачу.

По результатам анализа отчетов о прохождении тестов могут быть выявлены либо дефекты в самой системе, либо некорректно составленные или противоречивые требования. В обоих случаях результаты анализа служат основой для создания запросов на изменение требований и/или кода системы. После исправления дефектов при регрессионном тестировании неуспешно выполненные тестовые примеры должны выполниться успешно (Рис.19) [Синицын 2006].

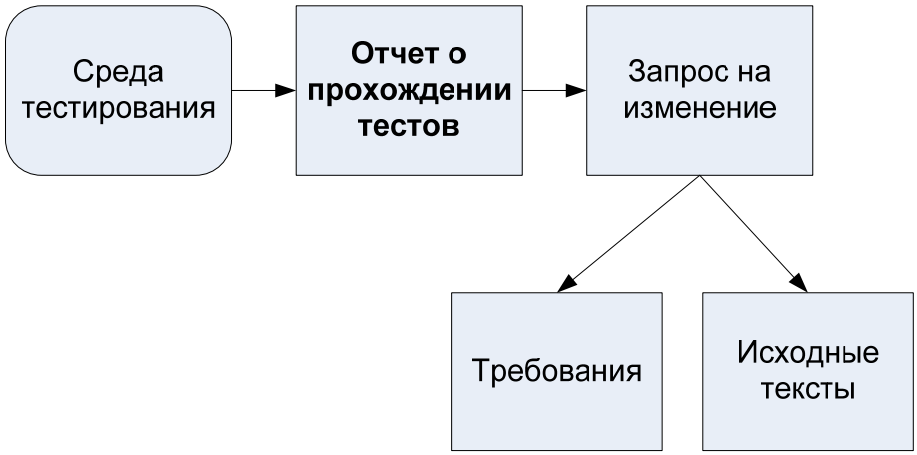

Рис.19 Генерация отчета о прохождении тестов и изменения по результатам его анализа

Отчеты о прохождении тестов служат основой для отслеживания состояния проекта – если количество обнаруживаемых дефектов (неуспешно выполненных тестовых примеров) с течением времени падает, при условии сохранения качества тестирования, – это свидетельствует о повышении качества системы.

ВОЗМОЖНЫЕ ФОРМЫ ПРЕДСТАВЛЕНИЯ ОТЧЕТОВ О ПРОХОЖДЕНИИ ТЕСТОВ

В стандарте IEEE 829 отчет о прохождении тестов разделен на три различных документа: Test log (общий отчет о прохождении тестов), Test incident report (отчет о проблемах, выявленных в результате выполнения тестов) и Test summary report (общая статистика прохождения тестов).

Отчет о прохождении тестов считается единым документом, разделенным на три части:

- ✓ общая (заголовочная информация);
- ✓ результаты выполнения тестовых примеров (положительные и отрицательные);
- ✓ итоговая информация о выполнении тестовых примеров (общая статистика по выполненным тестам).

ЗАГОЛОВОЧНАЯ ЧАСТЬ ОТЧЕТА о прохождении тестов служит для идентификации отчета и протоколирования того, какая часть разрабатываемой системы подвергалась тестированию, какая ее

версия, какая конфигурация тестового стенда использовалась для выполнения тестов. В заголовочную часть отчета о выполнении тестов включается следующая информация:

- ✓ Название проекта или тестируемой системы.
- ✓ Общий идентификатор группы тестовых примеров, включенных в отчет.
- ✓ Идентификатор тестируемого модуля или группы модулей и номера их версий.
- ✓ Ссылку на разделы и версии тест-требований или функциональных требований, по которым написаны тесты, для которых сгенерирован отчет.
- ✓ Время начала выполнения теста и его продолжительность.
- ✓ Конфигурацию тестового стенда, на которой выполнялся тест.
- ✓ Имена и фамилии автора тестов и/или лица, выполнявшего тесты.

Ниже показан пример заголовочной части отчета [Синицын 2006]. Красными цифрами в скобках обозначены пункты приведенного выше списка.

```
************************************************

** Document Test Environment

**     User's Computer:        COMPUTER_185 (6)

**     Testing Host Application: Facility Test (6)

**     Testing Host Version:    5.12 (6)

************  Server Related Data    *************

**     Server Computer:        SERVER_105 (6)

**     Server Version:         6.24.0 (Build 16) (6)

**     Configuration:          Control remote bench (6)

**     Mode:                   Real time (6)

**     Test executed on: 7/29/06; at 10:09:40 AM (5)

**     Tester Name is [ Sidorov A. ] (7)

**     Software Version is: CNTRL 115 01 5 (1)

**     Test Station being used is: COMPUTER_185 (6)

************************************************

=====================================================================

            REMOTE CONTROL FUNCTION SOFTWARE TEST REPORT

=====================================================================P
roject Name         : Facility Remote Control (1)

Function Name       : Infrared Transmitter Signal Handler (3)

Test Name           : IRDA_C05A_1091K (2)

Document Name       : SSRD for the Remote Control Function (4)
```

```
Paragraph Name       : Button Signals (4)
Primary Paragraph Tag : [PTAG::SSRD IR BTN SIGNALS] (4)
Template Class       : Test
Shall Tag(s)         : SSRD IR BTN SIGNALS 10 (4)
Shall(s) template    : Test
 ---------------------------------------------------------------
                    MODIFICATION HISTORY:
 ---------------------------------------------------------------
 Ver   Date        Author      Change Description        CR No.
 ---   ---------   ----------  ------------------------  --------
 01    19 Jul 06   Ivanov K. (7)Initial Development.      CR_10
;  SIMULATION RESULTS FILE
;  Matrix Compiler CORE VERSION 3.00
;       TEST PLAN
;  ELEMENT: IRDA_IA.TMC   (2)
;  TITLE: Test Plan for Inrfrared source files test (1)
;  TEST DATE/TIME Wed 02.11.2005 23:12:53            (5)
;  SYS section: 2.3.5.6 Version: 24                  (4)
;  SRD section: 6.3 Version: 12                      (4)
;  SDD section: 12.3 Version: 33                     (4)
;  SOURCE FILE(S):  IRDA.C Version: 18               (4)
;                                IRDA.H Version: 2   (4)
;
;  SIMULATOR SETUP:          (6)
;    MODE HIGH      (6)
;    INC CIP.INC    (6)
```

СЛЕДУЮЩАЯ ЧАСТЬ ОТЧЕТА о прохождении тестов содержит информацию о результате выполнения каждого тестового примера – завершился ли он успешно или в результате его выполнения были выявлены несоответствия с ожидаемым результатом. Эта часть отчета может быть представлена в одной из двух форм – полной или краткой. Полная форма содержит всю информацию о тестовом примере, краткая – только информацию об обнаруженных в результате выполнения тестового примера несоответствий ожидаемых и реальных выходных значений.

Запись о результате прохождения каждого тестового примера в полной форме содержит следующую информацию:

1. Идентификатор тестового примера.
2. Краткое описание тестового примера.
3. Перечисление всех входных значений тестового примера.
4. Перечисление всех ожидаемых и реальных выходных значений теста.
5. Для каждой пары “ожидаемое и реальное выходное значение” – информацию о совпадении или несовпадении этих значений.
6. Сообщение о том, пройден или не пройден тестовый пример.

В краткой форме каждая запись обычно содержит следующую информацию:

1. Идентификатор тестового примера.
2. Перечисление не совпавших ожидаемых и реальных выходных значений тестового примера.
3. Для каждой пары “ожидаемое и реальное выходное значение” – информацию о совпадении или несовпадении этих значений.
4. Сообщение о том, пройден или не пройден тестовый пример.

Ниже приведен пример информации о прохождении тестового примера в краткой и полной формах [Синицын 2006]. Красными цифрами в скобках отмечены соответствующие пункты приведенных выше списка для краткой и полной форм.

```
[Testcase 163] (1) :True: <EQ> :True: (4) ** Passed Number 163 **

[Testcase 164]       :True: <EQ> :True:     ** Passed Number 164 **

[Testcase 165]       :True: <EQ> :True:     ** Passed Number 165 **

[Testcase 166]       :False:<EQ> :True: (4) ** Fail Number 1 **

*** Inputs for Testcase 166

DisplayTextLine2.ItemChecked  = 2 (2 expected)

DisplayTextLine2.ItemChecked  = 2 (2 expected)

*** Outputs for Testcase 166

DisplayTextLine2.ItemChecked  = 2 (2 expected) (2)

(3) --- DisplayTextLine2.ItemChecked  = 2 (1 expected)

DisplayTextLine9.ItemChecked  = 2 (2 expected)


; 1)   Test group 1, case a. (1)

; Test case verifies that infrared watchdog is activated by

; startup pulse sequence (2)

; Test requirements section 6.4.3.1.2
```

CASE DEFAULTS : (3)

T_FL_Sys_Fail_Called = 0

T_Update_Time = 1828ACh

T_CMT_Menu_Last_Update = 18639Ch

T_Level_1_Status = 180004h

T_Level_2_Status = 180304h

T_Stop_Method = 0

T_Fault_Report = 1

INPUTS : (3)

num_iterations = 1

entry_procedure = 1

T_NV_Power_On_Count = 1

T_Reset_Value = 0

T_Time_Since_Power_On = 1

OUTPUTS:	*EXPECTED (4)*	*ACTUAL*	*(4) RESULT (5)*
T_NV_Power_On_Count	*= 1*	*1*	*PASS*
T_NV_Power_On_Count_Check	*= 65533*	*65533*	*PASS*
T_BBRAM_Power_On_Count	*= 1*	*1*	*PASS*
T_Time_Since_Power_On	*= 100*	*100*	*PASS*
T_FH_Queue_Msg_Count	*= 2*	*2*	*PASS*
T_Pulse[0].Data[0]	*= 0*	*0*	*PASS*
T_Pulse[0].Data[1]	*= 0*	*10*	*FAIL*

Test case FAILED (6)

ЗАВЕРШАЮЩАЯ ЧАСТЬ ОТЧЕТА о прохождении тестов содержит краткую итоговую информацию о выполнении всех тестовых примеров, по которым составлялся отчет. Обычно эта часть отчета содержит следующую информацию:

1. Общее количество выполненных тестовых примеров.
2. Количество успешно пройденных тестовых примеров.
3. Количество неуспешно пройденных тестовых примеров.
4. Общее количество проверенных выходных значений.
5. Количество выходных значений, у которых ожидаемое значение не совпало с реальным значением.

Ниже приведен пример этой части отчета [Синицын 2006].

> *TEST RESULTS:*
>
> *No. of Test Cases Failed : 0 (3)*
>
> *No. of Test Cases Passed : 45 (2)*
>
> *Total No. of Tests Included : 46 (1)*
>
> *Total No. of Outputs Checked : 2783 (4)*
>
> *No. of failed Outputs Checks : 128 (5)*

Часто в отчет о выполнении тестов кроме количественной статистики помещают раздел с подробным объяснением причин неуспешно пройденных тестовых примеров. Каждый пункт такого объяснения обычно содержит следующую информацию:

1. Идентификаторы тестовых примеров, благодаря неуспешному выполнению которых выявлена проблема.
2. Ссылка на разделы требований, по которым написаны тесты.
3. Ссылка на участки программного кода, в котором выявлена проблема.
4. Описание сути проблемы и (опционально) возможные пути ее решения с точки зрения тестировщика.

Данный раздел может служить основой для создания отчетов о проблемах, либо частично заменять их. Пример такого раздела приведен [Синицын 2006]:

> *TEST CASES WITH FAILURES SUMMARY*
>
> *==========*
>
> *testcases | failures total | explanation in section*
>
> *----------+----------+----------*
>
> *(1) 11b-l,n-p; 12b-d | 67 | 1*
>
> *----------+----------+----------*
>
> *total | 67 |*
>
> *1.*
>
> *LOCATION:*

PR_IR_DATA.C, lines 1323, 1347; (3)

Software requirements section 7.4.5.5; (2)

Test requirements section 7.4.8; (2)

PROBLEM:

Test requirements are not changed, but Software requirements are updated to reflect new system functionality. (4)

DEMONSTRATION:

Test cases: 11 b-l, n-p; 12 b-d (1)

PROPOSED SOLUTION:

Update test requirements section 7.4.8 to meet software requirements section 7.4.5.5. (4)

АВТОМАТИЧЕСКОЕ И РУЧНОЕ ТЕСТИРОВАНИЕ

Не все тесты могут быть выполнены в автоматическом режиме и поэтому требуют ручной работы тестировщика по их выполнению. Результаты выполнения ручных тестов могут заноситься в тот же самый документ, что и результаты выполнения автоматических тестов. Часто это делается, если и автоматические и ручные тесты проверяют одну и ту же функциональную часть тестируемой системы. В этом случае при генерации отчета о прохождении тестов для ручных тестов генерируется форма, в которую тестировщик заносит данные о результатах проведенного им ручного тестирования. Само ручное тестирование может заключаться либо в выполнении тестового сценария, заданного в тест-плане, либо в экспертном анализе участков программного кода системы, которые не могут быть выполнены при автоматическом тестировании на тестовом стенде.

Форма для ручного тестирования содержит следующую информацию:

1. Идентификатор ручного тестового примера.
2. Описание сценария ручного теста или задачи экспертного анализа.
3. Имя лица, проводившего ручное тестирование.
4. Версии требований, на основании которых проводилось ручное тестирование.
5. Ссылки на участки программного кода, для которого проводится ручное тестирование.
6. Результат тестирования. Информацию о соответствии программного кода требованиям – соответствует или не соответствует.
7. Информацию о потенциально возможных проблемах внутри допустимого диапазона значений и за его пределами.
8. Информацию о возможности покрытия тестируемого вручную программного кода при достижении условий, указанных в требованиях.
9. Информацию об итоговом результате ручного тестового примера – успешно/неуспешно.

Ниже приведен пример заполненной формы для ручного тестирования [Синицын 2006]. Красными цифрами в скобках выделены соответствующие пункты приведенного списка, зеленым выделен текст, вводимый в форму тестировщиком.

****

Manual Analysis of Testcase 1 (1): verify that the IR scan goes at a 10Hz rate. (2)

****

1. Activity Description: Independent Code Analysis.

Tester Name: Petrov P. (3)

Pass Criteria: Tester name is not the same as programmer name

2. Activity Description: Name and CM Version of file under review

Module and/or Function Name: IRDA.C	*CM Version: 56 (4,5)*
Document chapter: 7.8.5	*CM Version: 12 (4)*

Pass criteria: All documents exists in respective versions

3. Activity Description: Requirements Implemented.

Identify which lines of code in the module under review incorporate requirements

PassCriteria: Code implements the requirements as defined in the requirement document

Result (Pass/Fail): Pass (6)

4. Activity Description: Code Robustness

Identify line numbers and give a brief description on the software design to handle the following cases

Analysis for at, Inside boundary : The variable refreshRate is set to 10 at line 235 of IRDA.C and later used in IR_Init() function to set NVRAM values on line 590. (5,7)

Analysis for out-of-boundary (robustness) : N/A (7)

5. Activity Description: Structural Coverage Analysis

PassCriteria: Software and software structures (when applicable) are accessible under conditions specified by requirements

Y/N: YES (8)

6. Activity Description: Overall manual test result

Result (Pass/Fail): Pass (9)

ОТЧЕТЫ О ПРОБЛЕМАХ

Каждое несоответствие с требованиями, найденное тестировщиком, должно быть документировано в виде отчета о проблеме. Вероятность обнаружения и исправления ошибки, вызвавшей это несоответствие, зависит от того, насколько качественно она документирована.

ОТЧЕТЫ О ПРОБЛЕМАХ могут поступать не только от тестировщиков, но и от специалистов технической поддержки или пользователей, однако их общая цель – указать на наличие проблемы в системе, которая должна быть устранена.

ОТЧЕТ О ПРОБЛЕМЕ – один из самых важных документов в цепочке тестовой документации.

Главное, что должно быть включено в отчет об ошибке, это:

- ✓ **СПОСОБ ВОСПРОИЗВЕДЕНИЯ ПРОБЛЕМЫ**. Для того, чтобы разработчик смог устранить проблему, он должен разобраться в ее причинах, самостоятельно воспроизведя ее (и, возможно, не один раз).
- ✓ **АНАЛИЗ ПРОБЛЕМЫ С КРАТКИМ ЕЕ ОПИСАНИЕМ**. Описание должно проводиться в тех же терминах, в которых составлены требования на часть системы, в которой обнаружена проблема. В этом случае минимизируется вероятность недопонимания сути проблемы.

Любой отчет о проблеме должен быть составлен **НЕМЕДЛЕННО** после ее обнаружения. Если отчет будет составлен спустя значительное время, повышается вероятность того, что в него не попадет какая-либо важная информация, которая поможет устранить причину проблемы в кратчайшие сроки.

СТРУКТУРА ОТЧЕТОВ О ПРОБЛЕМАХ

Структура отчёта о проблеме в целом мало различается в различных проектах, изменения обычно касаются только порядка и имен следования полей. Некоторые поля могут отражать специфику данного конкретного проекта, однако обычно эти поля следующие.

1. **ОБЪЕКТ, В КОТОРОМ НАЙДЕНА ПРОБЛЕМА**. Здесь помещается максимально полная информация – для документации это название документа, раздел, автор, версия. Для исходных текстов это имя модуля, имя функции/метода или номера строк, версия.
2. **ВЫПУСК И ВЕРСИЯ СИСТЕМЫ**. Определяет место, откуда был взят объект с обнаруженной проблемой. Обычно требуется отдельная идентификация версии системы (а не только версии исходных текстов), поскольку может возникнуть путаница с повторно выявленными проблемами.
3. **ТИП ОТЧЁТА**.
   - ✓ Ошибка кодирования – код не соответствует требованиям.
   - ✓ Ошибка проектирования – тестировщик не согласен с проектной документацией.
   - ✓ Предложение – у тестировщика возникла идея, как можно усовершенствовать код.
   - ✓ Расхождение с документацией – поведение программного обеспечения не соответствует руководству пользователя или проектной документации, или вообще нигде не описано. При этом у тестировщика нет оснований объявлять, где именно находится ошибка.
   - ✓ Взаимодействие с аппаратурой – неверная диагностика плохого состояния устройства, ошибка в интерфейсе с устройством.
   - ✓ Вопрос – тестировщик не уверен, что это проблема и ему требуется дополнительная информация.
4. **СТЕПЕНЬ ВАЖНОСТИ**. Строгого критерия определения степени важности не существует и обычно это поле кодируют от 1 (незначительно) до 10 (фатально).

5. **СУТЬ ПРОБЛЕМЫ**. Краткое (не более 2 строчек) определение проблемы. Даже если две проблемы очень похожи, их описания должны различаться.
6. **МОЖНО ЛИ ВОСПРОИЗВЕСТИ ПРОБЛЕМНУЮ СИТУАЦИЮ?** Ответ: Да, Нет, Не всегда. Если проблема носит нерегулярный характер, то нужно описывать, когда она проявляется, а когда – нет.
7. **ПОДРОБНОЕ ОПИСАНИЕ ПРОБЛЕМЫ И СПОСОБ ЕЁ ВОСПРОИЗВЕДЕНИЯ**. При этом нужны подробности – и в описании условий воспроизведения, и в описании причины объявления получившейся реакции ошибкой.

Вид отчета о проблеме, соответствующего данной структуре, следующий:

ОТЧЁТ О ПРОБЛЕМЕ

Порядковый номер отчёта:

Автор отчёта: _____ Дата создания отчёта: __ __ __

Документы/разделы, связанные с проблемой:

……..

Идентификация объекта/процесса, где проявляется проблема:

…….

Определение проблемы:

…….

Автор решения: _____ Дата формирования решения __ __ __

Принятое решение:

(возможно, ссылки на изменяемые компоненты/запросы на изменения)

…….

Результаты анализа, определяющие, на содержание каких компонент

влияет решение:

…….

План проверок, восстанавливающих текущее состояние документов разработки

…….

Оценка принятого решения автором отчёта о проблеме: _

…….

(0 = полностью согласен

1 = не все аспекты проблемы учтены/разрешены

2 = основная часть проблемы осталась неразрешённой

3 = решение не адекватно проблеме – не устраняет её)

ДОКУМЕНТИРОВАНИЕ И ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА

Каждый дефект, обнаруженный в процессе тестирования, должен быть задокументирован 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:

1. Наименование подсистемы, в которой обнаружен дефект.
2. Версия продукта (номер build), на котором дефект был найден.
3. Описание дефекта.
4. Описание процедуры (шагов, необходимых для воспроизведения дефекта).
5. Номер теста, на котором дефект был обнаружен.
6. 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 в состояние "Resolved". При этом разработчик должен указать следующую информацию:

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

После этого тестировщик проверяет, исправлен ли был дефект 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.

ОТСЛЕЖИВАНИЕ ОШИБОК

Проблема или ошибка может быть рассмотрена как некоторый объект в системе, который может находиться в одном из восьми состояний.

- ✓ Проблема или ошибка зарегистрирована.
- ✓ Выполнен первичный анализ (эксперт подтвердил факт наличия ошибки).
- ✓ Понятна причина, вызвавшая ошибку.
- ✓ Выполнен окончательный анализ проблемы или ошибки.
- ✓ Принято решение о начале работы над исправлением.
- ✓ Выполнено исправление ошибки.
- ✓ Исправление вошло в состав программного продукта.
- ✓ Подтверждено исправление ошибки.

Обратим внимание на то, что текст на обнаруженную проблему или ошибку должен быть обязательно помещен в тестовую базу данных. В результате получается регрессионная тестовая сюита (сценарий). Регрессионное тестирование использует регрессионные тестовые сюиты (сценарии), чтобы убедиться в том, что однажды исправленная ошибка не появилась вновь.

ОЦЕНКА КАЧЕСТВА ТЕСТИРУЕМОГО КОДА. СТАТИСТИКА

В результате выполнения каждого тестового примера тестовое окружение сравнивает ожидаемые и реальные выходные значения. Если эти значения совпадают – тест считается пройденным, в противном случае – тест не пройден.

Каждый из не пройденных тестов указывает на потенциальный дефект в тестируемой системе, а общее их количество позволяет оценивать качество тестируемого программного кода и объем изменений, которые необходимо в него внести для устранения дефектов.

Для построения такой интегральной оценки после выполнения всех тестовых примеров тестовым окружением собирается статистика выполнения, которая, как правило, записывается в файл отчета о выполнении тестов.

Существует несколько степеней подробности статистики выполнения тестов.

1. Вывод количества пройденных и не пройденных тестовых примеров, а также их общего количества [Синицын 2006].

> *180 test cases passed*
>
> *20 test cases failed*
>
> *200 test cases total*

2. Вывод количества пройденных и не пройденных тестовых примеров, их общего количества и идентификаторов не пройденных тестовых примеров. Позволяет локализовать тестовые примеры, выявившие дефект [Синицын 2006].

> *Invoking test case 1 … Passed*
>
> *Invoking test case 2 … Failed*
>
> *Invoking test case 3 … Failed*
>
> *<…>*
>
> *Invoking test case 200 … Passed*
>
> *Final stats:*
>
> *180 test cases passed*
>
> *20 test cases failed*
>
> *200 test cases total*

3. Вывод количества пройденных и не пройденных тестовых примеров, их общего количества, идентификаторов не пройденных тестовых примеров и вывод не совпавших ожидаемых и реальных

выходных данных. Позволяет проводить более глубокий анализ причин неуспешного прохождения теста [Синицын 2006].

```
Invoking test case 1 … Passed
---
Invoking test case 2 … Failed
Expected values:                Actual values:
  A = 200                       A = 0
  B = 450                       B = 0
  Message = “Submenu 1”         Message = “”
---
Invoking test case 3 … Failed
Expected values:                Actual values:
  A = 0                                 A = 200
  B = 0                                 B = 300
  Message = “”                  Message = “Main Menu”
---
<…>


Invoking test case 200 … Passed
---
Final Stats
180 test cases passed
20 test cases failed
200 test cases total
```

4. Вывод количества пройденных и не пройденных тестовых примеров, их общего количества, идентификаторов не пройденных тестовых примеров и вывод всех ожидаемых и реальных выходных данных [Синицын 2006].

```
Invoking test case 1 … Passed
---
Invoking test case 2 … Failed
```

```
Expected values:          Actual values:
  A = 200                 A = 0                    FAIL
  B = 450                 B = 0                    FAIL
  C = 500                 C = 500                  P
  D = 600                 D = 600                  P
  Message = "Submenu 1"   Message = ""             FAIL
---
Invoking test case 3 … Failed
Expected values:          Actual values:
  A = 0                   A = 200                  FAIL
  B = 0                   B = 300                  FAIL
  C = 500                 C = 500                  P
  D = 600                 D = 600                  P
  Message = ""            Message = "Main Menu"    FAIL
---
<…>
Invoking test case 200 … Passed
---


Final Stats
180 test cases passed
20 test cases failed
200 test cases total
```

5. Полный вывод ожидаемых и реальных выходных данных с отметками о совпадении и несовпадении и отметками об успешном/неуспешном завершении для каждого тестового примера [Синицын 2006].

```
Invoking test case 1 … Passed
  A = 0                   A = 0                    P
  B = 0                   B = 0                    P
  C = 500                 C = 500                  P
  D = 600                 D = 600                  P
```

```
  Message = “”                Message = “”              P
Invoking test case 2 … Failed
Expected values:            Actual values:
  A = 200                   A = 0                     FAIL
  B = 450                   B = 0                     FAIL
  C = 500                   C = 500                   P
  D = 600                   D = 600                   P
  Message = “Submenu 1”     Message = “”              FAIL
Invoking test case 3 … Failed
Expected values:            Actual values:
  A = 0                          A = 200              FAIL
  B = 0                          B = 300              FAIL
  C = 500                   C = 500                   P
  D = 600                   D = 600                   P
  Message = “”              Message = “Main Menu”     FAIL
<…>
Invoking test case 200 … Passed
  Message = “Submenu 1”     Message = “Submenu 1      P
  Prompt  = “>”             Prompt = “>”              P
Final Stats
180 test cases passed
20 test cases failed
200 test cases total
```

ЛИТЕРАТУРА

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