BOPF Test Framework

SAP AG, 2013

Disclaimer

This presentation outlines our general product direction and should not be relied on in making a purchase decision. This presentation is not subject to your license agreement or any other agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or to develop or release any functionality mentioned in this presentation. This presentation and SAP's strategy and possible future developments are subject to change and may be changed by SAP at any time for any reason without notice. This document 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. SAP assumes no responsibility for errors or omissions in this document, except if such damages were caused by SAP intentionally or grossly negligent.

© 2012 SAP AG. All rights reserved.

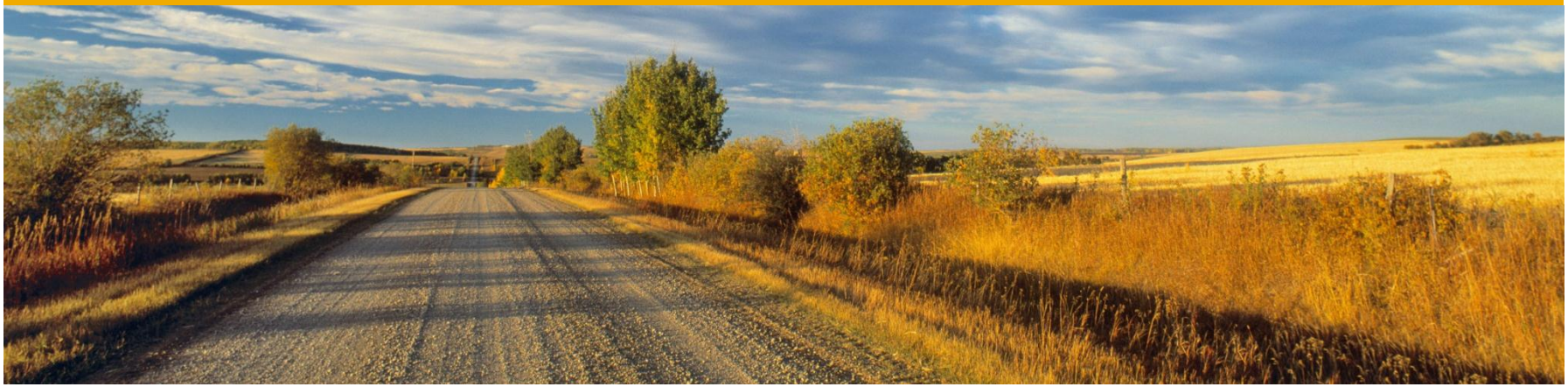
Agenda

Unit Testing

Scenario Testing

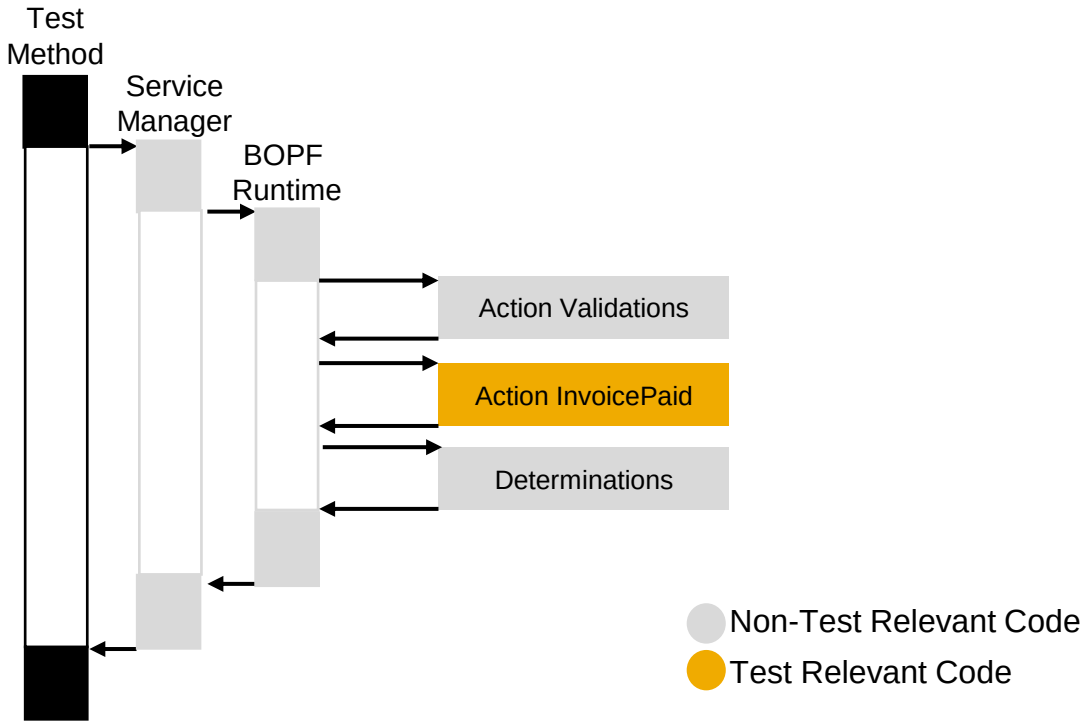
Unit Test Implementation

© 2012 SAP AG. All rights reserved.



Unit Testing

Unit Testing
Introduction



Testing a certain entity (e.g. Action “InvoicePaid”) by calling the Service Manager’s core service executes much coding that is not test relevant.
Thus the test result does not clearly indicate wether there is an issue in the entity or in its environment (e.g. other determinations). For instance, a determination may overwrite changes that are done correctly in the action that is being tested.

© 2012 SAP AG. All rights reserved.

Unit Testing
Definitions

Unit
A unit is the smallest isolatable part of an application that can be tested.
In case of business objects (BOs), a unit is, for instance, an action, a determination or a validation.

Unit Testing

- Testing each unit separately eases the finding of errors
- By testing only the unit but not its environment, tests are more stable
- Unit testing eases the test-driven development approach

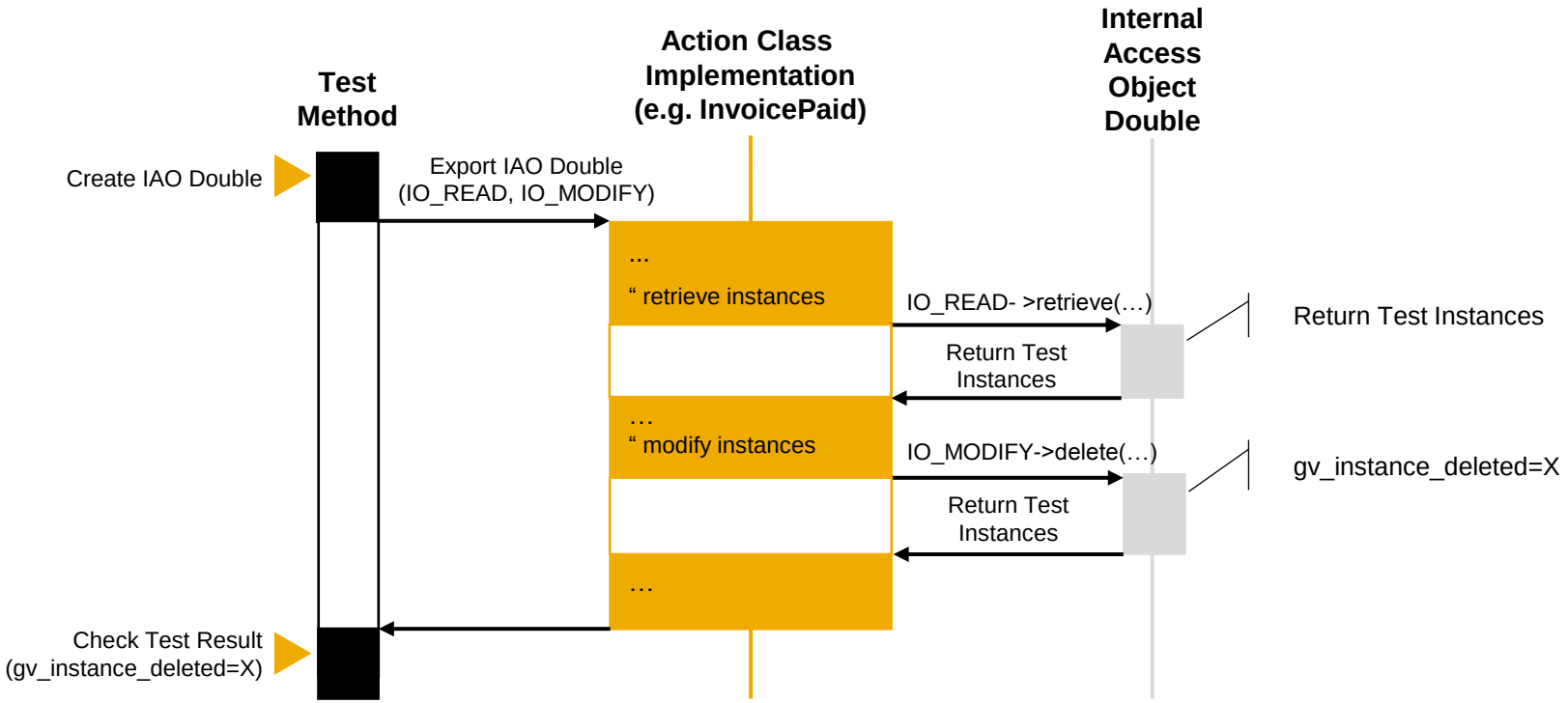
Mock Object / (Test) Double (both terms are used synonymously in the slides)

A mock object is a replacement of a certain object that implements the original interfaces but returns only test data.

© 2012 SAP AG. All rights reserved.

Unit Testing

Mocking the Internal Access Object (IAO)



Usually, an entity implementation accesses the own BO.
For instance, node instances are retrieved. Thus, the internal access objects IAO casted as `IO_READ` and `IO_MODIFY` needs to be mocked to return only test data.

© 2012 SAP AG. All rights reserved.

Unit Testing
Example: Mocking the Internal Access Object (IAO)

Declare and implement the Internal Access Double. Redefine `~retrieve`/`~delete`.

```
CLASS lcl_internal_access_double DEFINITION FOR TESTING
 INHERITING FROM /bobf/cl_tool_test_double_int.
  PUBLIC SECTION.
    METHODS /bobf/if_frw_read~retrieve REDEFINITION.
    METHODS /bobf/if_frw_modify~delete REDEFINITION.
ENDCLASS.
```

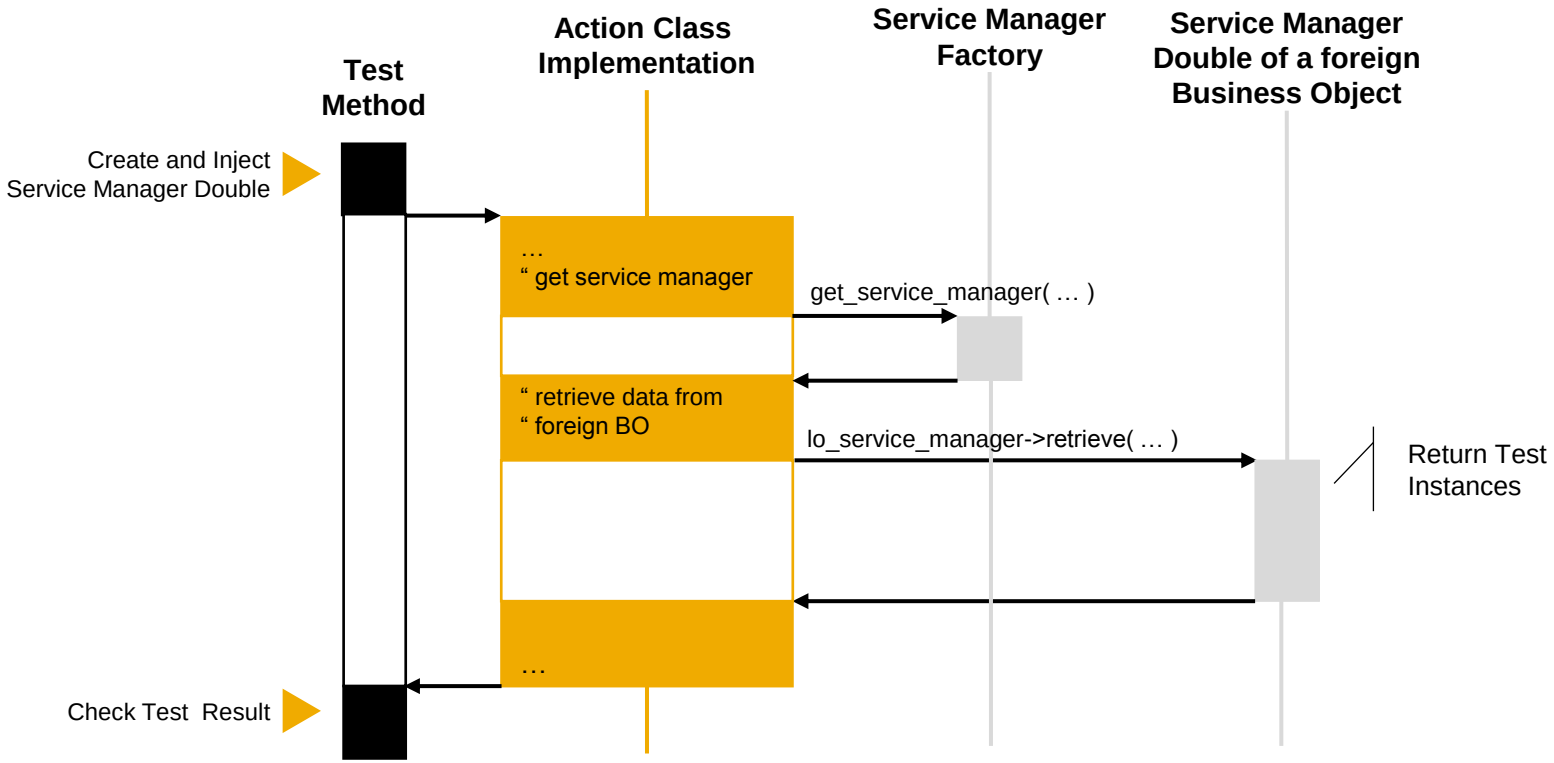
Test the action `Invoice Paid` with the Internal Access Double.

```
METHOD test_invoice_paid.
  ...

  " instantiate the internal access double and cast the interfaces...
  CREATE OBJECT lo_internal_access_double.
  lo_read ?= lo_internal_access_double.
  lo_modify ?= lo_internal_access_double.

  " call/test the action with the internal access double...
  TRY.
      CREATE OBJECT lo_action_invoice_paid.
      lo_action_invoice_paid->/bobf/if_frw_action~execute(
          …
          io_read   = lo_read
          io_modify = lo_modify
      ).
    CATCH /bobf/cx_frw.
  ENDTRY.
  ...
ENDMETHOD.
```

© 2012 SAP AG. All rights reserved.

Unit Testing

Mocking the Service Manager



If the entity (e.g. action), that is being tested, accesses foreign business objects, the corresponding service manager needs to be mocked.
Instead of executing the business logic of the foreign BO, the service manager double only returns test data.

© 2012 SAP AG. All rights reserved.

Unit Testing
Example: Mocking the Service Manager

The productive action code that shall be tested.

```
METHOD /BOBF/IF_FRW_ACTION~EXECUTE.
  ...
  " call retrieve at other BO via Service Manager...
  lo_service_manager_bo2->retrieve( ... ).
```

Declare and implement the service manager double. Redefine `~retrieve`.

```
CLASS lcl_service_manager_double DEFINITION FOR TESTING
 INHERITING FROM /bobf/cl_tool_test_double_sm.
  PUBLIC SECTION.
    METHODS /bobf/if_tra_service_manager~retrieve REDEFINITION.
```

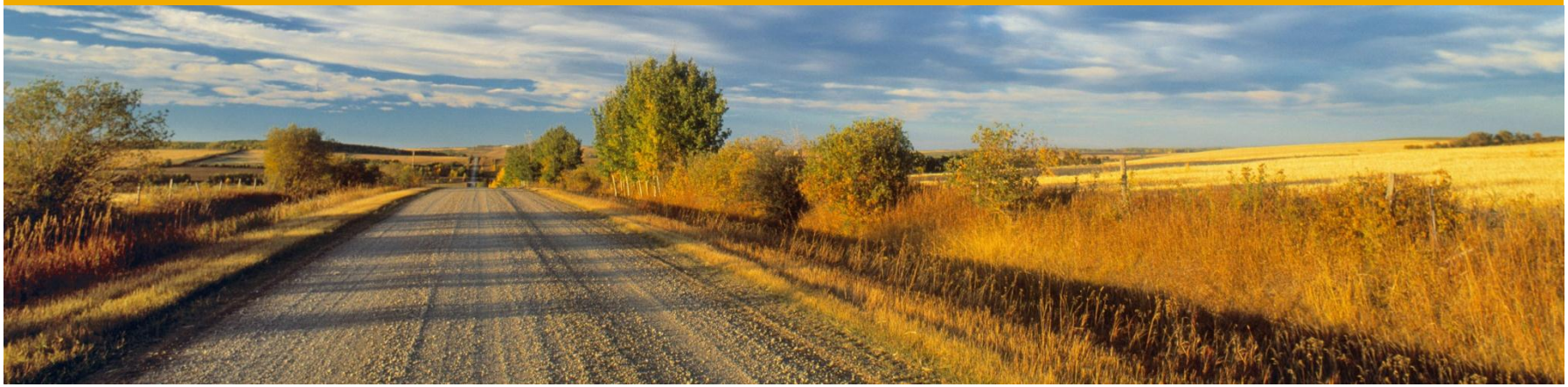
Instantiate and inject the service manager double in test setup phase.

```
METHOD class_setup.
  …
  CREATE OBJECT lo_service_manager_double.
  lo_test_manager->inject_service_manager_double(
    iv_bo_key    = lv_bo2_key
    io_sm_double = lo_service_manager_double ).
  lo_test_manager->activate_injections( ).
```

Test the action that now internally uses now the service manager double.

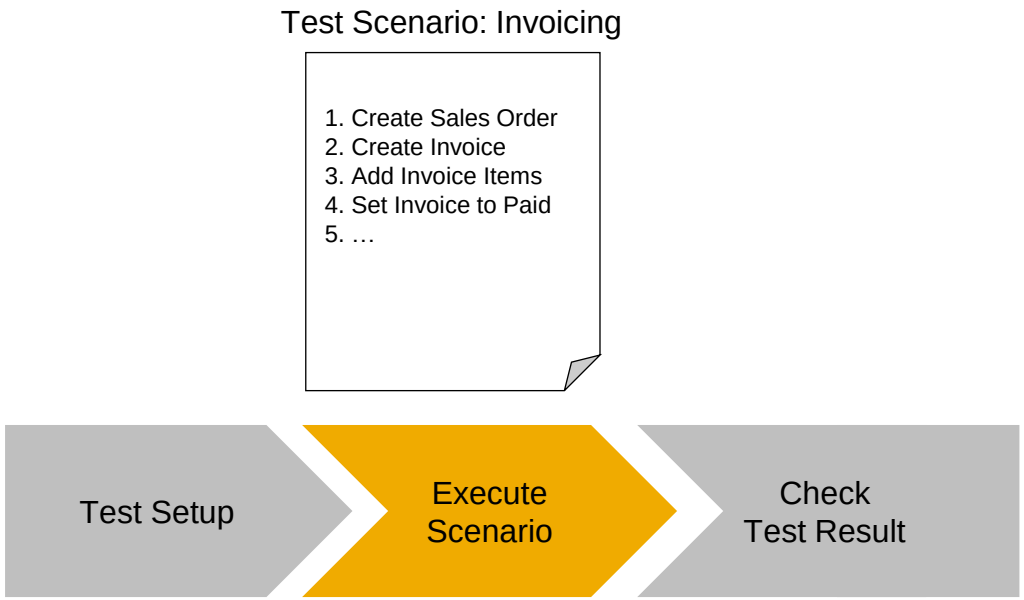
```
METHOD test_action.
  ...
  " call/test the action which internally uses the SM double...
  lo_action->/bobf/if_frw_action~execute( … ).
```

© 2012 SAP AG. All rights reserved.



Scenario Testing

Introduction
Scenario Testing

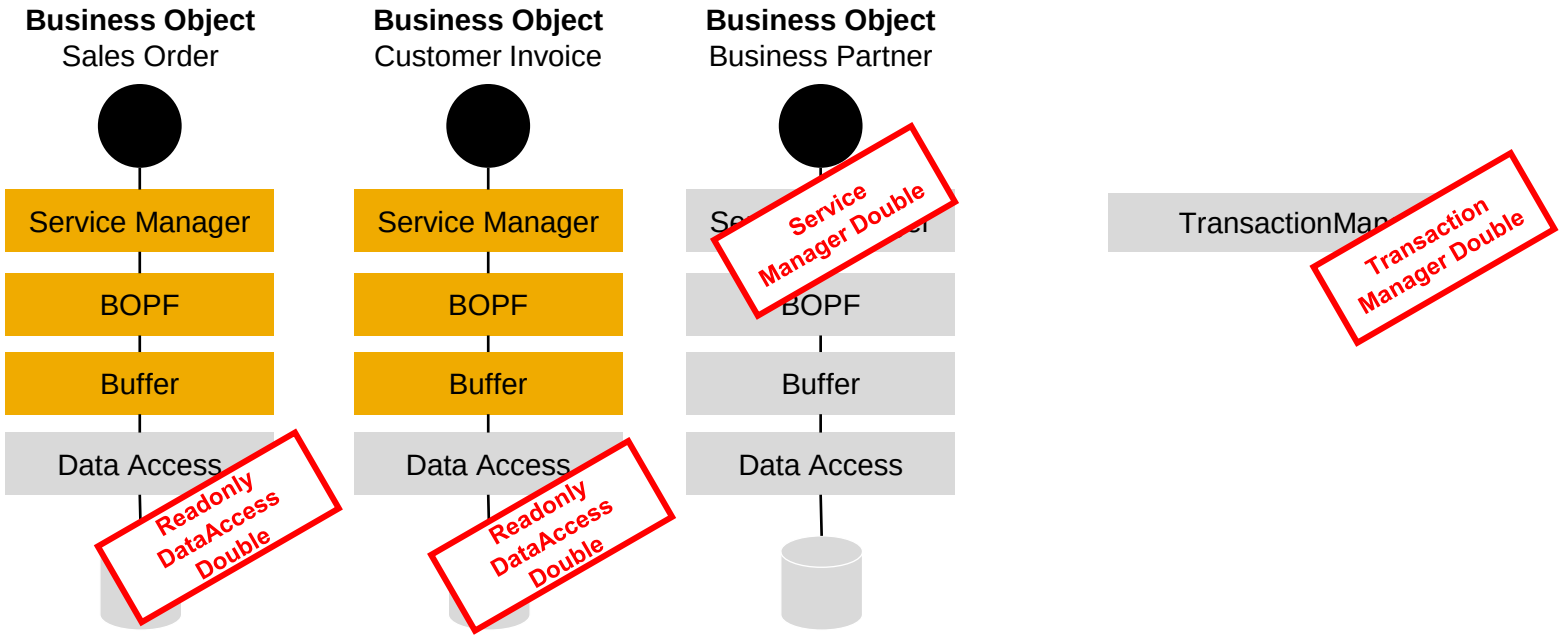


A scenario test describes a process consisting of several steps. In contrast to unit testing, it allows to test integration aspects of different units.

© 2012 SAP AG. All rights reserved.

Scenario Testing

Test Setup



Usually, several BOs participate in a single scenario test. During the test setup phase, you have to decide which functionality shall be mocked:

- The BOPF data access layer must be completely mocked to ensure, that no data is written to the databases during a scenario test.
- The Transaction Manager must be mocked to disable plugins (e.g. PPF)
- If the business logic of a BO is not in scope of the test, it can also be mocked completely.

© 2012 SAP AG. All rights reserved.

Scenario Testing
Example: Test Setup

The productive action code that shall be tested.

```
METHOD /BOBF/IF_FRW_ACTION~EXECUTE.
  " internal retrieve, implicitely DAC is run through...
  io_read->retrieve( ... ).
  " retrieve at other BO via SM...
  lo_service_manager_bo2->retrieve( ... ).
```

Declare and implement the Data Access Class (DAC) and the service manager double. Redefine `~retrieve`/`~read`.

```
CLASS lcl_service_manager_double DEFINITION FOR TESTING
 INHERITING FROM /bobf/cl_tool_test_double_sm.
  PUBLIC SECTION. METHODS /bobf/if_tra_service_manager~retrieve REDEFINITION.
…
class /BOBF/CL_TST_DAC_DOUBLE definition
 inheriting from /BOBF/CL_DAC_TABLE
  public section. methods /BOBF/IF_BUF_DATA_ACCESS~READ redefinition .
```

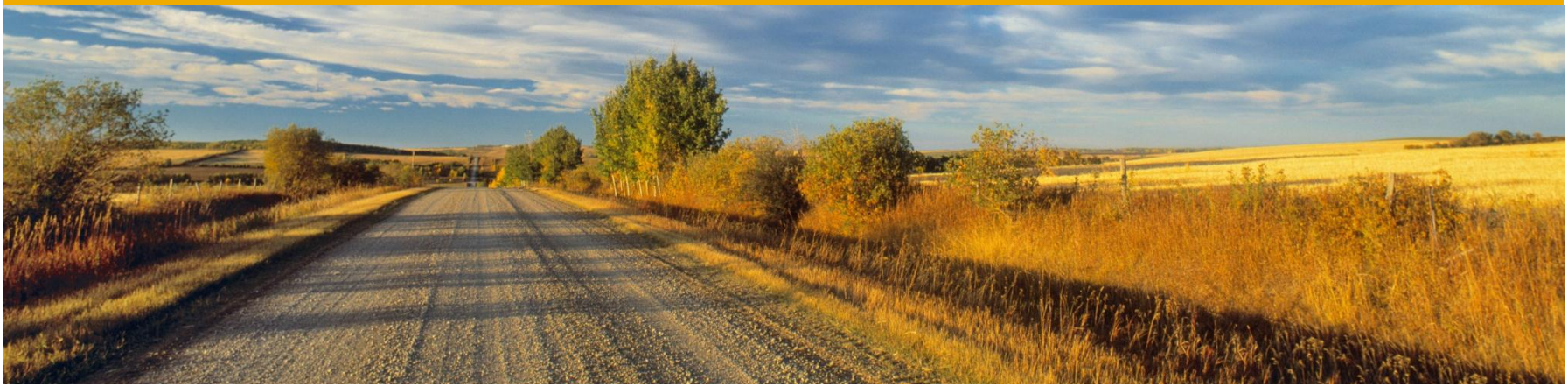
Instantiate and inject the DAC and the service manager double in test setup phase.

```
METHOD class_setup.
  lo_test_manager->inject_service_manager_double(
    iv_bo_key    = lv_bo2_key
    io_sm_double = lo_service_manager_double ).
  lo_test_manager->modify_dac_configuration(
    iv_bo_key   = if_xxx_c=>sc_bo_key
    iv_node_key = if_xxx_c=>sc_node-root
    iv_dac_name = '/BOBF/CL_TST_DAC_DOUBLE' ).
```

Test the action that now internally uses the DAC and the service manager double.

```
METHOD test_action.
  " call/test the action which internally uses the DAC and the SM double...
  lo_action->/bobf/if_frw_action~execute( … ).
```

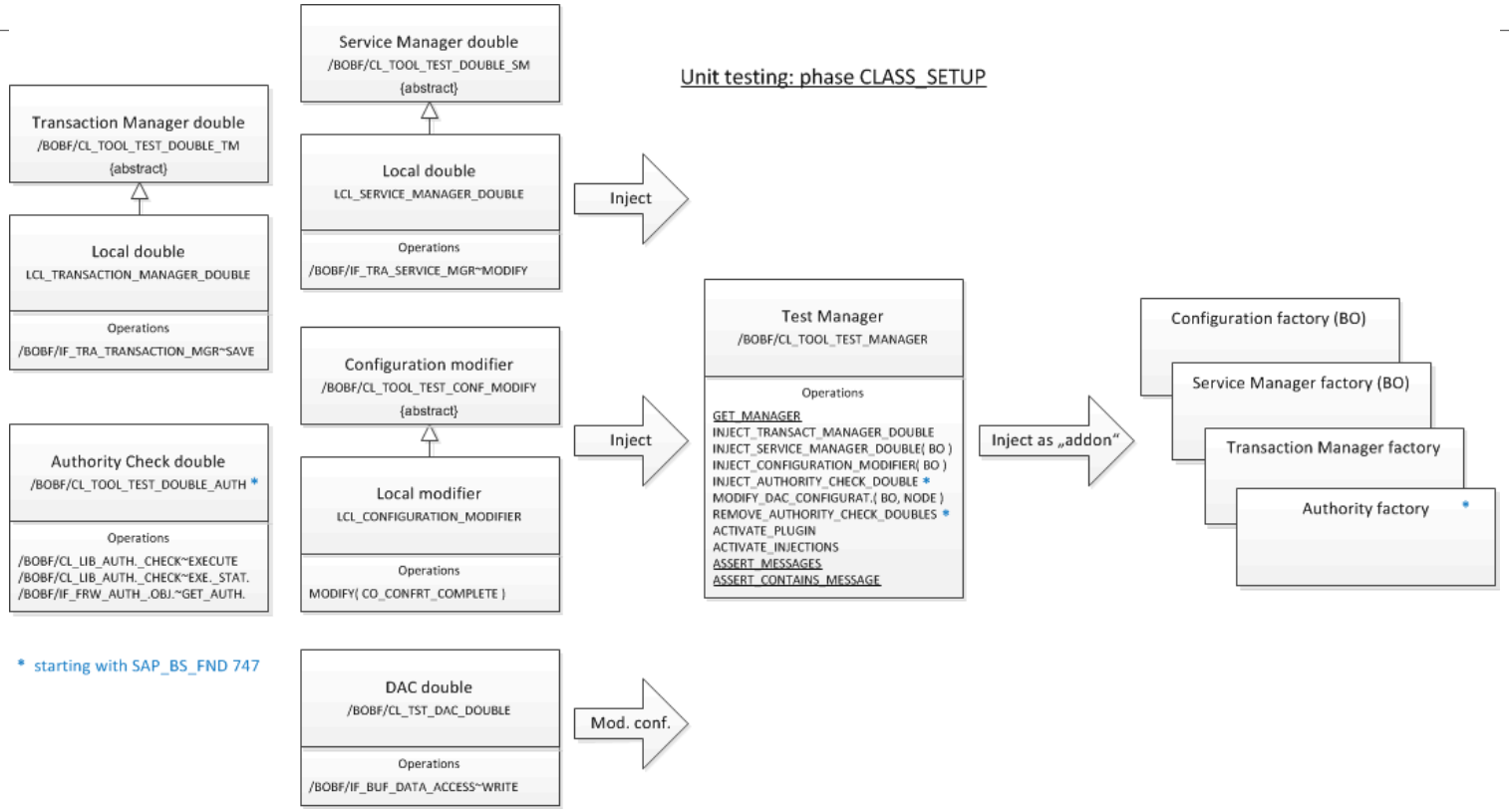
© 2012 SAP AG. All rights reserved.



Unit Test Implementation

Unit Testing

BOPF Test Framework



For easy unit testing, **BOPF supports injection of class doubles in all layers**. You can inject doubles for Transaction and Service Manager. You can moreover inject a DAC or an Authority Check (>= SAP_BS_FND 747) double. If this is not sufficient, you can also modify any class name in the configuration, e.g. the buffer class name.

© 2012 SAP AG. All rights reserved.

Unit Testing

BOPF Test Framework - Continued

The following code snippets may bring some light in the usage of the BOPF test double injection capabilities.

It is important to know that the injection itself has to be done at the very beginning of a test session. During a session, the injection cannot be changed. **Therefore, the unit test method `CLASS_SETUP` is a good place for injecting.**

By activating the test double injection, **all plugins are deactivated automatically**. If you do not want this, you can re-activated a plugin with the method `activate_plugin()`.

© 2012 SAP AG. All rights reserved.

Unit Testing

BOPF Test Framework - Continued

First of all the (local) double classes that shall be injected, have to be defined. For example a local double for Service Manager that overwrites the method `RETRIEVE`.

```
CLASS lcl_service_manager_double DEFINITION FOR TESTING
 INHERITING FROM /bobf/cl_tool_test_double_sm.
  PUBLIC SECTION.
    METHODS /bobf/if_tra_service_manager~retrieve REDEFINITION.
ENDCLASS.

CLASS lcl_service_manager_double IMPLEMENTATION.
  METHOD /bobf/if_tra_service_manager~retrieve.
    LOOP AT it_key INTO DATA(ls_key).
      " do the needful ;-)
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.
```

© 2012 SAP AG. All rights reserved.

Unit Testing

BOPF Test Framework - Continued

As next step, the double classes have to be instantiated and injected.

As already mentioned, the unit test method CLASS_SETUP is a good place for injecting.

Activating the injections results in the “standard” instances (e.g. for Service Manager) being replaced by the doubles.

```
METHOD class_setup.

  " instantiate service manager double...
  DATA lo_service_manager_double TYPE REF TO lcl_service_manager_double.
  CREATE OBJECT lo_service_manager_double.

  " >>>>> inject and activate the doubles...

  DATA lo_test_manager TYPE REF TO /bobf/cl_tool_test_manager.
  lo_test_manager = /bobf/cl_tool_test_manager=>get_manager( ).

  lo_test_manager->inject_service_manager_double(
    iv_bo_key    = /bobf/if_tst_test_double_c=>sc_bo_key
    io_sm_double = lo_service_manager_double ).

  lo_test_manager->activate_injections( ).

ENDMETHOD.
```

© 2012 SAP AG. All rights reserved.

Unit Testing

BOPF Test Framework - Continued

While running a unit test method, the injected code is run through.

```
METHOD test_sm_double.

  " get service manager...
  DATA lo_service_manager TYPE REF TO /bobf/if_tra_service_manager.
  lo_service_manager = /bobf/cl_tra_serv_mgr_factory=>get_service_manager(
   /bobf/if_tst_test_double_c=>sc_bo_key ).

  " call retrieve, retrieve is implemented in service manager double...
  DATA lt_root_data TYPE /bobf/t_tstdouble_rt.
  lo_service_manager->retrieve(
    EXPORTING
      iv_node_key  = /bobf/if_tst_test_double_c=>sc_node-root
      it_key       = VALUE #( ( key = gc_root_key_1 ) )
    IMPORTING
      et_data      = lt_root_data ).

  …

ENDMETHOD.
```

© 2012 SAP AG. All rights reserved.

Unit Testing

Scenario Testing - Example

The following code snippets demonstrate the capabilities of the BOPF test double injection for testing a more complex scenario.

In this example the test method shall **test an action that modifies another BO (using Service Manager) and saves data (using Transaction Manager).**

This shall be done in an easy way without saving data to the database.

Therefore, the Service Manager and the Transaction Manager have to be doubled.

© 2012 SAP AG. All rights reserved.

Unit Testing

Scenario Testing – Example - Continued

The code of the action that shall be tested.

```
METHOD /BOBF/IF_FRW_ACTION~EXECUTE.

  " >>>>> call a modify via SM to second BO /BOBF/TST_TEST_DOUBLE_2
  " >>>>> and call afterwards a save via TM...

  " get transaction manager...
  DATA lo_tm TYPE REF TO /bobf/if_tra_transaction_mgr.
  lo_tm = /bobf/cl_tra_trans_mgr_factory=>get_transaction_manager( ).

  " get service manager for /BOBF/TST_TEST_DOUBLE_2...
  DATA lo_sm_bo2 TYPE REF TO /bobf/if_tra_service_manager.
  lo_sm_bo2 = /bobf/cl_tra_serv_mgr_factory=>get_service_manager(
    /bobf/if_tst_test_double_2_c=>sc_bo_key ).

  " call modify for /BOBF/TST_TEST_DOUBLE_2...
  DATA lt_modification TYPE /bobf/t_frw_modification.
  lo_sm_bo2->modify( it_modification = lt_modification ).

  " call save...
  lo_tm->save( ).

ENDMETHOD.
```

© 2012 SAP AG. All rights reserved.

Unit Testing

Scenario Testing – Example - Continued

First of all the (local) double classes that shall be injected, have to be defined.

```
CLASS lcl_service_manager_double DEFINITION FOR TESTING
 INHERITING FROM /bobf/cl_tool_test_double_sm.
  PUBLIC SECTION.
    METHODS /bobf/if_tra_service_manager~modify REDEFINITION.
ENDCLASS.

CLASS lcl_service_manager_double IMPLEMENTATION.
  METHOD /bobf/if_tra_service_manager~modify.
    " do nothing...
  ENDMETHOD.
ENDCLASS.

CLASS lcl_transaction_manager_double DEFINITION FOR TESTING
 INHERITING FROM /bobf/cl_tool_test_double_tm.
  PUBLIC SECTION.
    METHODS /bobf/if_tra_transaction_mgr~save REDEFINITION.
ENDCLASS.

CLASS lcl_transaction_manager_double IMPLEMENTATION.
  METHOD /bobf/if_tra_transaction_mgr~save.
    " do nothing...
  ENDMETHOD.
ENDCLASS.
```

© 2012 SAP AG. All rights reserved.

Unit Testing

Scenario Testing – Example - Continued

As next step the double classes have to be instantiated and injected.

As already mentioned, the unit test method CLASS_SETUP is a good place for injecting.

```
METHOD class_setup.

  " instantiate transaction manager double...
  DATA lo_transaction_manager_double TYPE REF TO lcl_transaction_manager_double.
  CREATE OBJECT lo_transaction_manager_double.

  " instantiate service manager double...
  DATA lo_service_manager_double TYPE REF TO lcl_service_manager_double.
  CREATE OBJECT lo_service_manager_double.

  " >>>>> inject and activate the doubles...

  DATA lo_test_manager TYPE REF TO /bobf/cl_tool_test_manager.
  lo_test_manager = /bobf/cl_tool_test_manager=>get_manager( ).

  lo_test_manager->inject_transact_manager_double( lo_transaction_manager_double ).

  lo_test_manager->inject_service_manager_double(
    iv_bo_key    = /bobf/if_tst_test_double_2_c=>sc_bo_key " second BO !!!!!
    io_sm_double = lo_service_manager_double ).

  lo_test_manager->activate_injections( ). …
```

© 2012 SAP AG. All rights reserved.

Unit Testing

Scenario Testing – Example - Continued

While running the unit test method, that shall test the action, the injected code is run through.

```
METHOD TEST_ACTION.
  ...

  " call the action, which internally calls a second BO with the service manager double
  " and which internally calls a save with the transaction manager double.
  " service manager double for second BO was already instantiated and injected in CLASS_SETUP
  " transaction manager double was already instantiated and injected in CLASS_SETUP.
  TRY.
      DATA lo_action TYPE REF TO /bobf/cl_tst_a_double_sm_tm.
      CREATE OBJECT lo_action.
      lo_action->/bobf/if_frw_action~execute(
        EXPORTING
          is_ctx        = ls_ctx
          it_key        = lt_key
          io_read       = lo_read
          io_modify     = lo_modify
          is_parameters = ls_parameters
      ).
    CATCH /bobf/cx_frw.
  ENDTRY.

  ...
ENDMETHOD.
```

© 2012 SAP AG. All rights reserved.

Unit Testing

Scenario Testing – Example for Authority Check double injection (>= 747)

The following code snippets demonstrate the capabilities of the **Authority Check double** injection. This functionality is available starting with release SAP_BS_FND 747.

Remark: in contrast to an injection of a Transaction or Service Manager double – an Authority Check double can be injected, removed and re-injected at any time/place of the test coding. Therefore it's no need, to inject the double in the unit test method `CLASS_SETUP`.

© 2012 SAP AG. All rights reserved.

Unit Testing

Scenario Testing – Example for Authority Check double injection (>= 747) – Cont.

The code (`RETRIEVE` on `ROOT`) shall be tested with a specific authority profile.

```
METHOD test_something.
  ...
  " >>>>> inject an Authority double and call the RETRIEVE afterwards...

  data(lo_test_manager) = /bobf/cl_tool_test_manager=>get_manager( ).

  lo_test_manager->inject_authority_check_double(
    NEW /bobf/cl_tool_test_double_auth(
      iv_bo_key   = /bobf/if_xxx_c=>sc_bo_key
      iv_node_key = /bobf/if_xxx_c=>sc_node-root
      it_profile  = VALUE #(
        ( authority_object = 'BOBF_TST1'
          authority_field  = 'ACTVT'
          authorities      = VALUE #( ( sign =  'I' option = 'EQ'
             low = /bobf/cl_frw_authority_check=>sc_activity-display ) ) )
        ( authority_object = 'BOBF_TST1'
          authority_field  = 'COUNTRY'
          authorities      = VALUE #( ( sign =  'I' option = 'EQ' low = 'EN' ) ) ) ) ) ).

  lo_service_manager->retrieve( iv_node_key = /bobf/if_xxx_c=>sc_node-root … ).

  " >>>>> remove the Authority double...

  lo_test_manager->remove_authority_check_doubles( ).
```

© 2012 SAP AG. All rights reserved.

Thank you

© 2012 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, PowerPoint, Silverlight, and Visual Studio are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, z10, z/VM, z/OS, OS/390, zEnterprise, PowerVM, Power Architecture, Power Systems, POWER7, POWER6+, POWER6, POWER, PowerHA, pureScale, PowerPC, BladeCenter, System Storage, Storwize, XIV, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, AIX, Intelligent Miner, WebSphere, Tivoli, Informix, and Smarter Planet are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the United States and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are trademarks or registered trademarks of Adobe Systems Incorporated in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and its affiliates.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems Inc.

HTML, XML, XHTML, and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Apple, App Store, iBooks, iPad, iPhone, iPhoto, iPod, iTunes, Multi-Touch, Objective-C, Retina, Safari, Siri, and Xcode are trademarks or registered trademarks of Apple Inc.

IOS is a registered trademark of Cisco Systems Inc.

RIM, BlackBerry, BBM, BlackBerry Curve, BlackBerry Bold, BlackBerry Pearl, BlackBerry Torch, BlackBerry Storm, BlackBerry Storm2, BlackBerry PlayBook, and BlackBerry App World are trademarks or registered trademarks of Research in Motion Limited.

Google App Engine, Google Apps, Google Checkout, Google Data API, Google Maps, Google Mobile Ads, Google Mobile Updater, Google Mobile, Google Store, Google Sync, Google Updater, Google Voice, Google Mail, Gmail, YouTube, Dalvik and Android are trademarks or registered trademarks of Google Inc.

INTERMEC is a registered trademark of Intermec Technologies Corporation.

Wi-Fi is a registered trademark of Wi-Fi Alliance.

Bluetooth is a registered trademark of Bluetooth SIG Inc.

Motorola is a registered trademark of Motorola Trademark Holdings LLC.

Computop is a registered trademark of Computop Wirtschaftsinformatik GmbH.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase Inc. Sybase is an SAP company.

Crossgate, m@gic EDDY, B2B 360°, and B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG.