Unit Testing as a Cornerstone of SAS® Application Development

Dante Di Tommaso
Unit Test Foundation

Topics

• Definitions

• Key Concepts
  – Application Code Base
  – Test Code Base
  – Test Coverage

• Examples
Unit Testing

Definitions

  - Test actual against predetermined, expected results

- Wikipedia (today)
  - Verify the functionality of a specific section of code
  - Include the constructors and destructors (minimum)
  - "White-box" tests, written by developers as they work on code

- In Addition
  - Manage as a code base, separate from application
  - Simple and fast
  - Repeatable
Unit Testing

Definitions

• Howard N, Gayari M (2004)
  – Test actual against predetermined, expected results

• Wikipedia (today)
  – Verify the functionality of a specific section of code
  – Include the constructors and destructors (minimum)
  – "White-box" tests, written by developers as they work on code

• In Addition
  – Manage as a code base, separate from application
  – Simple and fast
  – Repeatable
Unit Testing

Concepts

  - Actual / Expected
- Wikipedia (today)
  - Specific code
  - Constructors / Destructors
  - White-box tests
- In Addition
  - Code base
  - Simple / fast
  - Repeatable
- Predictable understanding of
  - Inputs
  - Outputs
  - Likely use ...
  - ... including limitations
- Clearly defined agreed, consistent behaviour
Unit Testing Concepts

• Howard N, Gayari M (2004)
  – Actual / Expected

• Wikipedia (today)
  – Specific code
    – Constructors / Destructors
    – White-box tests

• In Addition
  – Code base
  – Simple / fast
  – Repeatable

• Clear, focused scope
  – well-defined functionality
  – transparent usage scope

• Identify behaviour to protect with unit tests
Unit Testing Concepts

  - Actual / Expected
- Wikipedia (today)
  - Specific code
  - Constructors / Destructors
  - White-box tests
- In Addition
  - Code base
  - Simple / fast
  - Repeatable
- Tests are independent
  - One test is not influenced by last test ...
  - And does not influence next test
- Each test sets up and cleans up the test environment
Unit Testing

Concepts

  - Actual / Expected
- Wikipedia (today)
  - Specific code
  - Constructors / Destructors
  - White-box tests
- In Addition
  - Code base
  - Simple / fast
  - Repeatable

- Knowledge of the Application Code influences the Test Code
  - Developer tests specific conditions and actions ...
  - Valid parameter values ...
  - Control of invalid parameter values ...
  - Interactions of parameters ...

- Test design based on code structure and internal logic
- White-box tests provide valuable design feedback
Unit Testing Concepts

  - Actual / Expected
- Wikipedia (today)
  - Specific code
  - Constructors / Destructors
  - White-box tests
- In Addition
  - Code base
  - Simple / fast
  - Repeatable
- Manage Test Code as an independent code base
  - Manage as you do the Application Code
  - Consistent construction conventions
  - Programming guidelines
  - Simple to
    - Read,
    - Critique,
    - Update,
    - Extend
  - Simple to ... MAINTAIN
Unit Testing

Concepts

  - Actual / Expected
- Wikipedia (today)
  - Specific code
  - Constructors / Destructors
  - White-box tests
- In Addition
  - Code base
  - Simple / fast
  - Repeatable
- Simple and Fast in every possible way!
  - Simple to read ... spare the cleverness
  - Simple to modify ... extend tests
  - Simple to extend ... protect new functionality
- Easy, fast, rewarding testing
Unit Testing

**Concepts**

  - Actual / Expected
- Wikipedia (today)
  - Specific code
  - Constructors / Destructors
  - White-box tests
- In Addition
  - Code base
  - Simple / fast
  - Repeatable

- Test early
- Test often
- It's your software. Test it. - testoften.com
- Confidence is good. Certainty is better.
Unit Testing

*The Challenge and the Art*

- Focus on the tests, not secondary activities
  - Strategic tests
  - We all know collection is not perfect
  - We all know to expect surprises

- Test Space and Coverage
  - Complete test coverage when feasible ...
  - ... recognize when it is not,
  - and test what matters most
Unit Testing

The Challenge and the Art

- Interface test space
  - "Black-box" tests: Explicit tests of inputs/outputs. Middle is irrelevant.
  - Input validity: Value (simple checks) vs. Content (data integrity checks)
  - Intended users: Internal developers vs. Black-box end-users
  - Intended usage: What comes before/after? Macro quoting required?

- Implementation test space
  - "White-box" tests: Explicit tests of code blocks
  - Design feedback: Redundant tests can expose sub-optimal design
  - Scope feedback: Unrelated tests can expose "scope creep", loss of focus
Unit Testing

The Challenge and the Art

- Reduce efforts for secondary tasks
  - Test definition
  - Execution
  - Collection
  - Assessment
  - Reporting

- %PASSFAIL standardises these secondary activities
Test Driven Development
Before standardization

```sas
/** MACRO DEFINITION **/
%macro count_words(  string
    delimiter = %str( )
);%let result = .;
&result
%mend count_words;

/** TEST STRATEGY **/
data _null_;
array exp_cnt [4] _temporary_   ( 4 3 1 0 );
array rsl_cnt [4] _temporary_;
rsl_cnt[1] = %count_words(I am.number 4, delimiter = %str( .));
rsl_cnt[2] = %count_words(Alt-Alt Alt, delimiter = %str( -));
rsl_cnt[3] = %count_words(-One-, delimiter = %str( -));
rsl_cnt[4] = %count_words(%str( ), delimiter = %str( ));

do idx = 1 to dim(exp_cnt);
  if rsl_cnt[idx] = exp_cnt[idx] then
    put 'PASS test ' idx;
  else put 'FAIL test ' idx;
end;
run;
```

FAIL test 1
FAIL test 2
FAIL test 3
FAIL test 4
Test Driven Development

Before standardization

- Test implementation deficiencies
  - How many tests should have run?
  - What is the purpose of each test?
  - What do results indicate?

- Maintainable?

- Extensible?

```plaintext
/*** TEST STRATEGY ***/
data _null_;
array exp_cnt [4] _temporary_   ( 4 3 1 0 );
array rsl_cnt [4] _temporary_;
  rsl_cnt[1] = %count_words(I am.number 4, delimiter = %str( .));
  rsl_cnt[2] = %count_words(Alt-Alt Alt,   delimiter = %str( -));
  rsl_cnt[3] = %count_words(-One-,         delimiter = %str( -));
  rsl_cnt[4] = %count_words(%str( ),       delimiter = %str( ));
  
  do idx = 1 to dim(exp_cnt);
    if rsl_cnt[idx] = exp_cnt[idx] then
      put 'PASS test ' idx;
    else put 'FAIL test ' idx;
  end;
run;
```
Test Driven Development

After standardization

- Test-driven Development

... After

```plaintext
data testdata;
attrib test_id length=$12 label='Test ID for COUNT_WORDS';
attrib test_dsc length=$80 label='Test Description';
attrib test_mac length=$32 label='Name of macro to test';
attrib test_type length=$10 label='Test Type < M | S | D | I >';
attrib pparm_string length=$50 label='Test values for 1st pos parm';
attrib kparm_delimiter length=$50 label='Test values for 1st keywd parm';
attrib test_expect length=$100 label='EXPECTED test results';
attrib test_expect_sym length=$100 label='EXPECTED global syms created';
test_id = 'CW.01.01';
test_dsc = 'Missing value parameter';
test_mac = 'count_words';
test_type = 'M';
pparm_string = '_NULLPARM_';
kparm_delimiter = '_NULLPARM_';
test_expect = '';
test_expect_sym = 'countw=0|longw=0' OUTPUT;
...
run;
%passfail(testdata);
```
Test Driven Development

After standardization

- Test-driven Development
- $\text{PASSFAIL}$ automates 2ry tasks

```plaintext
data testdata;
  attrib test_id     length=$12 label='Test ID for COUNT_WORDS';
  attrib test_dsc    length=$80 label='Test Description';
  attrib test_mac    length=$32 label='Name of macro to test';
  attrib test_type   length=$10 label='Test Type < M | S | D | I >';
  attrib pparm_string length=$50 label='Test values for 1st pos parm';
  attrib kparm_delimiter length=$50 label='Test values for 1st keywd parm';
  attrib test_expect length=$100 label='EXPECTED test results';
  attrib test_expect_sym length=$100 label='EXPECTED global syms created';

  test_id   = 'CW.01.01'                         ;
  test_dsc  = 'Missing value parameter'           ;
  test_mac  = 'count_words'                       ;
  test_type = 'M'                                ;
  pparm_string = '_NULLPARM_'                     ;
  kparm_delimiter = '_NULLPARM_'                  ;
  test_expect = ''                               ;
  test_expect_sym = 'countw=0|longw=0'           ; OUTPUT;

... run;

%PASSFAIL ( TESTDATA ) ;
```
Test Driven Development

After standardization

- Test-driven Development
- After
- Standard Description of tests

```plaintext
data testdata;
attrib test_id length=$12 label='Test ID for COUNT_WORDS';
attrib test_dsc length=$80 label='Test Description';
attrib test_mac length=$32 label='Name of macro to test';
attrib test_type length=$10 label='Test Type < M | S | D | I >';
attrib pparm_string length=$50 label='Test values for 1st pos parm';
attrib kparm_delimiter length=$50 label='Test values for 1st keywd parm';
attrib test_expect length=$100 label='EXPECTED test results';
attrib test_expect_sym length=$100 label='EXPECTED global syms created';

test_id = 'CW.01.01';
test_dsc = 'Missing value parameter';

test_mac = 'count_words';
test_type = 'M';
pparm_string = '_NULLPARAM_';
kparm_delimiter = '_NULLPARAM_';

test_expect = ' ';
test_expect_sym = 'countw=0|longw=0' ; OUTPUT;
...
run;
%passfail(testdata);
```
Test Driven Development

After standardization

- Test-driven Development ...
- Standard, Transparent Definition of Each Test

```plaintext
data testdata;

attrib test_id length=$12 label='Test ID for COUNT_WORDS';
attrib test_dsc length=$80 label='Test Description';

attrib test_mac length=$32 label='Name of macro to test';
attrib test_type length=$10 label='Test Type < M | S | D | I >';
attrib pparm_string length=$50 label='Test values for 1st pos parm';
attrib kparm_delimiter length=$50 label='Test values for 1st keywd parm';

attrib test_expect length=$100 label='EXPECTED test results';
attrib test_expect_sym length=$100 label='EXPECTED global syms created';

test_id = 'CW.01.01' ;
test_dsc = 'Missing value parameter' ;

test_mac = 'count_words' ;
test_type = 'M' ;
pparm_string = '_NULLPARM_' ;
kparm_delimiter = '_NULLPARM_' ;


test_expect = ' ' ;
test_expect_sym = 'countw=0|longw=0' ;

... run;

%passfail(testdata);
```
Test Driven Development

After standardization

- Test-driven Development
  - ... After
- Standard Declaration of Expected Results

```r
data testdata;

attrib test_id    length=$12 label='Test ID for COUNT_WORDS';
 attrib test_dsc   length=$80 label='Test Description';
 attrib test_mac   length=$32 label='Name of macro to test';
 attrib test_type  length=$10 label='Test Type < M | S | D | I >';
 attrib pparam_string length=$50 label='Test values for 1st pos parm';
 attrib kparam_delimiter length=$50 label='Test values for 1st keywd parm';
 attrib test_expect length=$100 label='EXPECTED test results';
 attrib test_expect_sym length=$100 label='EXPECTED global syms created';

test_id  = 'CW.01.01' ;
test_dsc = 'Missing value parameter' ;
test_mac = 'count_words' ;
test_type = 'M' ;
pparm_string = '_NULLPARAM_' ;
kparam_delimiter = '_NULLPARAM_' ;
test_expect = '' ;
test_expect_sym = 'countw=0|longw=0' ; OUTPUT;
...
run;

%passfail(testdata);
```
Test Driven Development

After standardization

- Test-driven Development ... After
- Standardized Execution, Evaluation, Reporting of tests

FAIL - CW.01.03, Non-missing string, default delim

Expected:
Returned:
Expected Global Syms: countw=3|longw=9
Returned Global Syms: countw=4|longw=9

### -----
###
### COUNT_WORDS Overall: ALL TESTS PASSED - 3 but Failed - 1 test(s)!
###
### -----

001   CW.01.01    Missing value parameter
002   CW.01.02    Basic call, duplicate words
003   CW.01.03    Non-missing string, default delim
004   CW.01.04    Non-missing string, multiple delims
Automated Test Frameworks

By Example

• Out of scope for this paper

• Open source SAS test frameworks are available
  – FUTS: ThotWave's Framework for Unit Testing SAS
  – SCLUnit: Scocca's framework
  – SASUnit: HMS Analytical Software's framework

• SASCommunity.org
  – FUTS
  – SCLUnit

• SourceForge.net
  – SASUnit