Why aren't my tests stable?
Test Isolation with the ABAP Unit Framework

Michael Sauter, SAP SE
September 7th 2023

Public

Disclaimer

The information in this presentation is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. Except for your obligation to protect confidential information, this presentation is not subject to your license agreement or any other service or subscription agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or any related document, or to develop or release any functionality mentioned therein.

This presentation, or any related document and SAP's strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by SAP at any time for any reason without notice. The information in this presentation is not a commitment, promise or legal obligation to deliver any material, code or functionality. This presentation is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This presentation is for informational purposes and may not be incorporated into a contract. SAP assumes no responsibility for errors or omissions in this presentation, except if such damages were caused by SAP's intentional or gross negligence.

All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

Motivation – Test Automation

- Manual tests need time

- Manual tests have gaps

- Protection from regressions

- More fun in development

- …

Motivation – Test Isolation

- Simulation of error cases

- Simulation of user input

- Easier analysis of errors

- Shorter runtimes

- Unwanted output of used components

- …

Definition of Used Terms

Code Under Test (CUT)

Depended-On Component (DOC)

Unit Tests

Test Double

Injection

```
┌─────────────┐          ┌─────────────┐          ┌──────────────┐
│             │          │ Code Under  │          │ Depended-On  │
│    Test     │─────────▶│    Test     │─────────▶│  Component   │
│             │          │             │          │              │
└─────────────┘          └─────────────┘          └──────────────┘
```

ABAP Unit

Framework integrated in the ABAP Language

Intended to be used for unit tests

Part of the xUnit family

```abap
1    CLASS ltc_add DEFINITION
2      FOR TESTING
3      RISK LEVEL HARMLESS
4      DURATION SHORT.
5
6      PRIVATE SECTION.
7        METHODS one_plus_one_is_two FOR TESTING.
8    ENDCLASS.
9
10   CLASS ltc_add IMPLEMENTATION.
11     METHOD one_plus_one_is_two.
12
13       "given
14       DATA(cut) = NEW cl_calculator( ).
15       "when
16       DATA(sum) = cut->add( i_summand_1 = 1
17                             i_summand_2 = 1 ).
18       "then
19       cl_abap_unit_assert=>assert_equals( act = sum
20                                           exp = 2 ).
21
22     ENDMETHOD.
23   ENDCLASS.
```
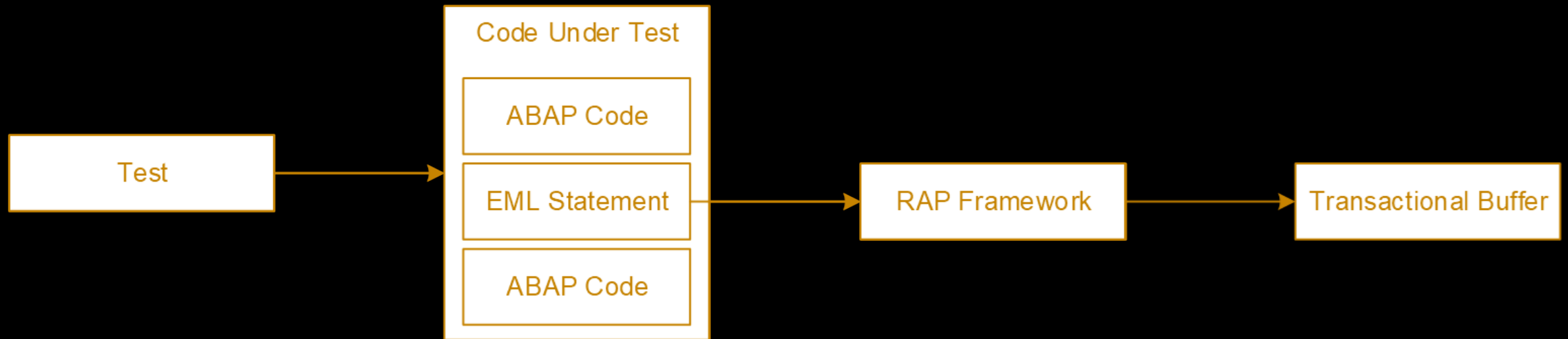
Isolation Techniques

Factory Pattern with Injector
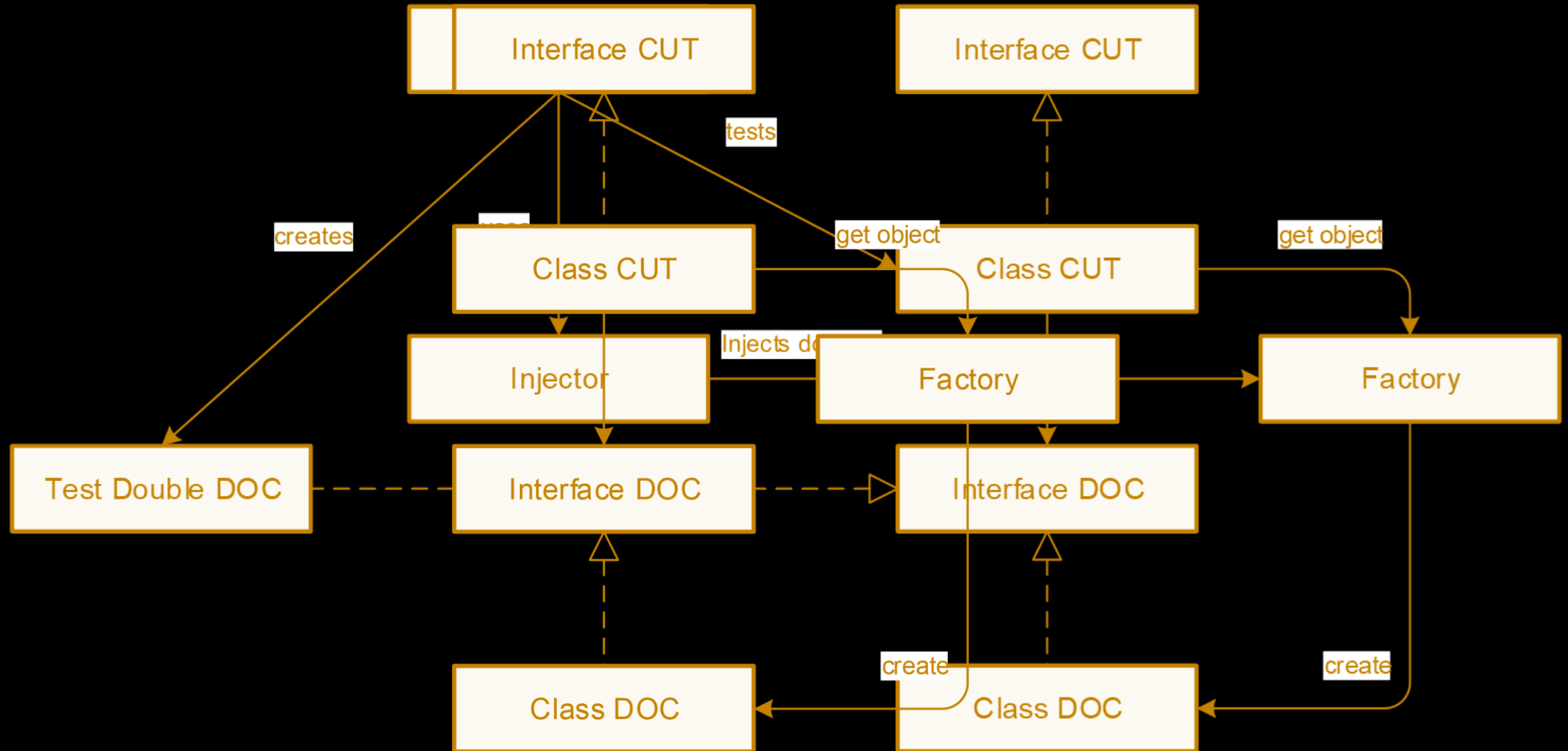
RAP - Transactional Buffer Double
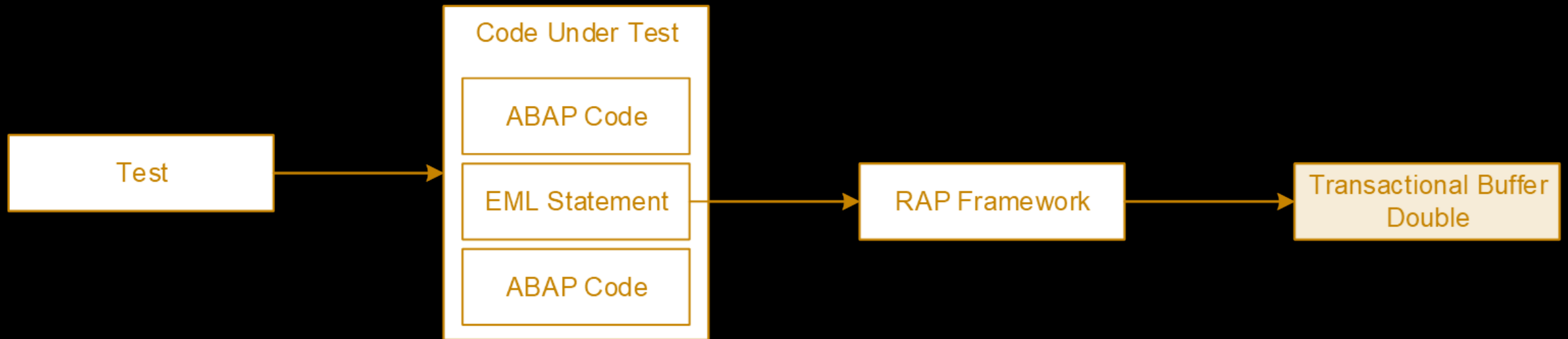
RAP - Mock EML API

Summary

Depended-On Component	Framework
Classes	Self-made test doubles or OO Double Framework
Database tables or CDS Entities	SQL Test Double Framework
Function Modules	Function Module Test Double Framework
Authority Checks	Authority Check Test Helper API
RAP-Business Objects	Transactional Buffer Double or Mock EML API
Database artefacts used in CDS	CDS Test Double Framework

Thank you.

Contact Information:

Michael Sauter
github.com/sautermi0
people.sap.com/michael.sauter

SAP

Self-made ABAP Object Testdoubles

Simulates the expected behaviour

```
 1    CLASS ltd_stub DEFINITION FOR TESTING.
 2      PUBLIC SECTION.
 3        INTERFACES if_cash_provider PARTIALLY IMPLEMENTED.
 4        DATA m_notes TYPE if_cash_provider=>tt_change.
 5    ENDCLASS.
 6
 7    CLASS ltd_stub IMPLEMENTATION.
 8      METHOD if_cash_provider~get_notes.
 9        r_notes = m_notes.
10      ENDMETHOD.
11    ENDCLASS.
```

ABAP Object Oriented Test Double Framework

Helps with creating test double instances

```abap
 1    ...
 2    " step 1: create the test double of interface type
 3    test_double ?= cl_abap_testdouble=>create( 'if_some_global_interface' ).
 4
 5    " step 2: configure the test double behavior
 6    cl_abap_testdouble=>configure_call( test_double )->returning( abap_true
 7                                                    )->ignore_all_parameters(
 8                                                    )->and_expect( )->is_called_once( ).
 9
10    " step 3: configure the test double method
11    test_double->method_to_double( i_parameter_1 = 0
12                                   i_parameter_2 = abap_true ).
13    ...
```

Function Module Test Double Framework

Offers a possibility to simulate output for function modules

```
1    method class_setup.
2      "create function test doubles for the given list of function modules and returns an instance
3      "of test environment. The test environment instance would be used to access the individual test double.
4      function_test_environment = cl_function_test_environment=>create( value #( ( 'FTD_CONVERT_CURRENCY' ) ) ).
5    endmethod.
6
7    method setup.
8      " to configure a test double, first we need to get the double from the test environment
9      data(currency_converter) = function_test_environment->get_double( 'FTD_CONVERT_CURRENCY' ).
10
11     " to configure a test double,
12     " 1. we need to create the input configuration which would contain the expected set of input values on function call
13     " 2. we need to create the output configuration which would contain the output values expected on function call
14     "    for the given input values.
15     " 3. configure test double with input and output test data configuration
16
17     " input test data configuration - for amount 100
18     data(conv_curr_input_config) = currency_converter->create_input_configuration( )->set_importing_parameter( name = 'AMOUNT' value = 100
19                                                                                    )->set_importing_parameter( name = 'SOURCE_CURRENCY' value = 'USD'
20                                                                                    )->set_importing_parameter( name = 'TARGET_CURRENCY' value = 'EUR' ).
21
22     " output test data configuration
23     data(conv_curr_output_config) = currency_converter->create_output_configuration( )->set_exporting_parameter( name = 'TARGET_CURR_AMOUNT' value = 80 ).
24
25     " configure test double to return the values configured in the output configuration "conv_curr_output_config"
26     " when the function module is invoked with the exact list of input arguments defined in "conv_curr_input_config"
27     currency_converter->configure_call( )->when( conv_curr_input_config_1 )->then_set_output( conv_curr_output_config_1 ).
28   endmethod.
```

ABAP SQL Test Double Framework

Replaces the contents of database tables with test data.

Ensures that changes won't affect the productive data.

```abap
1  method class_setup.
2    environment = cl_osql_test_environment=>create( VALUE #( ( 'DB_TABLE' ) ) ).
3  endmethod.
4
5  method setup.
6    environment->insert_test_data( <your_test_data> ).
7  endmethod
```

ABAP CDS Test Double Framework

Replaces the contents of database tables (or used CDS entities) with test data.

```abap
1   method class_setup.
2     environment = cl_cds_test_environment=>create( i_for_entity = 'CDS_VIEW'
3                                                     i_dependency_list = VALUE #( ( type = 'TABLE' name = 'DB_TABLE' ) ) ).
4   endmethod.
5
6   method setup.
7     environment->insert_test_data( <your_test_data> ).
8   endmethod.
```

Classic ABAP Authority Check Test Helper API

Restricts authorizations in tests to check behavior with limited authorizations.

```abap
 1  method setup.
 2    " Define a role with DISPLAY authorizations for authorization object S_DEVELOP.
 3    DATA(role_may_display)  = VALUE cl_aunit_auth_check_types_def=>role_auth_objects(
 4      ( object = 'S_DEVELOP'
 5        authorizations = VALUE #(
 6          ( VALUE #( ( fieldname   = 'ACTVT'
 7              fieldvalues  = VALUE #( ( lower_value = '03' ) ) ) ) )
 8      ) ).
 9
10    DATA(usrrl_may_display)  = VALUE cl_aunit_auth_check_types_def=>user_role_authorizations( ( role_authorizations = role_may_display ) ).
11
12    " Create an auth object set containing display authorizations.
13    DATA(auth_objset_with_disp_auth) = cl_aunit_authority_check=>create_auth_object_set( usrrl_may_display ).
14
15    " Set up environment - Get an instance of the test controller and set the user configurations.
16    DATA(auth_controller) = cl_aunit_authority_check=>get_controller( ).
17
18    " Set up environment - Configure users with the intended authorizations via the auth_objset for the test session.
19    auth_controller->restrict_authorizations_to( auth_objset ).
20  endmethod.
```

Follow us