The Enhancement Framework

for Nordic User group

Contributing Speakers

Michael Acker
Development Architect, NW AS ABAP Workbench, SAP AG

Dr. Roland Krämer
Senior Developer, NW AS ABAP Workbench, SAP AG

Learning Objectives

As a result of this workshop, you will be able to

- **Understand the fundamental idea of the Enhancement Framework and Switch Framework**
- **Reduce TCO by using enhancement technologies instead of modifications**
- **Enhance SAP standard objects**
- **Create your own Enhancement Spots for your own source code**
- **Work with enhancements during an upgrade or import of a support package**

- Enhancement Framework Overview
- Source Code Plugin – Technology
- Function Group Enhancement – Technology
- Class Enhancement – Technology
- BAdI – Technology
- Upgrade Adjustment
- Switch Framework
- Summary

Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

BAdI – Technology

Upgrade Adjustment

Switch Framework

Summary

Adapting SAP Software

One of the advantages of SAP software is the possibility to adapt the software to own requirements and the possibility of keeping the adaptations during upgrade.

Ways of adaptation:

- **Customizing**
- **Enhancement**
- **Modification**

Motivation

Reducing TCO

- **Enhancing objects instead of modifying them reduces the effort for adjustment during SP import or upgrade.**

Disadvantages of modifications

- **No support for multiple users or projects**
- **No grouping**
- **No support for parallel developments**
- **Will appear much more often in adjustment tools**
- **Higher adjustment effort**

Motivation: Goals of Enhancement Framework

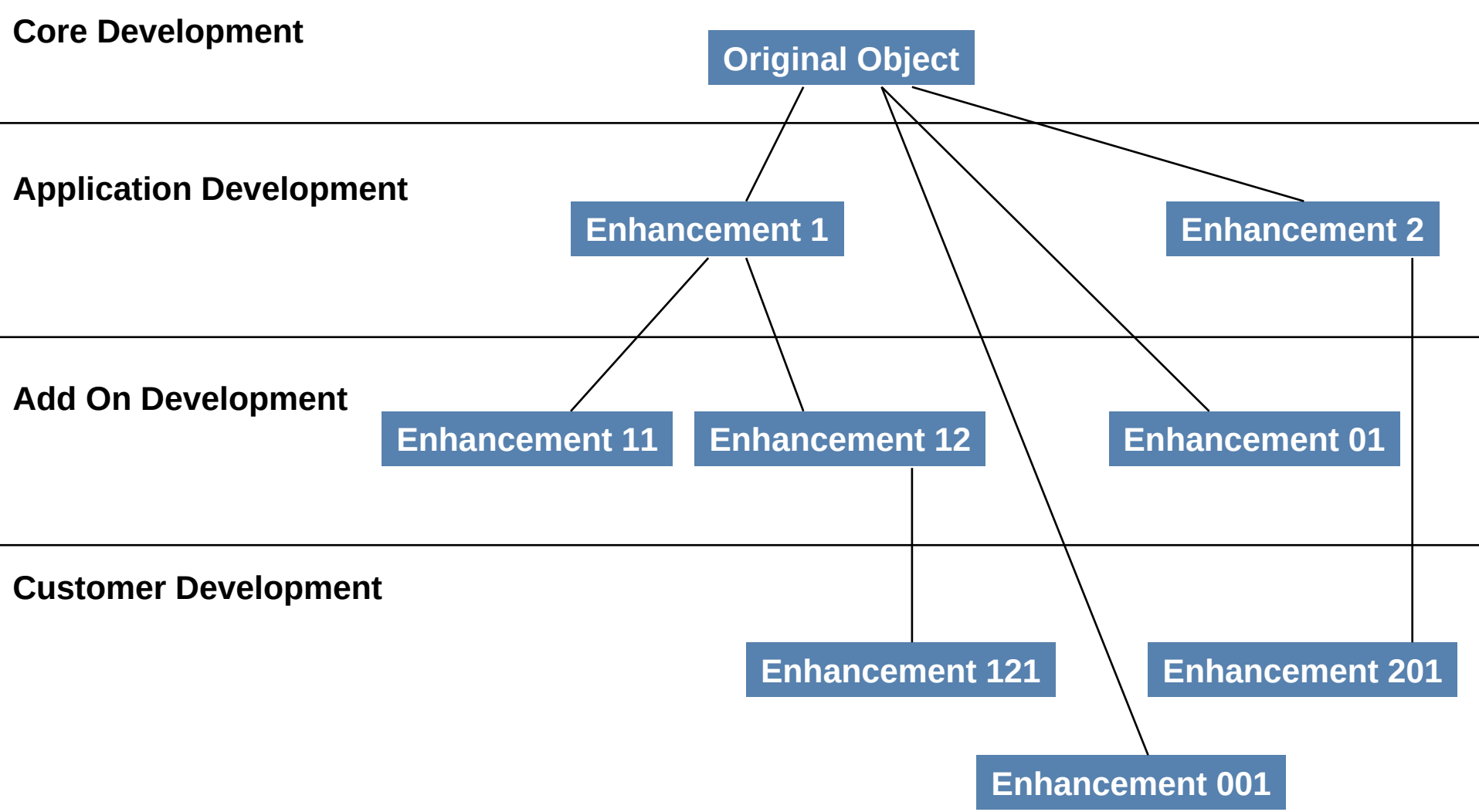
Integration of several enhancement types

- **BAdIs**
- **Functiongroup Enhancement**
- **Class/Interface Enhancement**
- **Source Code PlugIns**
- **WebDynpro Enhancement**
- **...**

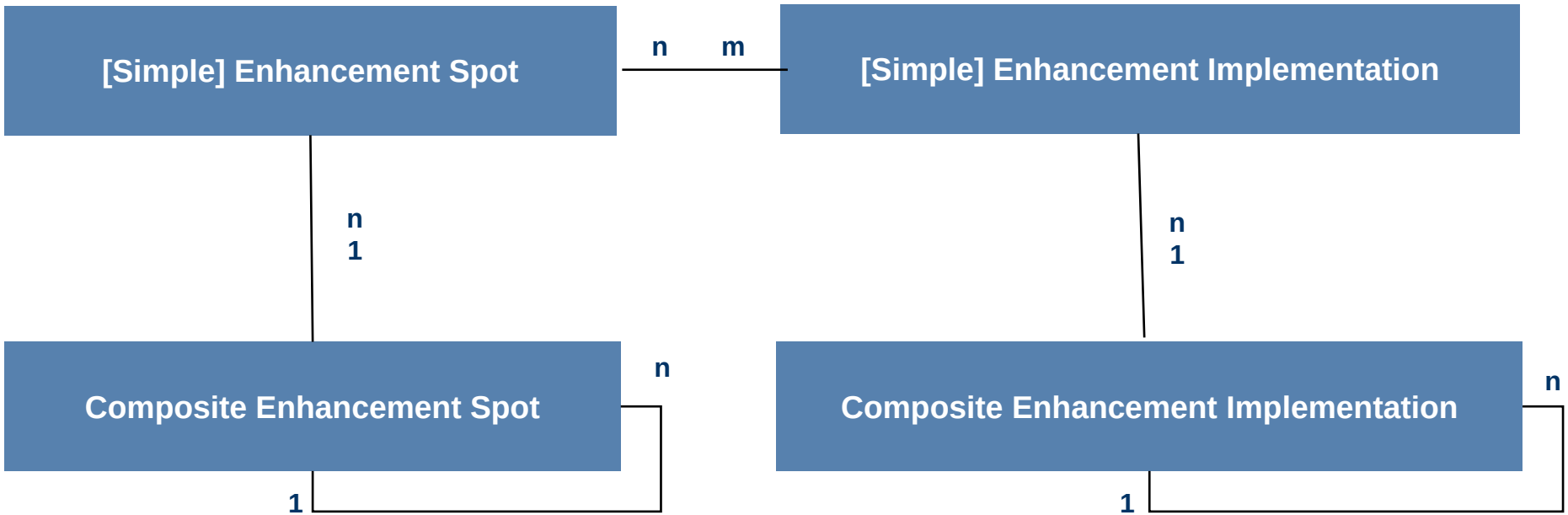
into the Enhancement Framework

- **Switchable by Switch Framework**
- **Enhancement Browser**
- **Upgrade support**
- **Possibility to document and group enhancements**
- **Multilayer support**

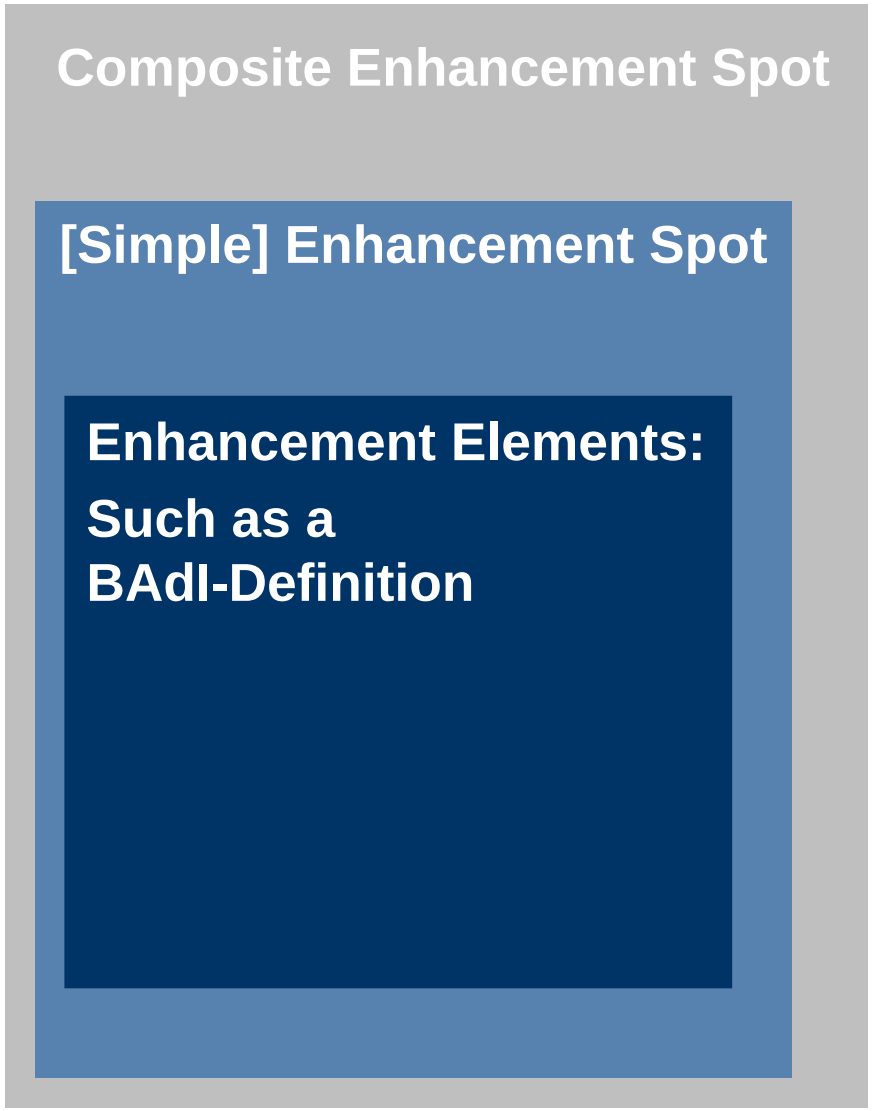
Multilayer Support



Enhancements – Relations



The Big Picture – Another Perspective (1)



Composite Enhancement Spots

- **Container Objects**
- **Can contain**
  - **Other Composite Enhancement Spots**
  - **[Simple] Enhancement Spots**

[Simple] Enhancement Spots

- **Container Objects**
- **Can contain Enhancement Elements**

Explicit Enhancement Options

- **Enhancement Definitions**

The Big Picture – Another Perspective (2)

Composite Enhancement Implementation

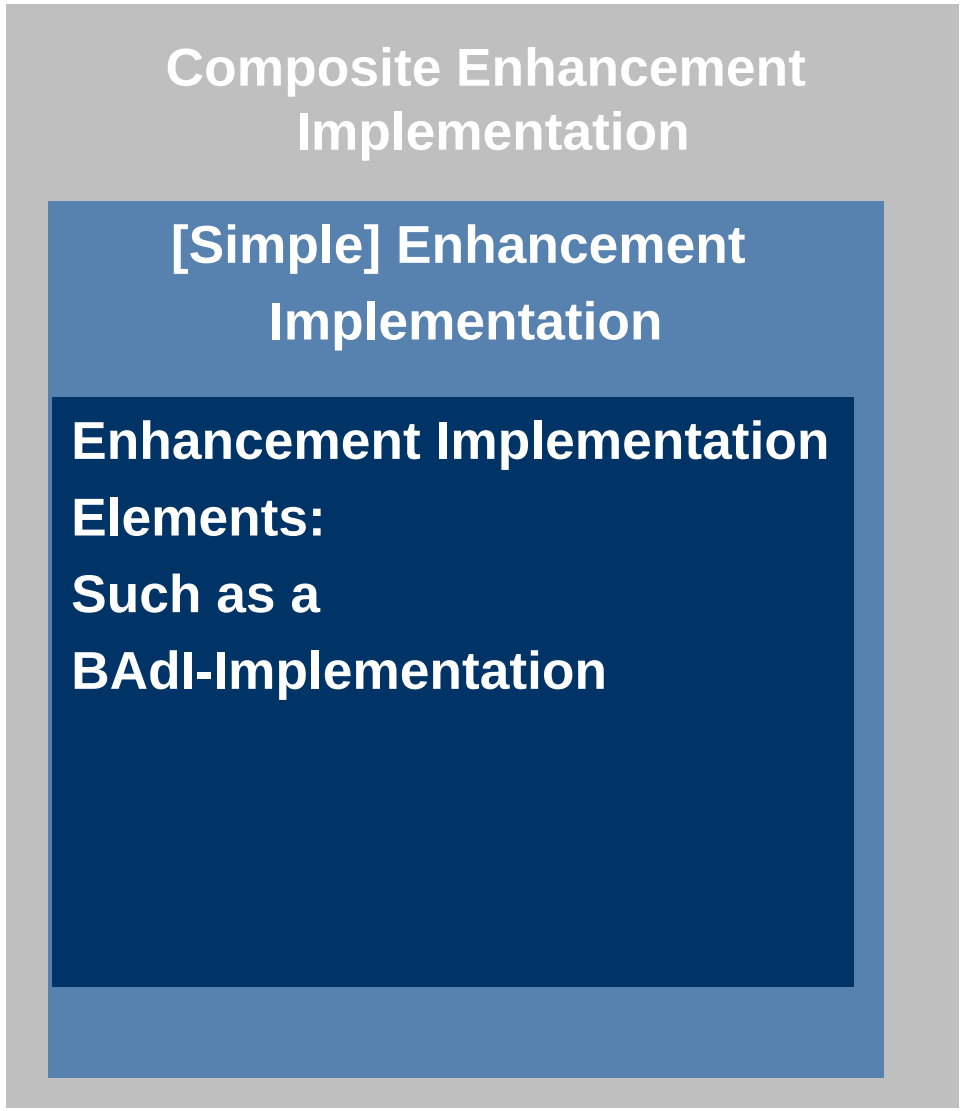
- **Container Objects**
- **Can contain**
  - **Other Composite Enhancement Implementations**
  - **[Simple] Enhancement Implementations**

[Simple] Enhancement Implementation

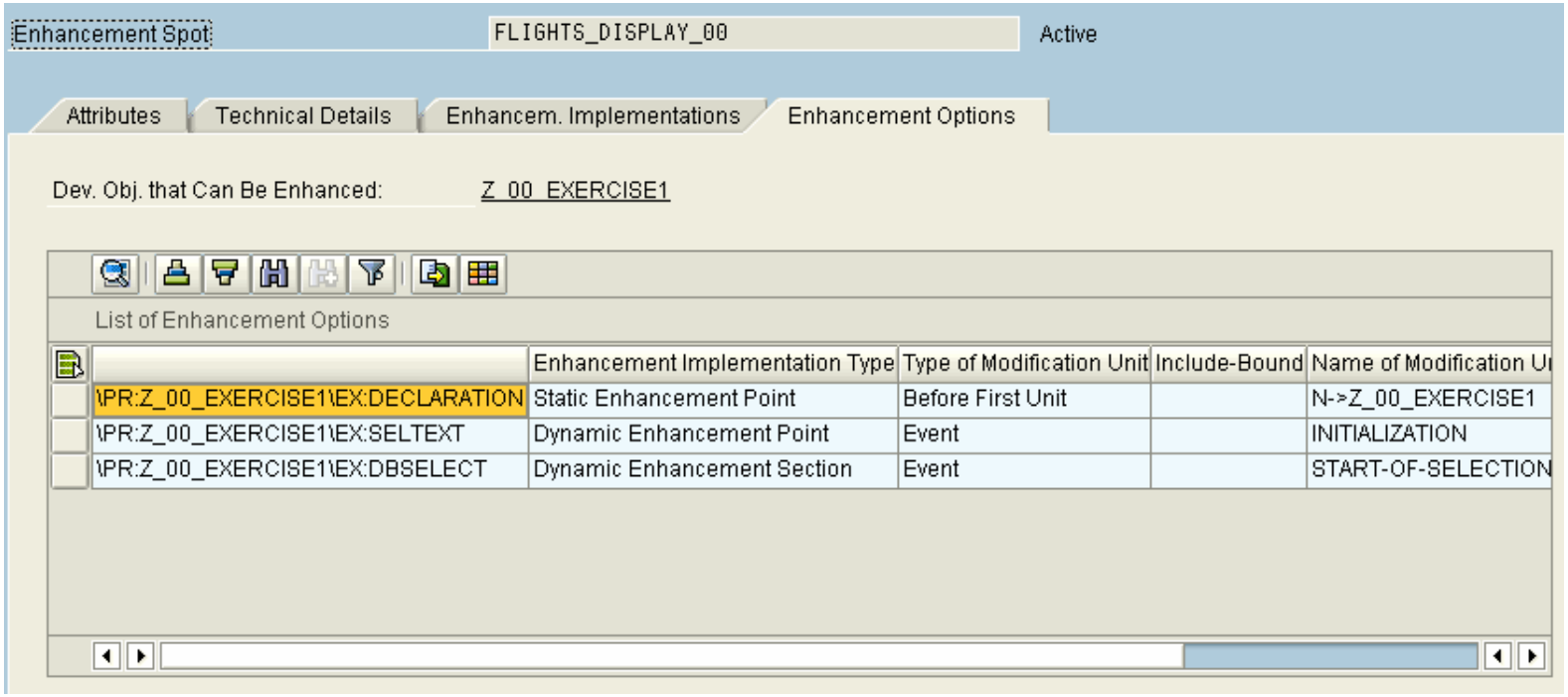
- **Can contain Enhancement Implementation Elements**

Enhancement Implementation Elements

- **Implementations**

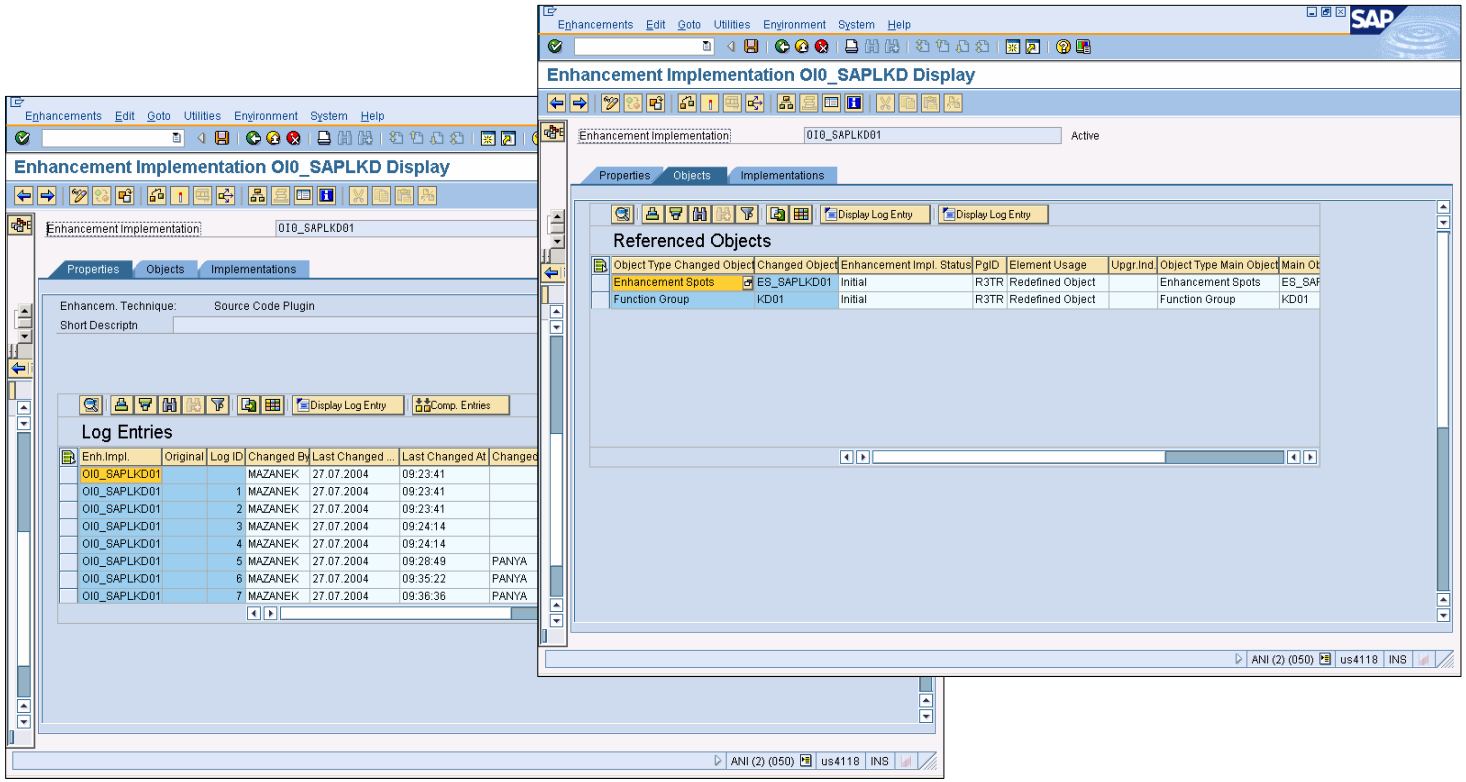


Enhancement Spot Editor



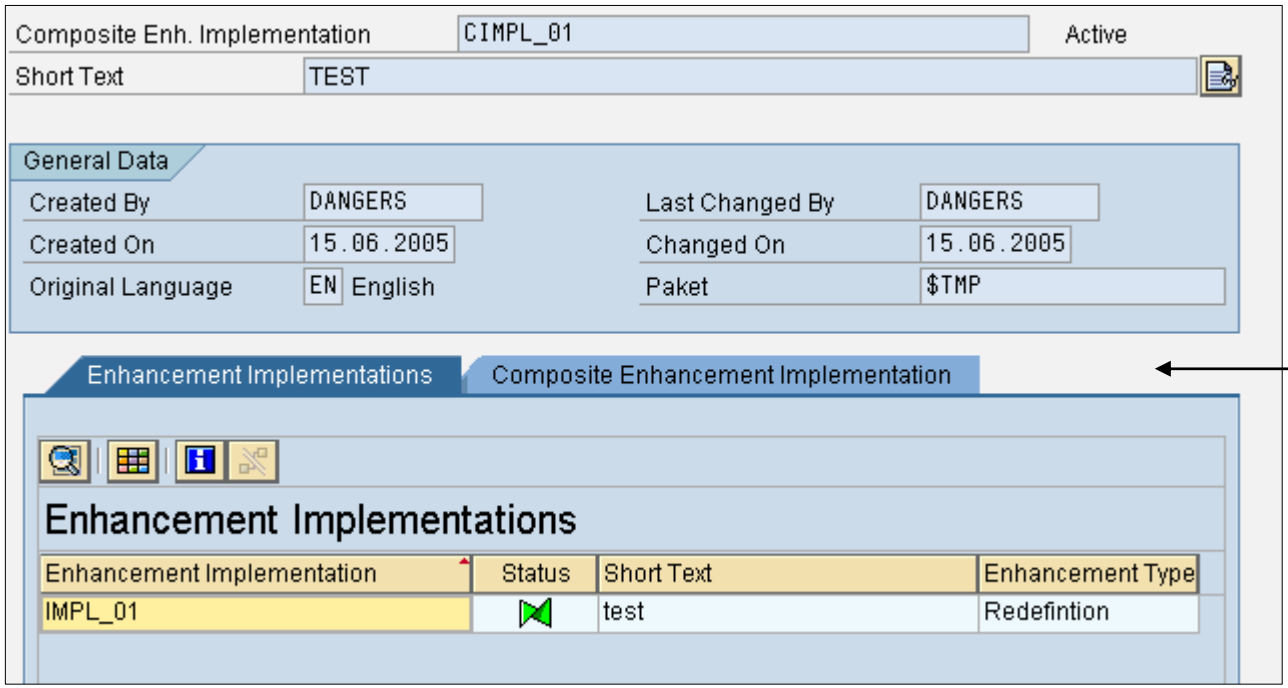
- **Editor for Predefined Enhancement Possibilities (Source Code Enhancements & BAdIs)**
- **Integrated in Object Navigator (SE80)**
- **common tabs for all Enhancement Spots**
- **Tab 3 ➔ dependent on enhancement technology: BAdIs or Source Code Enhancements**

Enhancement Implementation Editor



- **Editor for Enhancement Implementations**
- **Integrated in Object Navigator (SE80)**
- **Tab Properties & Objects ➔ common for all enhancement types**
- **Tab 3 ➔ dependent on enhancement technology:**
  e.g. BAdI-Implementation or Source Code Enhancements

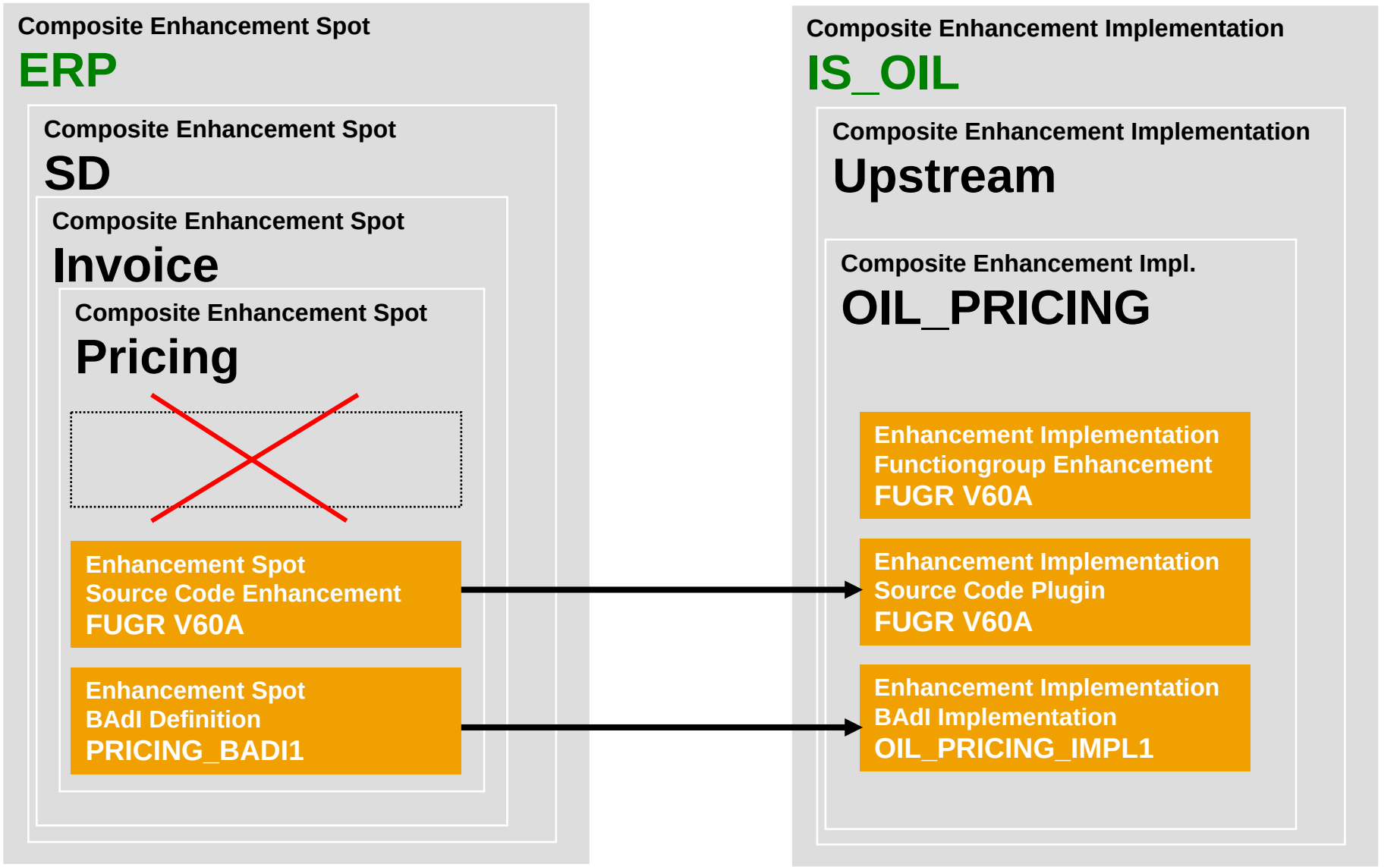
Composite Enhancement Implementation



Contained Composite Enhancement Implementations

contains [Simple] Enhancement Implementations

Terminology by Example



Differences: Implicit and Explicit Enhancement Options

Features of explicit enhancement options

- **More stable, pre-defined**
- **Few changes in definition to expect**
- **Only at valid source code locations**

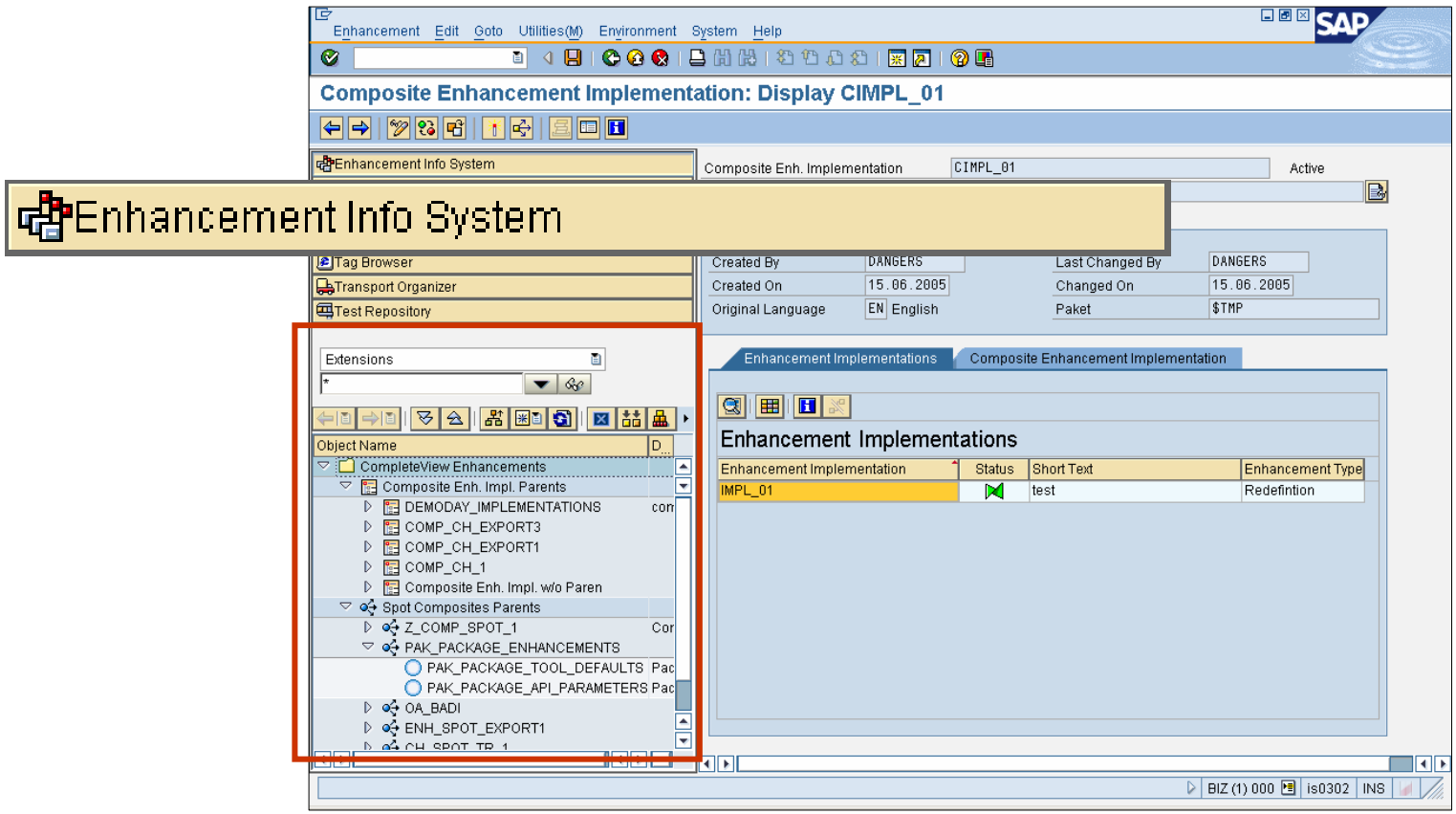
Features of implicit enhancement options

- **enhancement of „arbitrary“ objects**
- **No enhancement spots necessary**

Enhancement Browser

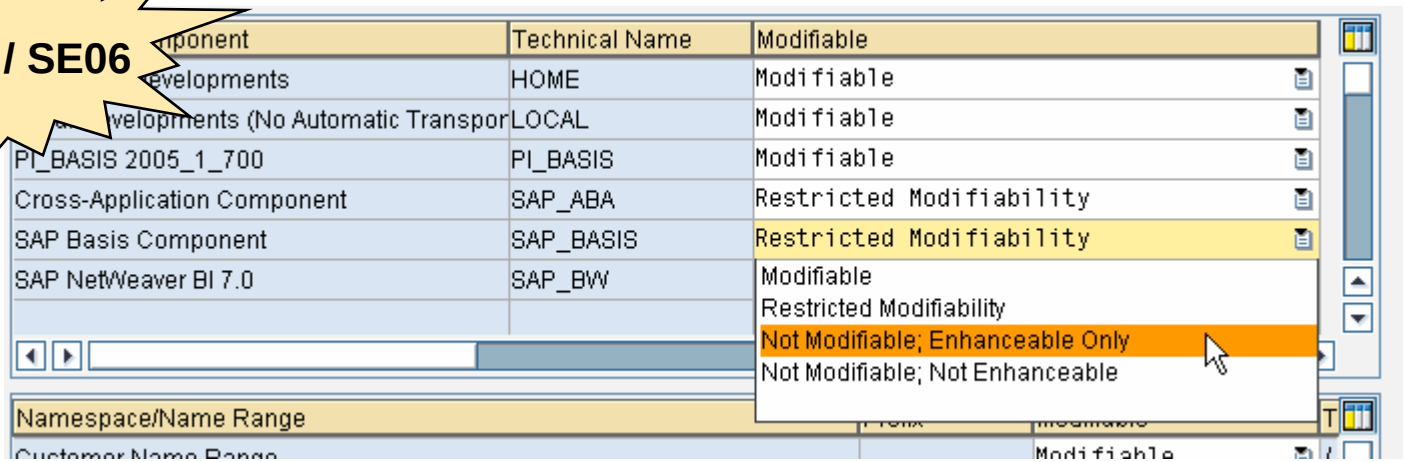
Search for

- **Enhancements possibilities**
- **Existing Enhancement Implementations**
- **Enhancement Implementations to be adjusted after upgrade**



System Change Option – Enhanceable

SE03 / SE06



New option:

Enhanceable ➔ **Only creation of enhancement implementations is allowed**

Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

BAdI – Technology

Upgrade Adjustment

Switch Framework

Summary

Source Code Plugin Technology

```
PROGRAM p1.

WRITE ‘Hello World’.

ENHANCEMENT-POINT ep1 SPOTS
s1.

..
..
..

ENHANCEMENT-SECTION ep2
SPOTS s1.
  WRITE ’Original’.
END-ENHANCEMENT-SECTION.
```

```
ENHANCEMENT 1.
  WRITE ’Hello
Paris’.
ENDENH
```

```
ENHANCEMENT 2.
  WRITE ’Hello
London’.
ENDENHANCEMENT.
```

```
ENHANCEMENT 3.
  WRITE ’Enhanced’.
ENDENHANCEMENT.
```

Modification-free enhancement of source code

Explicit Enhancement Option

- Predefined enhancement options can be defined in source code. They are additionally stored inside Enhancement Spots.

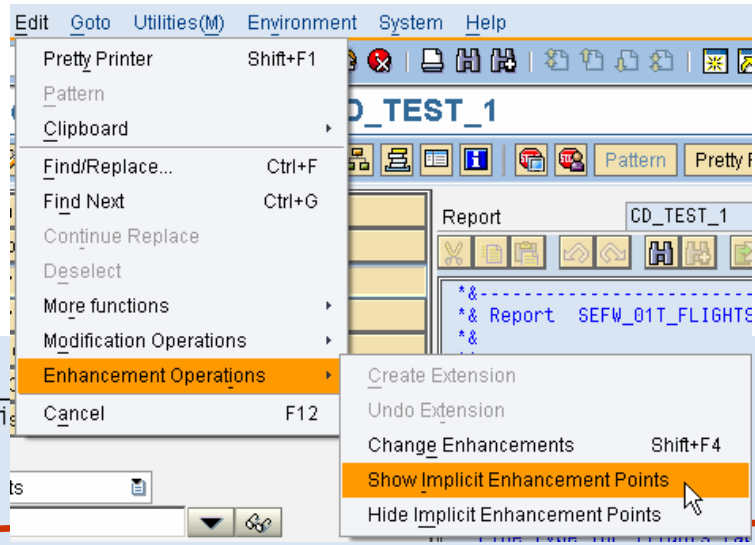
Implicit Enhancement Option

- **At common enhancement places, implicit Enhancement options are available. Examples:**
  - **Beginning/End of Include**
  - **Beginning/End of Method/Function Module/Form Routine**
  - **End of a structure**
  - **End of Private/Protected/Public Section of a local class**
  - ...

Explicit Enhancement Options

```
* Selection screen
SELECT-OPTIONS:
  so_carr FOR gv_carrid,
  so_conn FOR gv_connid.

ENHANCEMENT-POINT FLIGHTS_DECLARATION SPOTS FLIGHTS_DISPLAY STATIC.

START-OF-SELECTION.

* Select Data
ENHANCEMENT-SECTION     FLIGHTS_DBSELECT SPOTS FLIGHTS_DISPLAY.
  SELECT carrid connid fldate price currency
    from sflight
    into table gt_flights
    where carrid in so_carr
      and connid in so_conn.
END-ENHANCEMENT-SECTION.

* Listoutput
    LOOP AT gt_flights INTO gs_flights.
      WRITE: / gs_flights-carrid,
      gs_flights-connid,
```

Implicit Enhancement Options




```
* Listoutput
    LOOP AT gt_flights INTO gs_
      WRITE: / gs_flights-carri
      gs_flights-connid,
      gs_flights-fldate,
      gs_flights-price,
      gs_flights-currency.
    ENDLOOP.
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""$"$\SE:(1 ) Include ZSEFW_00_FLIGHTS_DISPLA
```

```
METHOD GET_PROPERTY.
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""$"$\SE:(4 ) Klasse CL_GUI_OBJECT, Methode G
* (PROPERTY, P1, P16, P_COUNT, QUEUE_ONLY; => VALUE)
* check handle
  DATA : STRPROPNAME TYPE STRING.
  DATA : STRVALUE TYPE STRING.
  DATA : QUEUE_EMPTY TYPE CHAR01.

""
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""$"$\SE:(3 ) Klasse CL_GUI_CONTROL, PUBLIC S
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""$"$\SE:(2 ) Klasse CL_GUI_CONTROL, PROTECTE
  *"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""$"$\SE:(1 ) Klasse CL_GUI_CONTROL, PRIVATE
```

Enhancement Types in Source Code

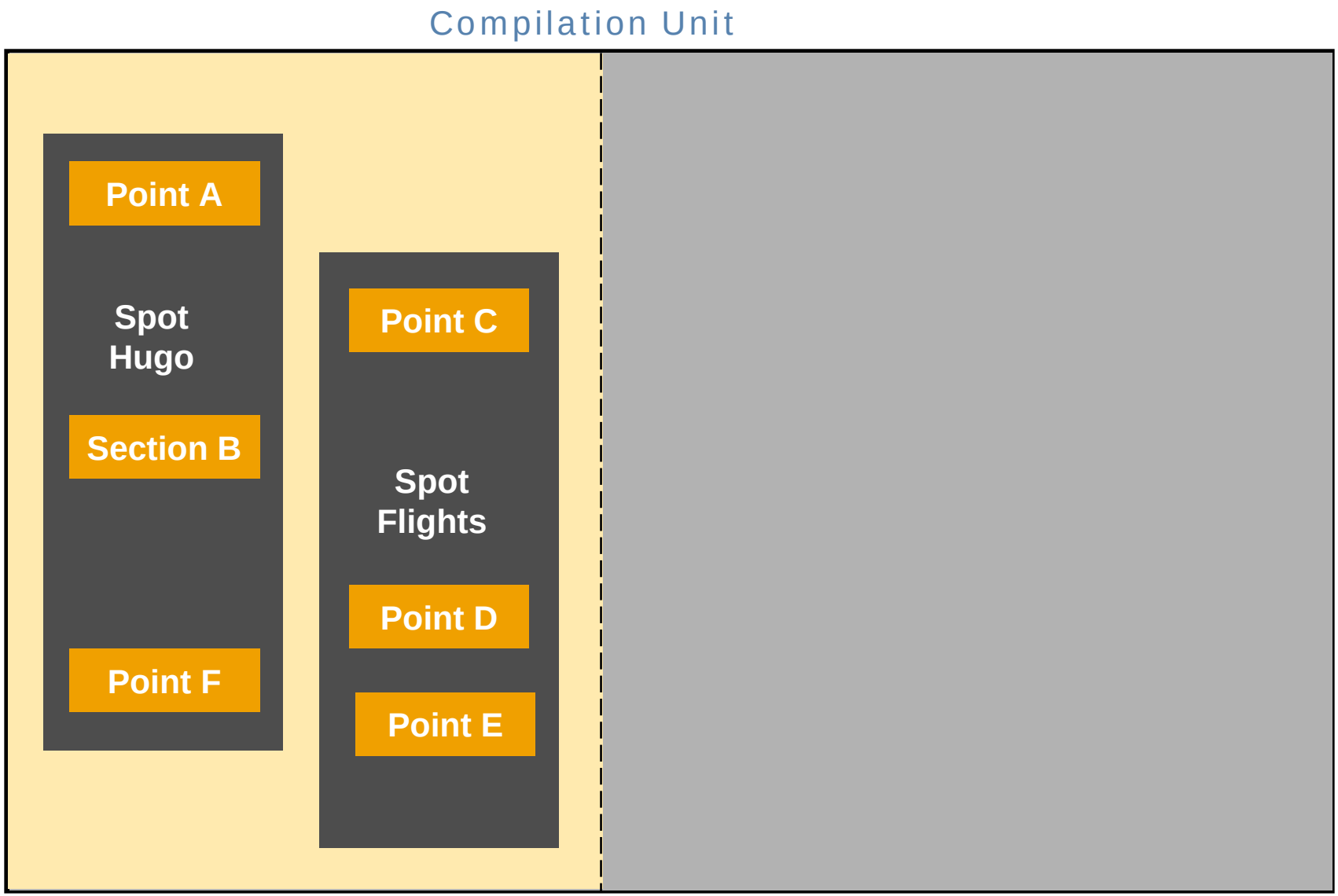
ENHANCEMENT-POINT <name> SPOTS <spot1> [<spot2>] .. [STATIC]

- **Static – e.g. additional data declaration**
- **Dynamic – e.g. additional source code**

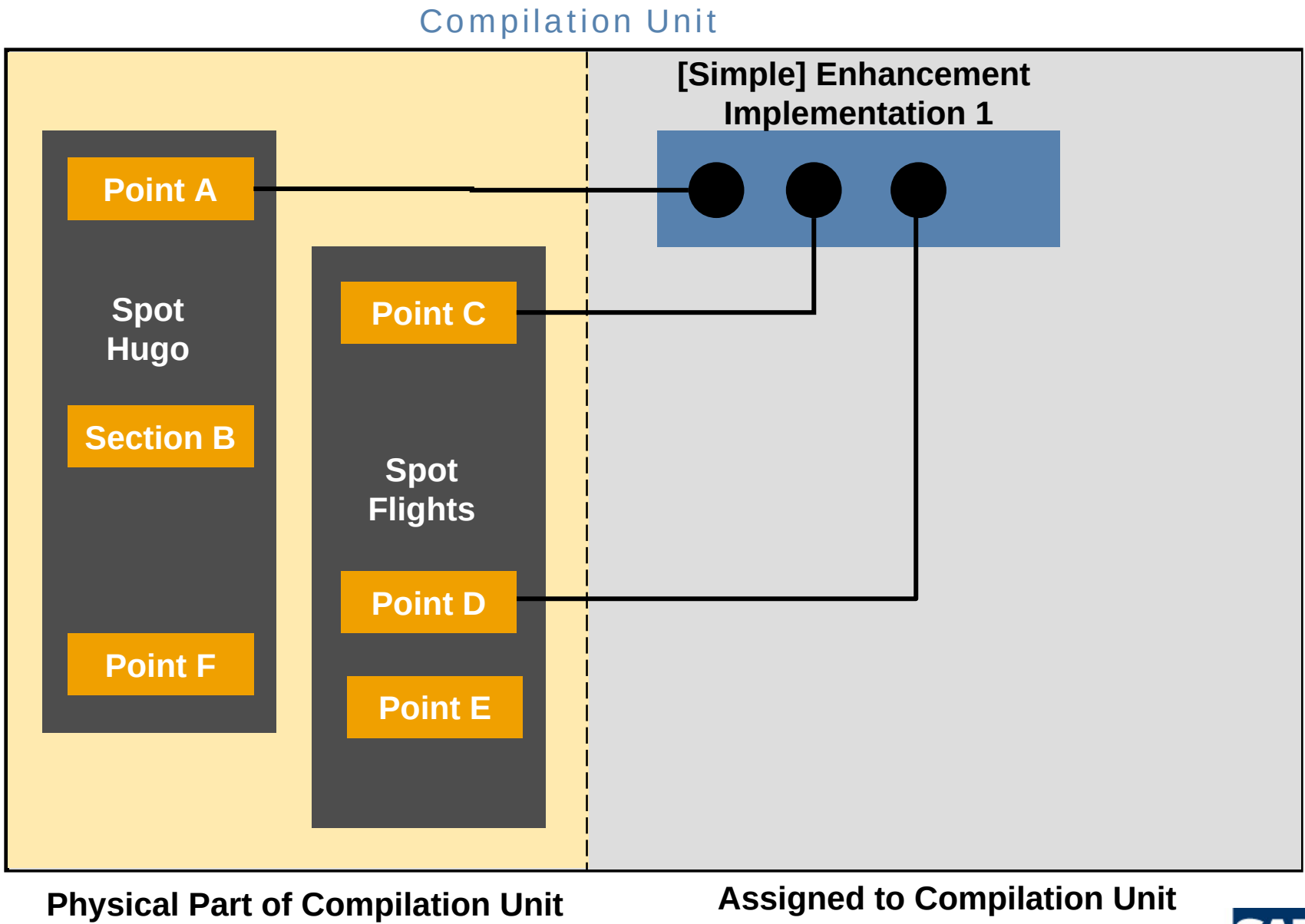
ENHANCEMENT-SECTION <name> SPOTS <spot1> [<spot2>] .. [STATIC]

- **Static – e.g. replace an existing data declaration**
- **Dynamic – e.g. replace source code**

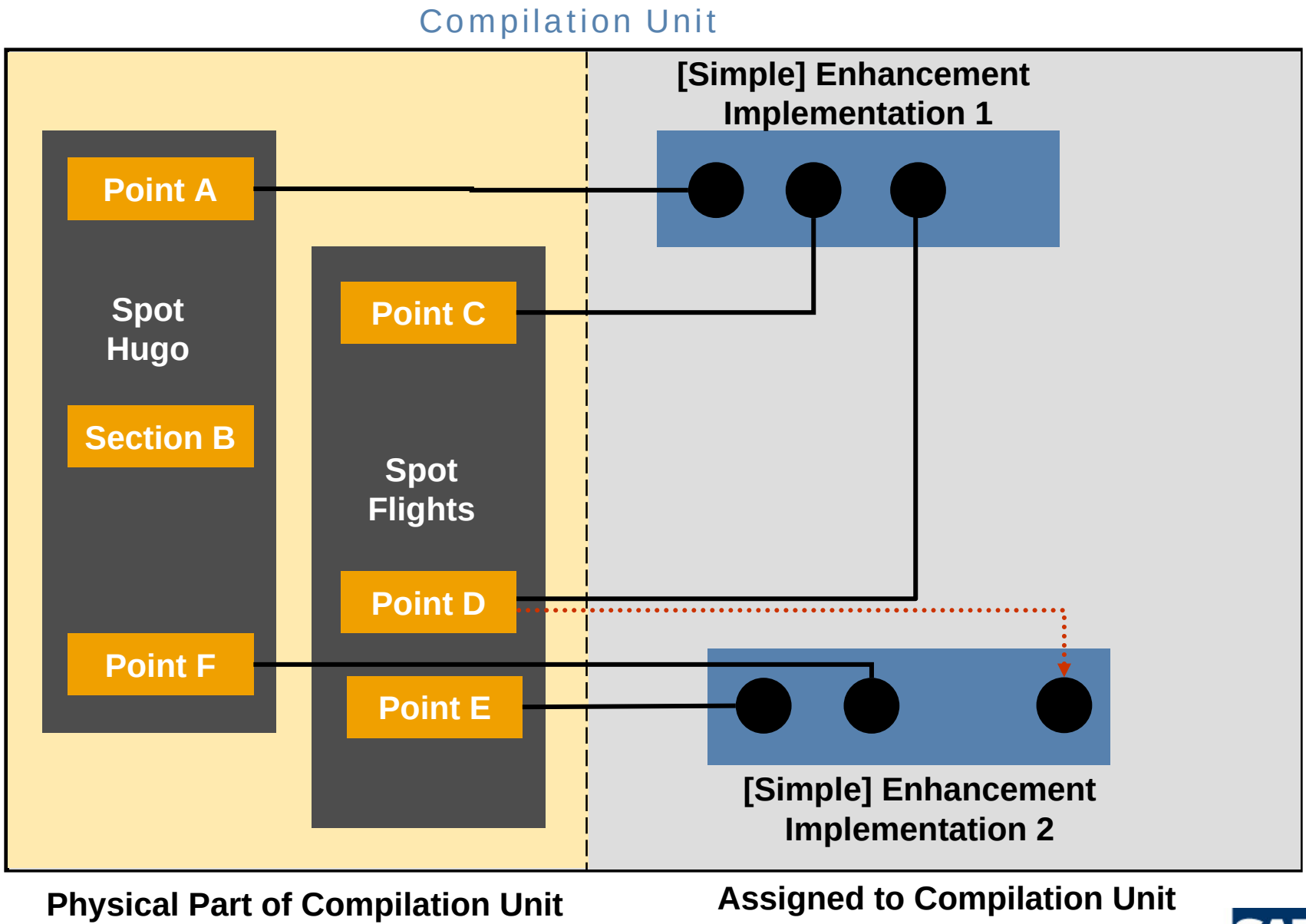
Enhancement Spots & Implementations



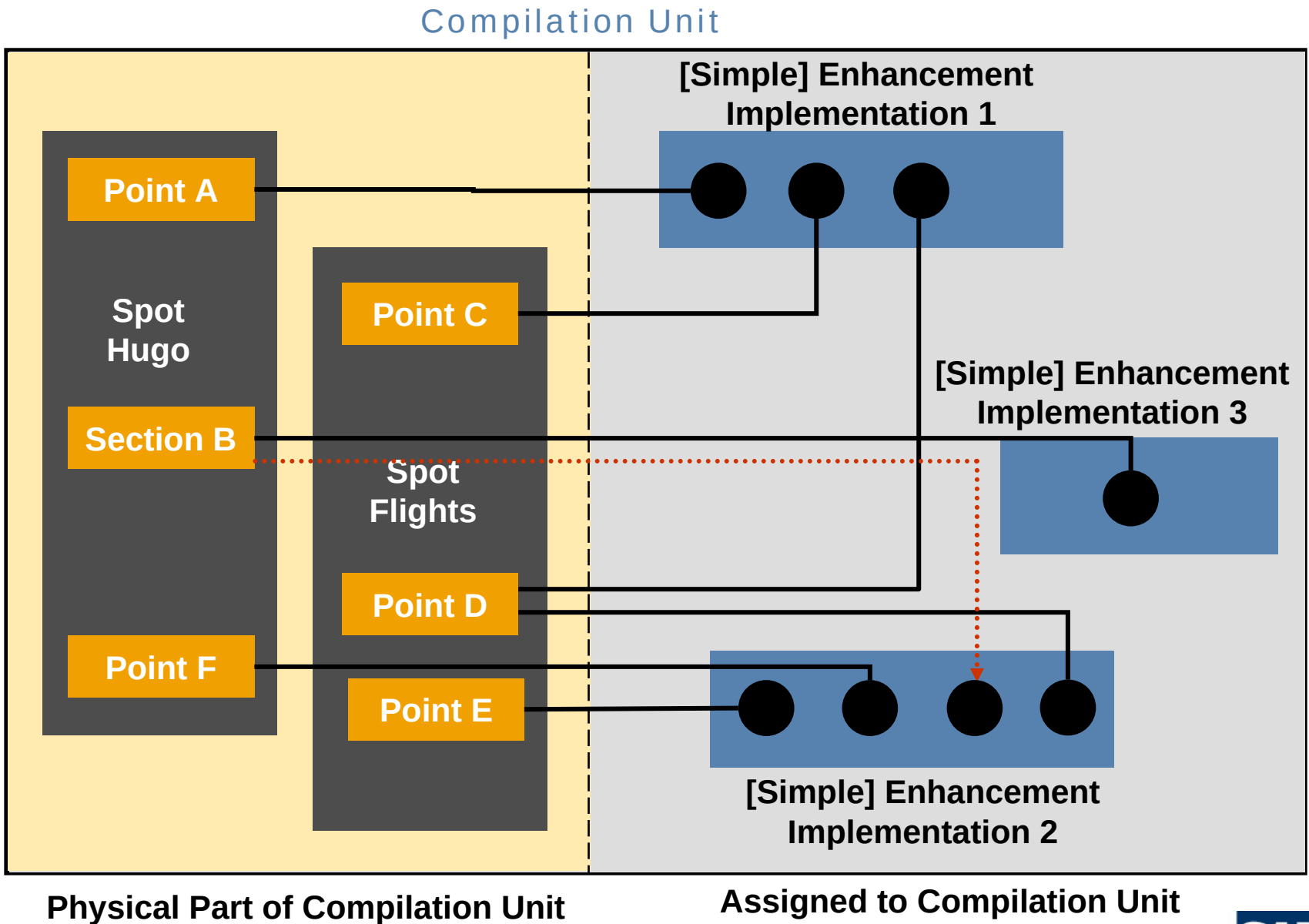
Enhancement Spots & Implementations



Enhancement Spots & Implementations

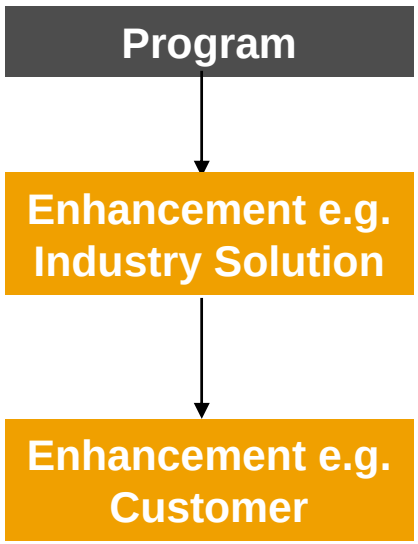


Enhancement Spots & Implementations



Overlay Enhancement

Existing Enhancements must not be modified. Enhance the Enhancement instead!



Example:

An SAP Program is enhanced by an Industry Solution.

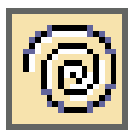
The source code of the Industry Solution enhancement is exchanged by a customer enhancement.

Editor Modes for Enhancements

Use Change Mode for creating enhancement points & sections.

- **use button  ”Display <-> Change” to switch to change mode.**

Use Enhancement Mode for creating enhancement implementations.

- **use button ”Change Enhancements”  to switch to Enhancement mode**
- **use button ”Display <-> Change”  to leave Enhancement mode**

DEMO

Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

BAdI – Technology

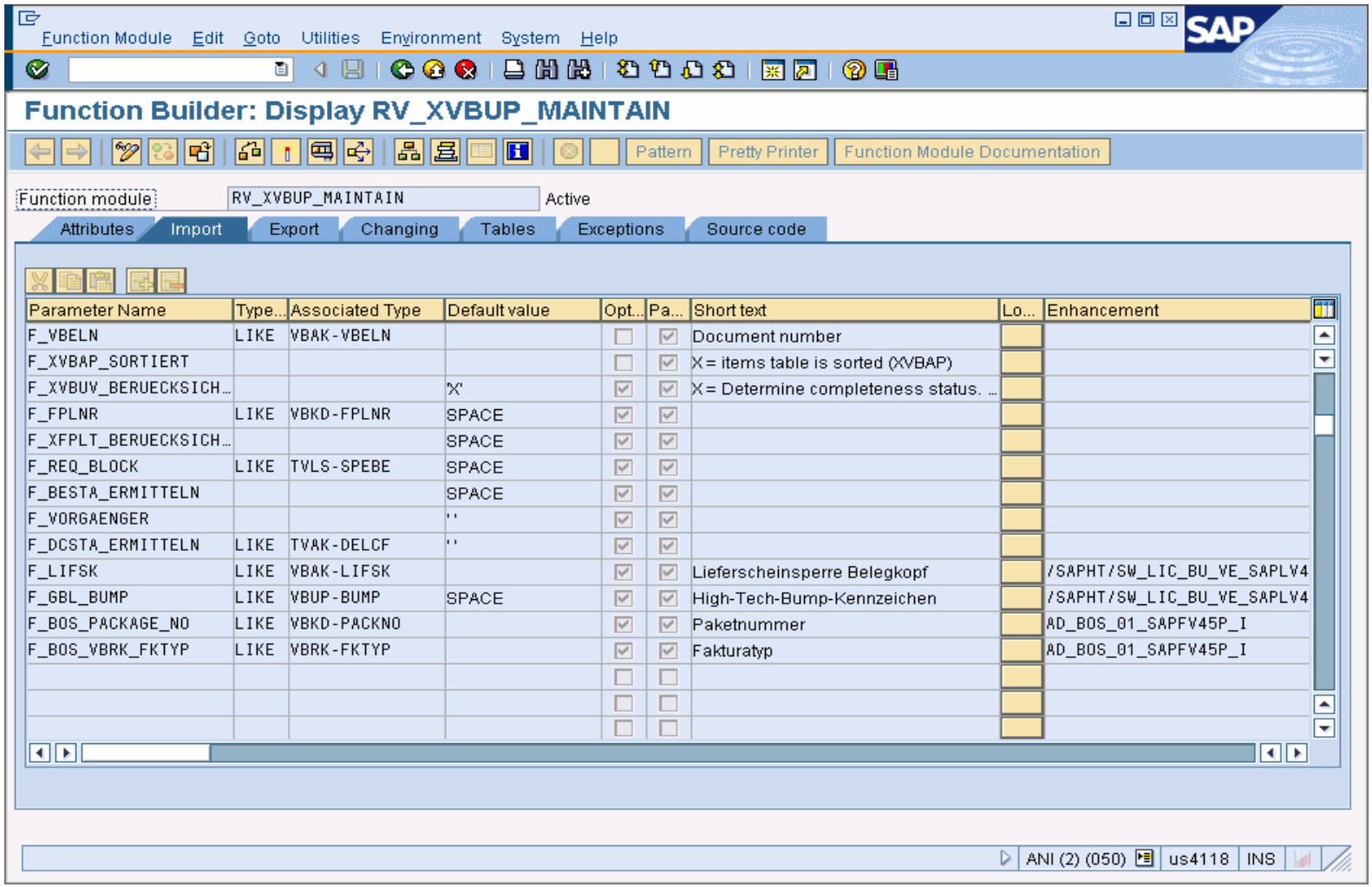
Upgrade Adjustment

Switch Framework

Summary

Function Group Enhancements allow

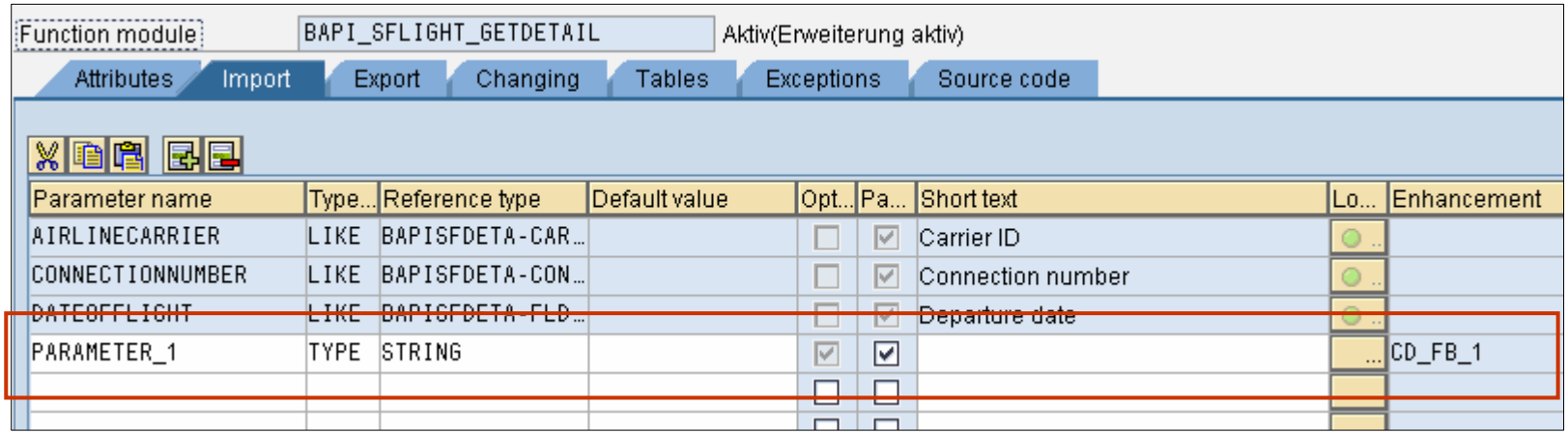
- **Adding new optional parameters to existing function modules**



Parameter Name	Type...	Associated Type	Default value	Opt...	Pa...	Short text	Lo...	Enhancement
F_VBELN	LIKE	VBAK-VBELN		☐	☑	Document number		
F_XVBAP_SORTIERT				☐	☑	X = items table is sorted (XVBAP)		
F_XVBUV_BERUECKSICH…			'X'	☑	☑	X = Determine completeness status. …		
F_FPLNR	LIKE	VBKD-FPLNR	SPACE	☑	☑			
F_XFPLT_BERUECKSICH…			SPACE	☑	☑			
F_REQ_BLOCK	LIKE	TVLS-SPEBE	SPACE	☑	☑			
F_BESTA_ERMITTELN			SPACE	☑	☑			
F_VORGAENGER			''	☑	☑			
F_DCSTA_ERMITTELN	LIKE	TVAK-DELCF	''	☑	☑			
F_LIFSK	LIKE	VBAK-LIFSK		☑	☑	Lieferscheinsperre Belegkopf		/SAPHT/SW_LIC_BU_VE_SAPLV4
F_GBL_BUMP	LIKE	VBUP-BUMP	SPACE	☑	☑	High-Tech-Bump-Kennzeichen		/SAPHT/SW_LIC_BU_VE_SAPLV4
F_BOS_PACKAGE_NO	LIKE	VBKD-PACKNO		☑	☑	Paketnummer		AD_BOS_01_SAPFV45P_I
F_BOS_VBRK_FKTYP	LIKE	VBRK-FKTYP		☑	☑	Fakturatyp		AD_BOS_01_SAPFV45P_I
				☐	☐			
				☐	☐			
				☐	☐			

Additional Parameters

Add optional parameters to existing function modules



Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

BAdI – Technology

Upgrade Adjustment

Switch Framework

Summary

Class/Interface Enhancements

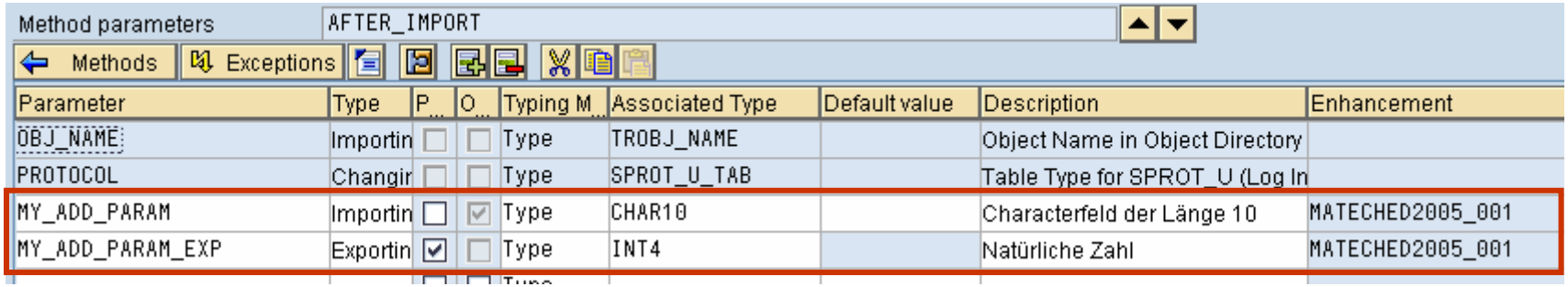
Class/Interface Enhancements allow addition of:

- **optional parameters to existing methods**
- **methods**
- **events and event handlers**
- **references to interfaces**
- **types***
- **exits to existing methods**
  - **Pre-Exit – Called at the beginning of a method**
  - **Post-Exit – Called at the End of a method**
  - **Overwrite-Exit – Replaces the original method**
  - **access to private and protected attributes of the original class***

*** as from next release**

Additional Parameters in Methods

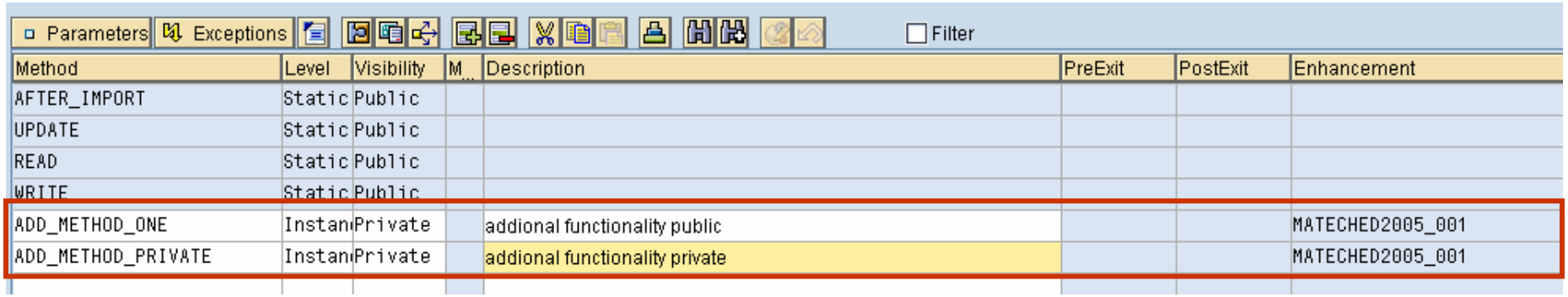
Add optional parameters to existing methods



Parameter	Type	P...	O...	Typing M...	Associated Type	Default value	Description	Enhancement
OBJ_NAME	Importin	☐	☐	Type	TROBJ_NAME		Object Name in Object Directory	
PROTOCOL	Changin	☐	☐	Type	SPROT_U_TAB		Table Type for SPROT_U (Log In	
MY_ADD_PARAM	Importin	☐	☑	Type	CHAR10		Characterfeld der Länge 10	MATECHED2005_001
MY_ADD_PARAM_EXP	Exportin	☑	☐	Type	INT4		Natürliche Zahl	MATECHED2005_001

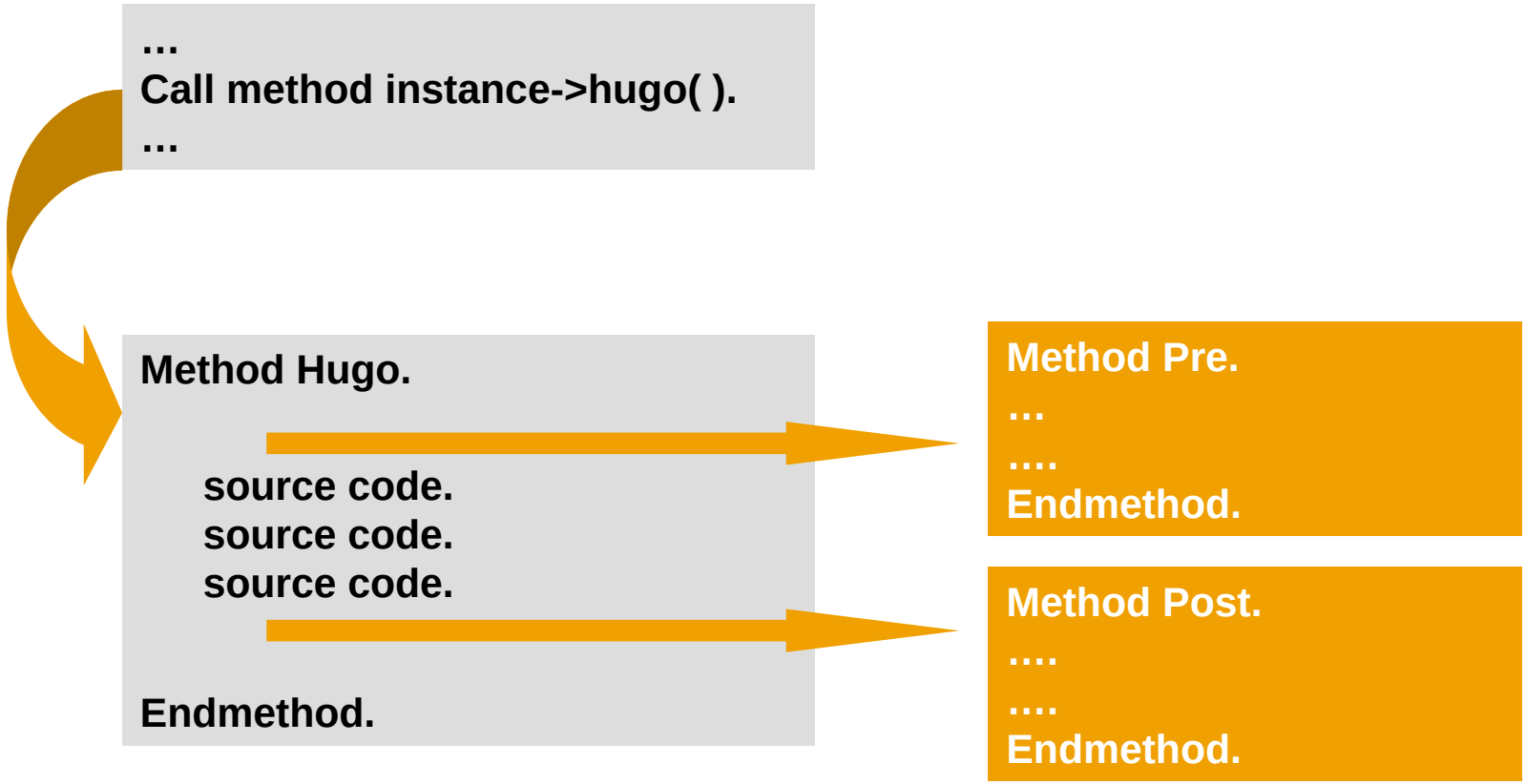
Additional Methods

Add new methods



Method	Level	Visibility	M...	Description	PreExit	PostExit	Enhancement
AFTER_IMPORT	Static	Public					
UPDATE	Static	Public					
READ	Static	Public					
WRITE	Static	Public					
ADD_METHOD_ONE	Instan	Private		addional functionality public			MATECHED2005_001
ADD_METHOD_PRIVATE	Instan	Private		addional functionality private			MATECHED2005_001

Pre/Post Exits



DEMO

Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

BAdI – Technology

Upgrade Adjustment

Switch Framework

Summary

What are Business Add-Ins

A BAdI

- **is an anticipated point of extension – these points act like sockets and exist in the original source code**
- **is a predefined anchor for an Object PlugIn**
- **has a well-defined interface in contrast to source code plugIns and is therefore more stable to changes in the original source code**
- **has switchable implementation(s) (by switches introduced by the Switch Framework)**

BAdI Patterns – (1) BAdI as a Service or Delegation

The calculation depends on the filter value. Only one active implementation is allowed for one filter value.

```
GET BADI mytaxbadi
  FILTERS LANGUAGE = sy-
langu.
..

CALL BADI mybadi->add_tax
  EXPORTING
    value = my_value
  IMPORTING
    result = my_result.
..
write / my_result.
```

language = ‚XY‘

language = ‚DE‘

```
METHOD if_mybadi~add_tax.
 result = value +
myfunc(value).
ENDMETHOD.
```

```
METHOD if_mybadi~add_tax.
  result = value * 1.16.
ENDMETHOD.
```

BAdI Patterns – (2) BAdI as classical User Exit

**The BadI allows to perform an addional step inside a standard process. The core functionlity doesn‘t need any implementation, but the implemantation may react on a core event.
Several implementations may be called in a sequence.**

```
GET BADI mybadi.
..
CALL BADI mybadi->value_changed
  EXPORTING
    value = my_value

..
```

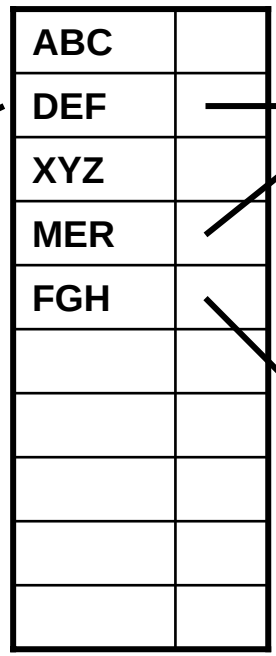
```
METHOD
if_mybadi~value_changed.
  insert value ..
ENDMETHOD.
```

```
METHOD
if_mybadi~value_changed.
  write value ..
ENDMETHOD.
```

BAdI Patterns – (3) Registry With BAdI

Use the advantages (performance, upgrade adjustment) to implement a registry of classes.

```
GET BADI myregbadi.
   FILTERS reg = 'DEF'
..
CALL BADI myregbadi->do


..
```




```
METHOD
if_mybadi~get_result.
  do …
ENDMETHOD.
```

```
METHOD
if_mybadi~get_result.
  do ..
ENDMETHOD.
```

New BAdIs – New Features

- **Are integrated directly in the ABAP Language/Runtime**
- **Improved filter support allows non-character filter types (packed, numeric, string) and complex filter conditions**
- **Enable reusable implementation instances**
- **Different kinds of default implementations**
- **Control of the lifetime of implementations (BAdI-context)**
- **Allow for inheritance of implementations**

Comparison: Usage of Old BAdIs vs. New BAdIs

With Classic BAdI

```
DATA: bd TYPE REF TO if_intf.
DATA: flt TYPE flt.

CALL METHOD cl_exithandler=>
get_instance
  EXPORTING
    exit_name = `BADI_NAME`
  CHANGING
    instance = bd.

flt-lang = `D`.
CALL METHOD bd->method
  EXPORTING
    x       = 10
    flt_val = flt.
```

selecting implementations and issuing calls is mixed

calls are redirected over a proxy class

With New BAdI

```
data bd type ref to badi_name.
get badi bd filters lang = `D`.
call badi bd->method
  exporting x       = 10.
```

Selection occurs when the handle is requested

Implementations are called directly (without a proxy)

Evaluation at Compile Time (1)

The new BAdI evaluates as much information as possible during compile time.

Better Performance/Lower Memory consumption

- **Database access only at compile time**
- **Statically typed comparisons at runtime**
- **Internal handle-class integrated in SAP Kernel**

➔ **40-600 times faster than Classic BAdIs**

Evaluation at Compile Time (2)

Active implementations are evaluated at compile time and included in the load of the BAdI-handle.

- **Switch Framework – only implementations, that are switched on in at least one client are considered.**
- **Constant Filter values – Implementations that do not match the filter are excluded**
- **Active Flag – Only active implementations are considered**

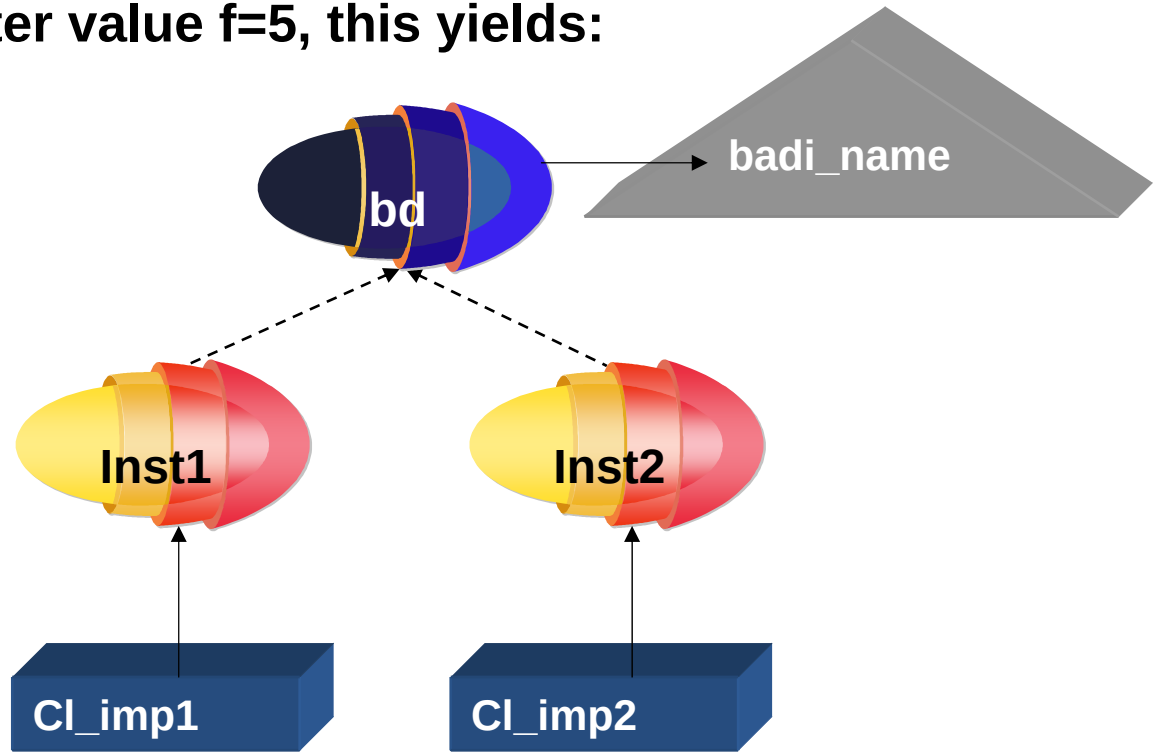
Special optimizations for BAdIs with

- **no active implementations:**
  - **‘GET BADI’ is ignored while compiling**
  - **‘CALL BADI’ just needs the time for a simple if-statement.**
- **one active implementation**
  - **Direct call of that implementation**

BAdIs are represented by a reference to BAdI-Handles:

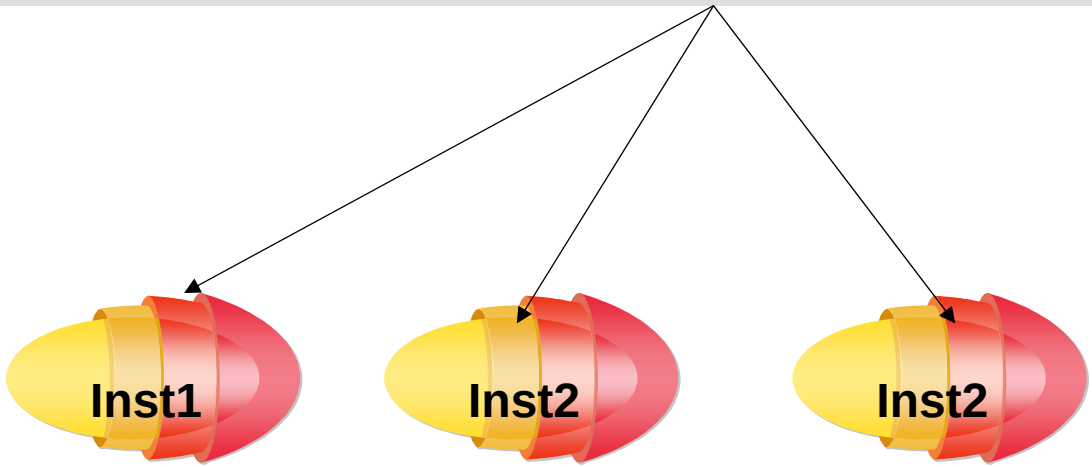
```
DATA bd type ref to badi_name.
GET BADI bd FILTER f = 5.
```

If there are two implementations of badi_name that are selected for the filter value f=5, this yields:

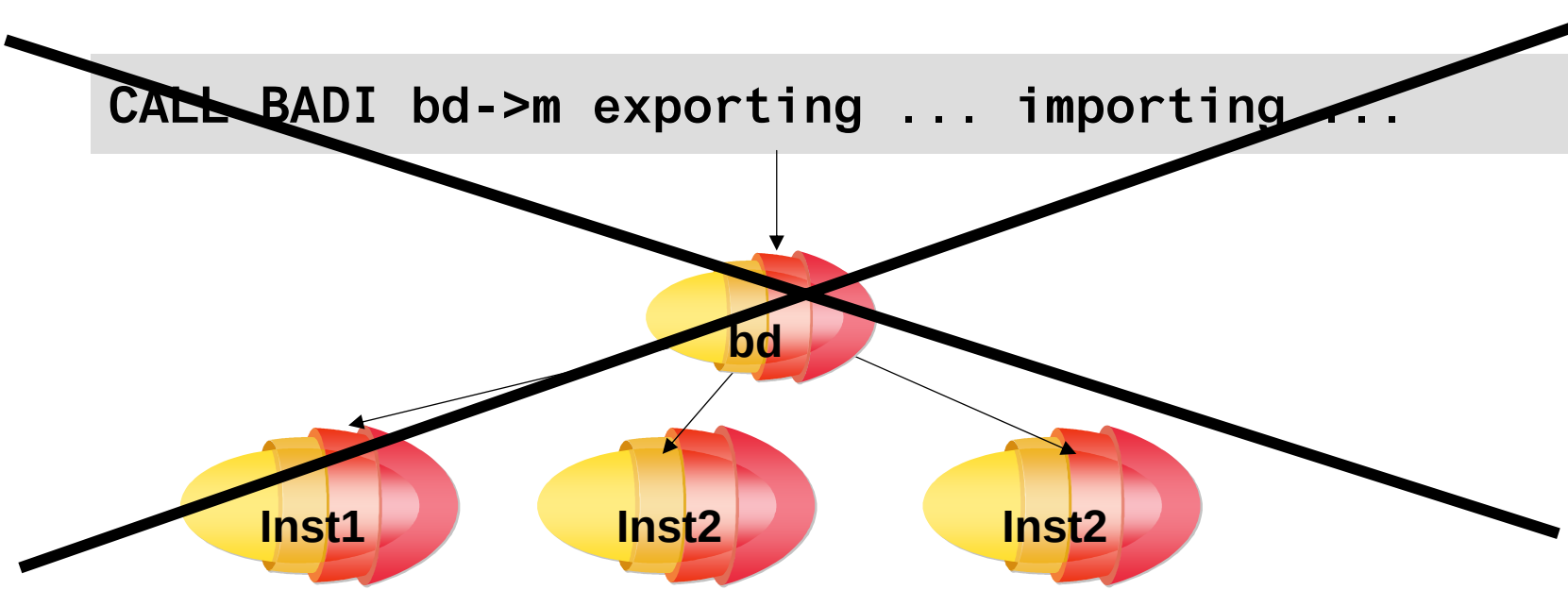


BAdI-Handle is No Proxy

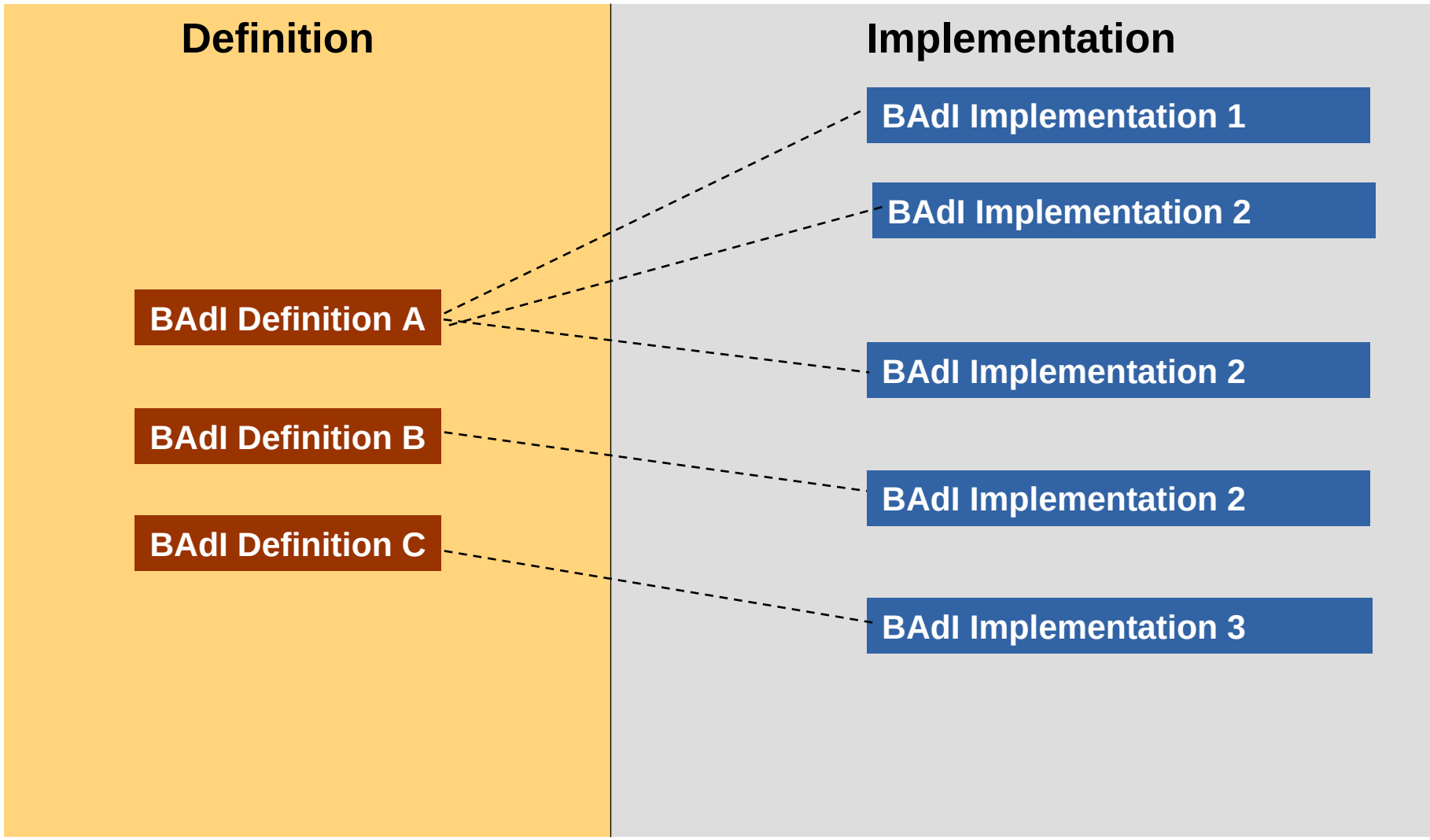
```
CALL BADI bd->m exporting ... importing ...
```



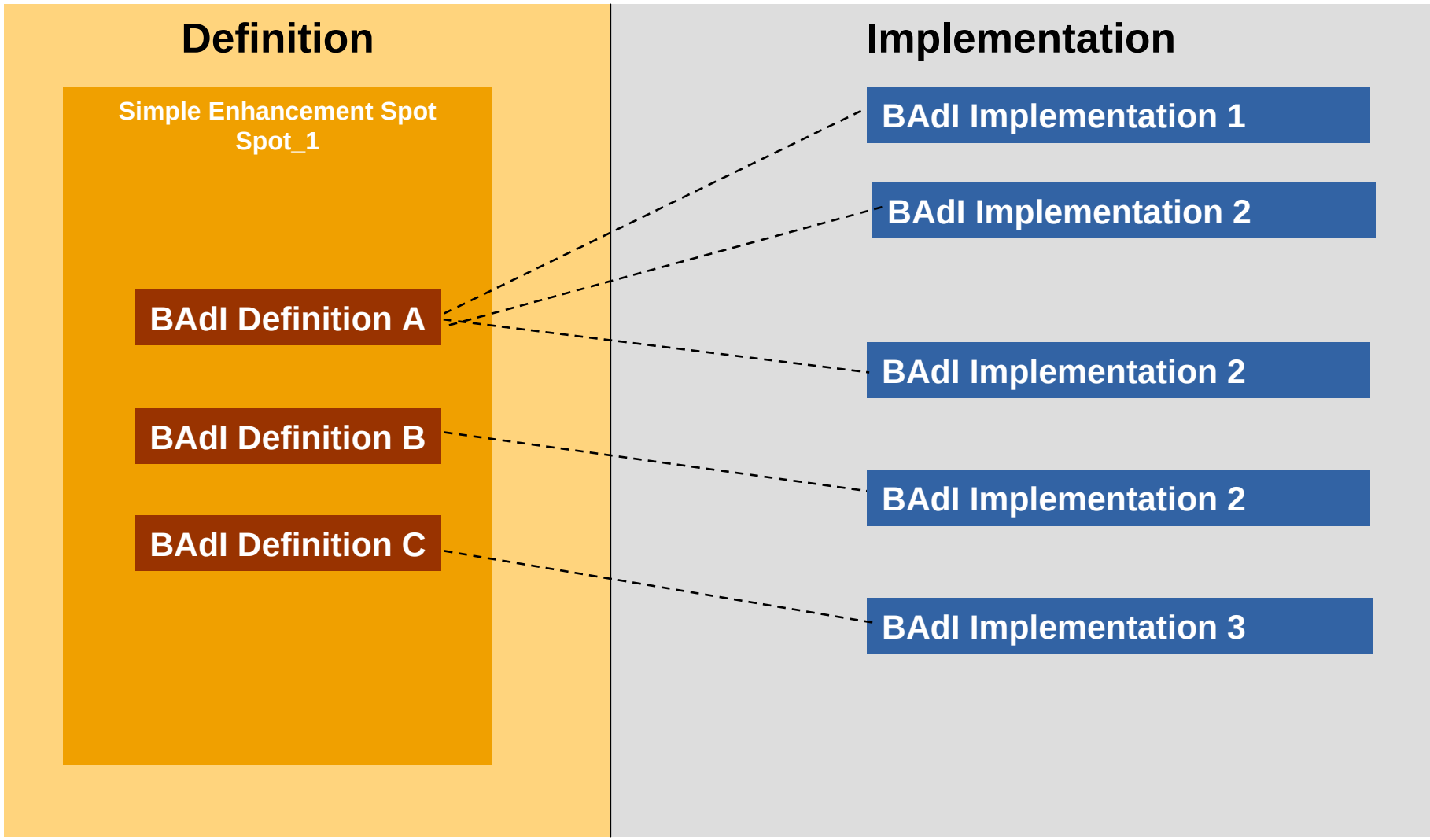
~~CALL BADI bd->m exporting ... importing ...~~



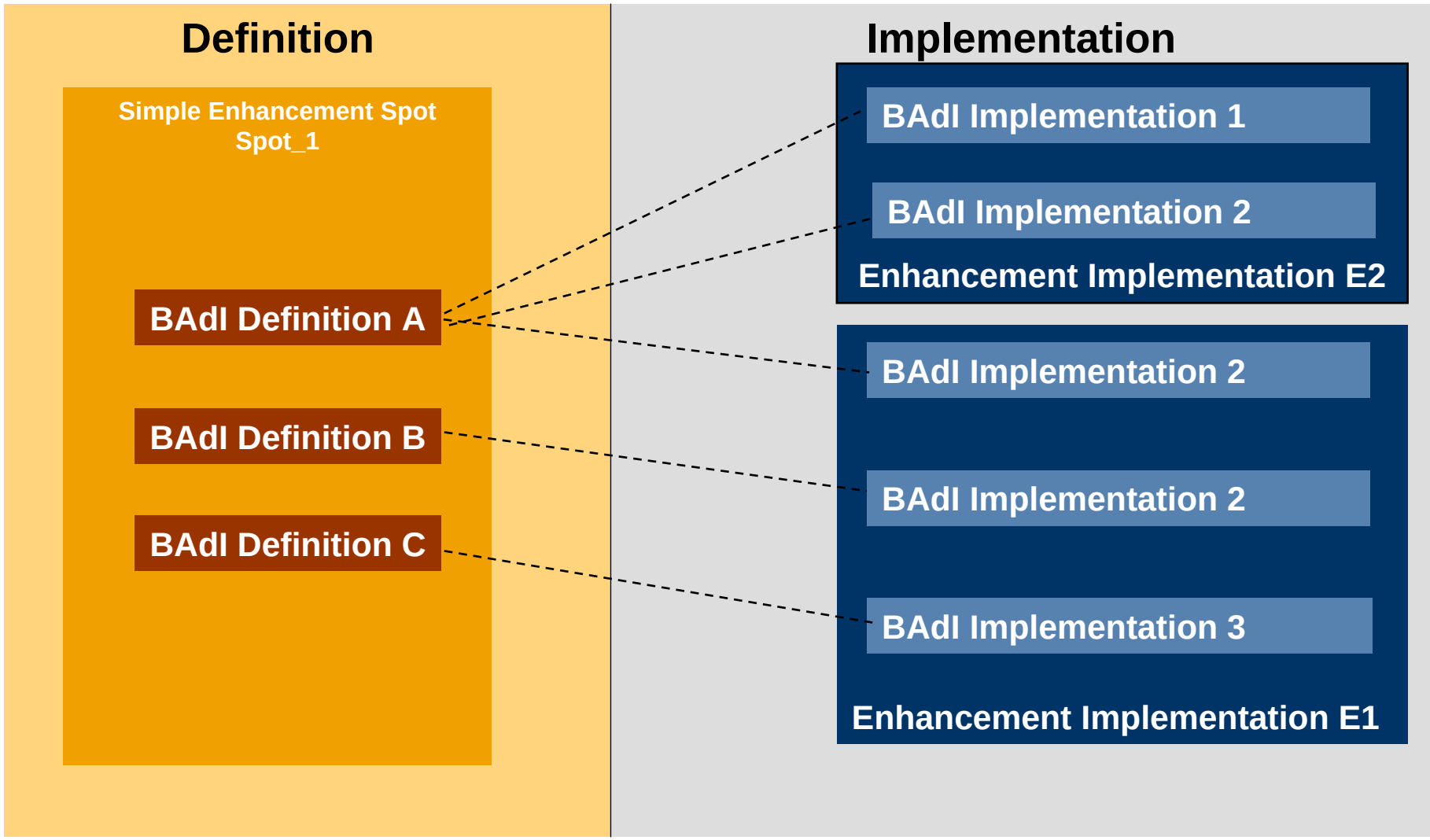
BAdI-Definition and Enhancement Spot



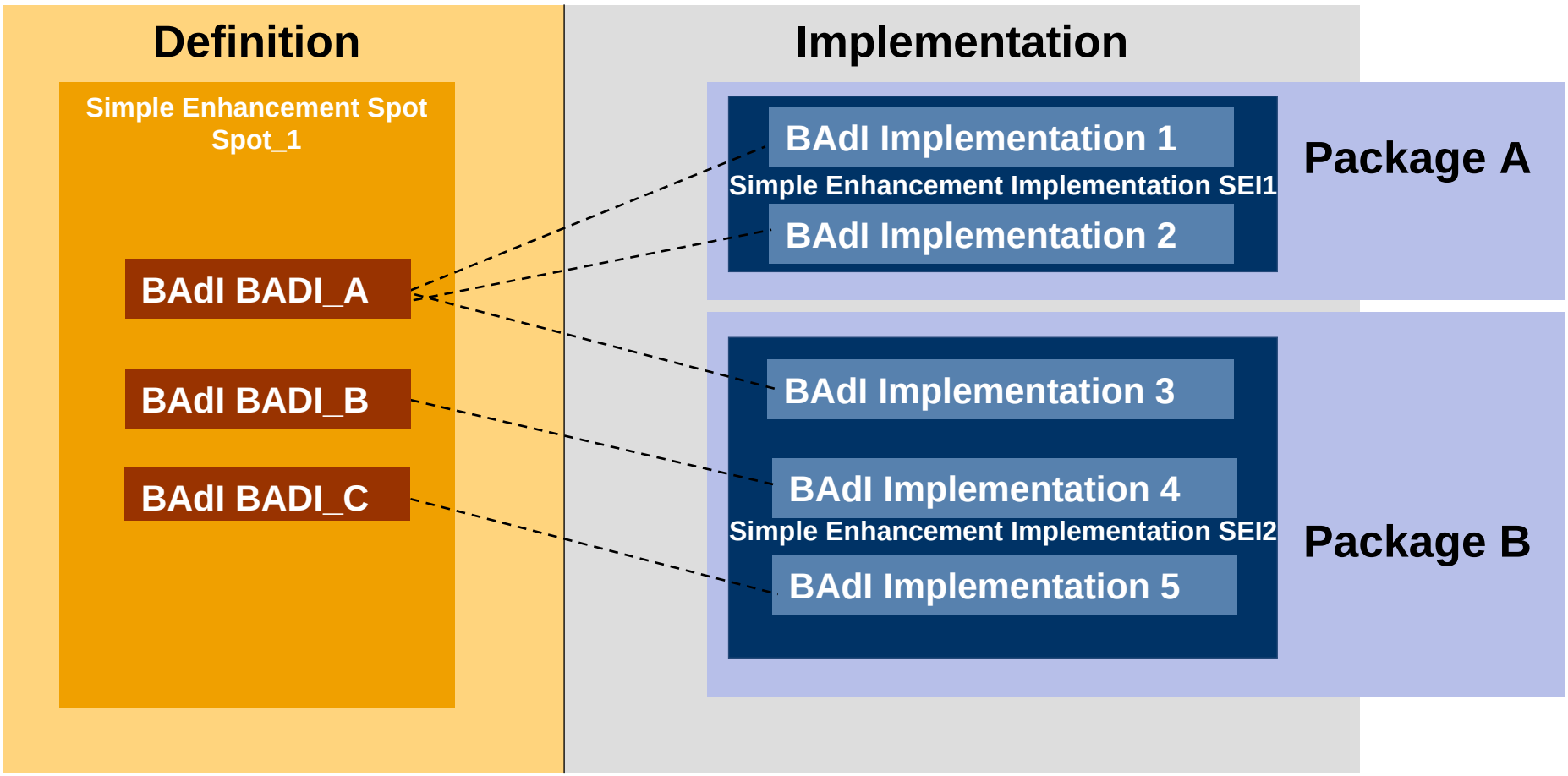
BAdI-Definition and Enhancement Spot



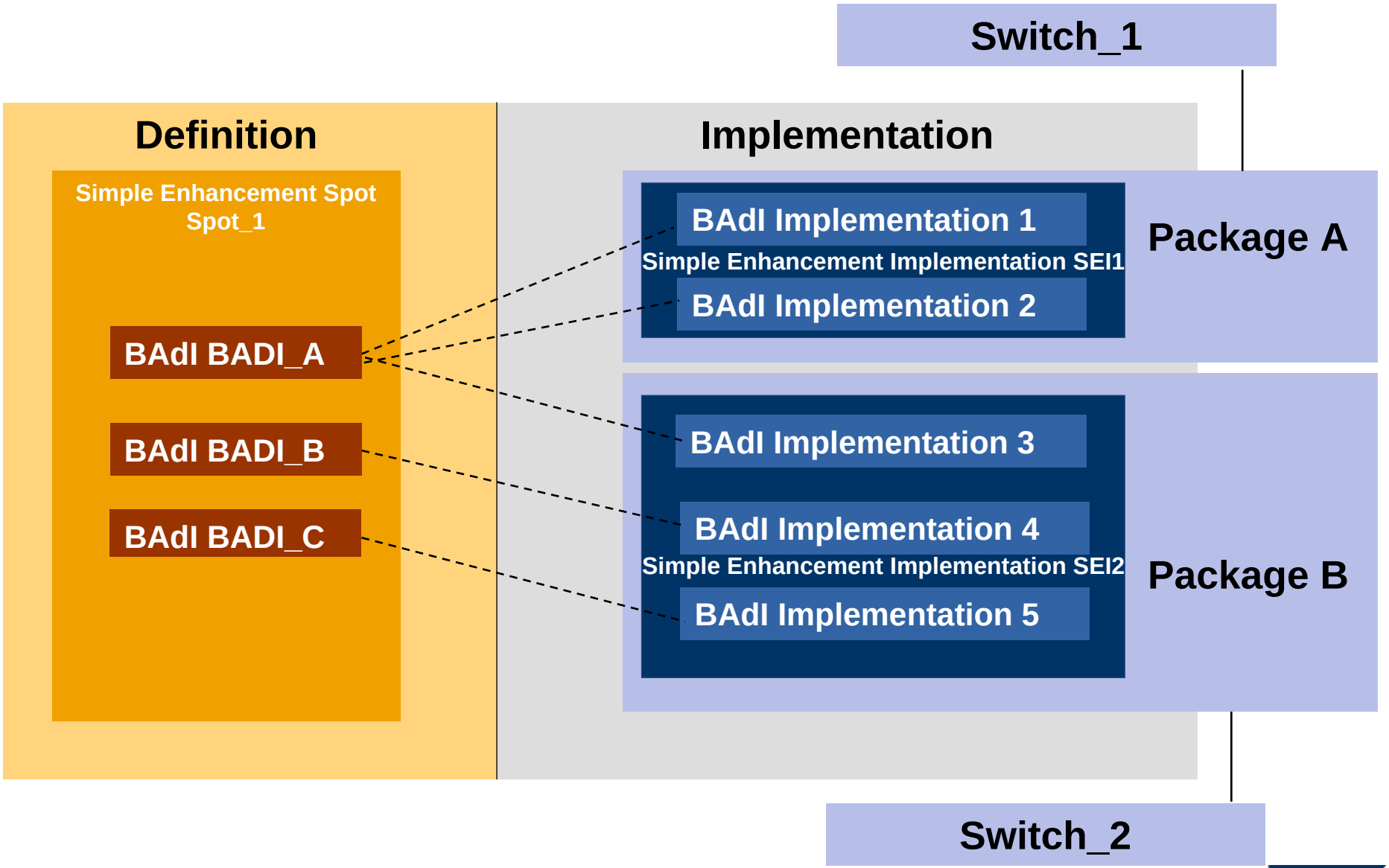
BAdI-Definition and Enhancement Spot



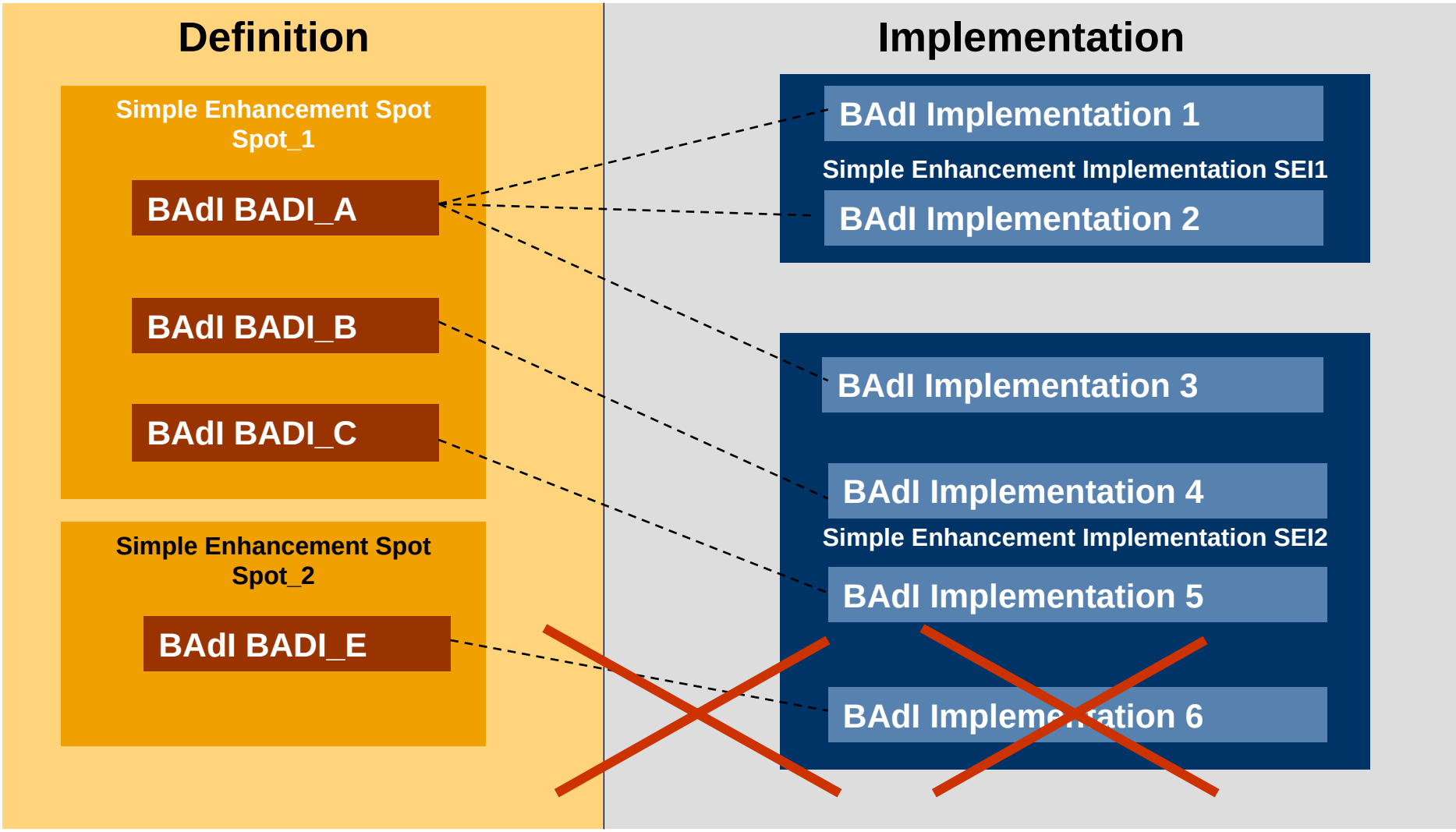
Enhancement Spots and Enhancement Implementation (2)



Enhancement Spots and Enhancement Implementation (2)



Enhancement Spots and Enhancement Implementation (5)



Default Implementations

Types of Implementations

- **A BAdI-Definition may have an associated fallback class**
- **A BAdI-Implementation is either default or non-default**

Selection Procedure: (during GET BADI)

1. **Apply selection to all non-default implementations**
2. **if nothing has been selected, apply selection to all default implementations**
3. **if still nothing has been selected and there is a fallback class, take the default class**

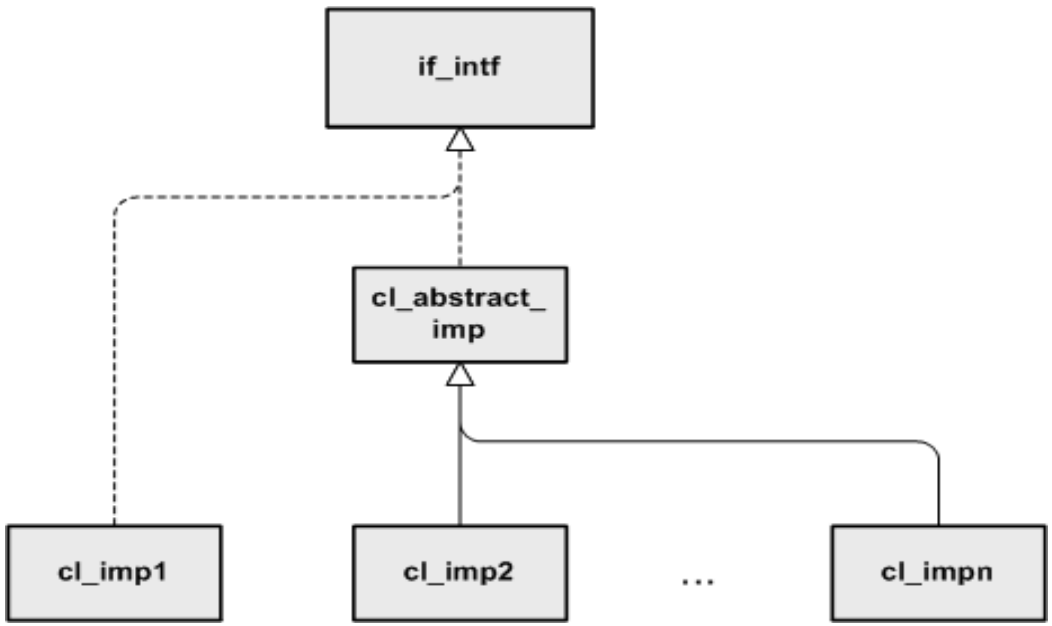
Implementation Inheritance

Together with the BAdI, abstract or normal implementation classes may be supplied

`cl_imp1` has to implement all interface methods

`cl_imp2`, ... implement methods not supplied by `cl_abstract_imp` or override some methods

It is also allowed to inherit from the default class or example class, if they are not defined as final

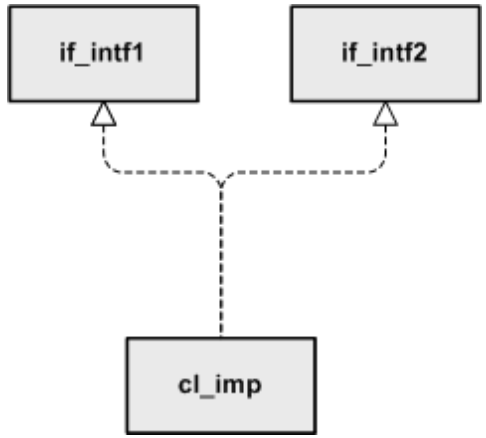


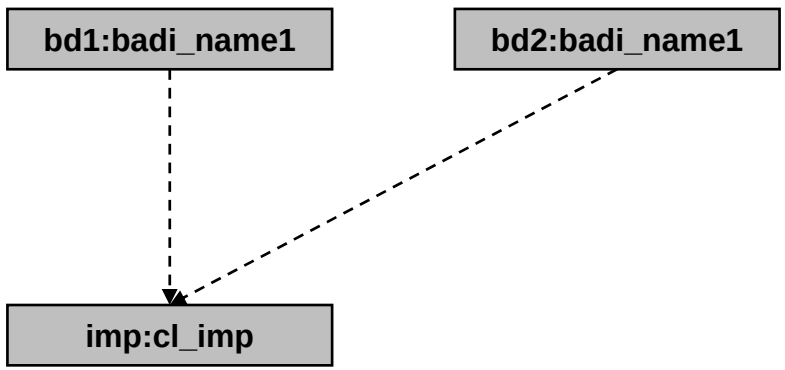
Sharing Data Within BAdIs

There are two BAdIs `badi_name1` and `badi_name2`, with interfaces `if_intf1` and `if_intf2`, respectively

`cl_imp` implements both of these interfaces

```
DATA: bd1 TYPE ref to badi_name1,
      bd2 TYPE ref to badi_name2.

GET BADI bd1.
GET BADI bd2.
```



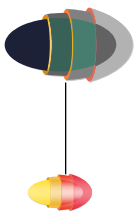


Types of Instantiations

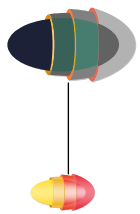
ever new instances are created (Scenario 1)

per implementation class there is one instance (Scenario 2)

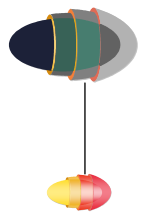
per *context reference* and implementation class there is one instance



ctx_a



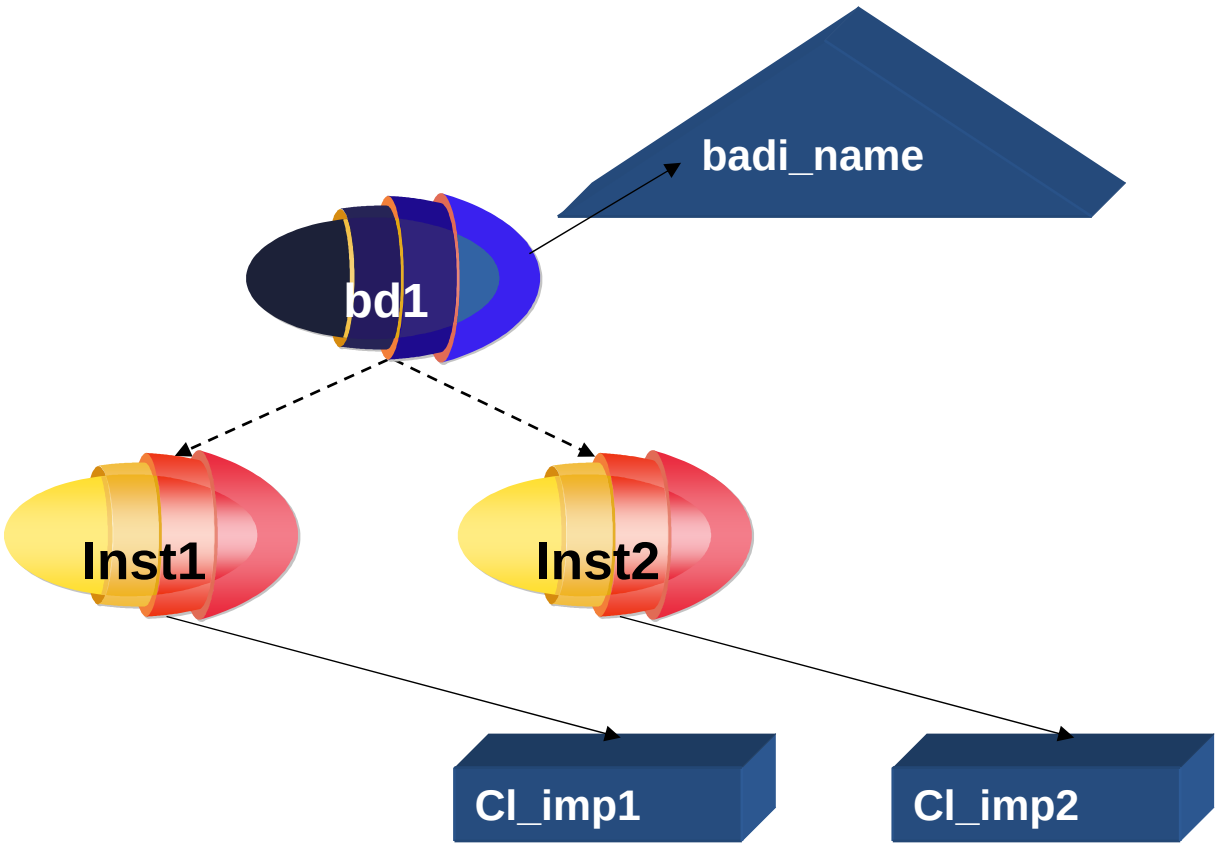
ctx_b



ctx_c

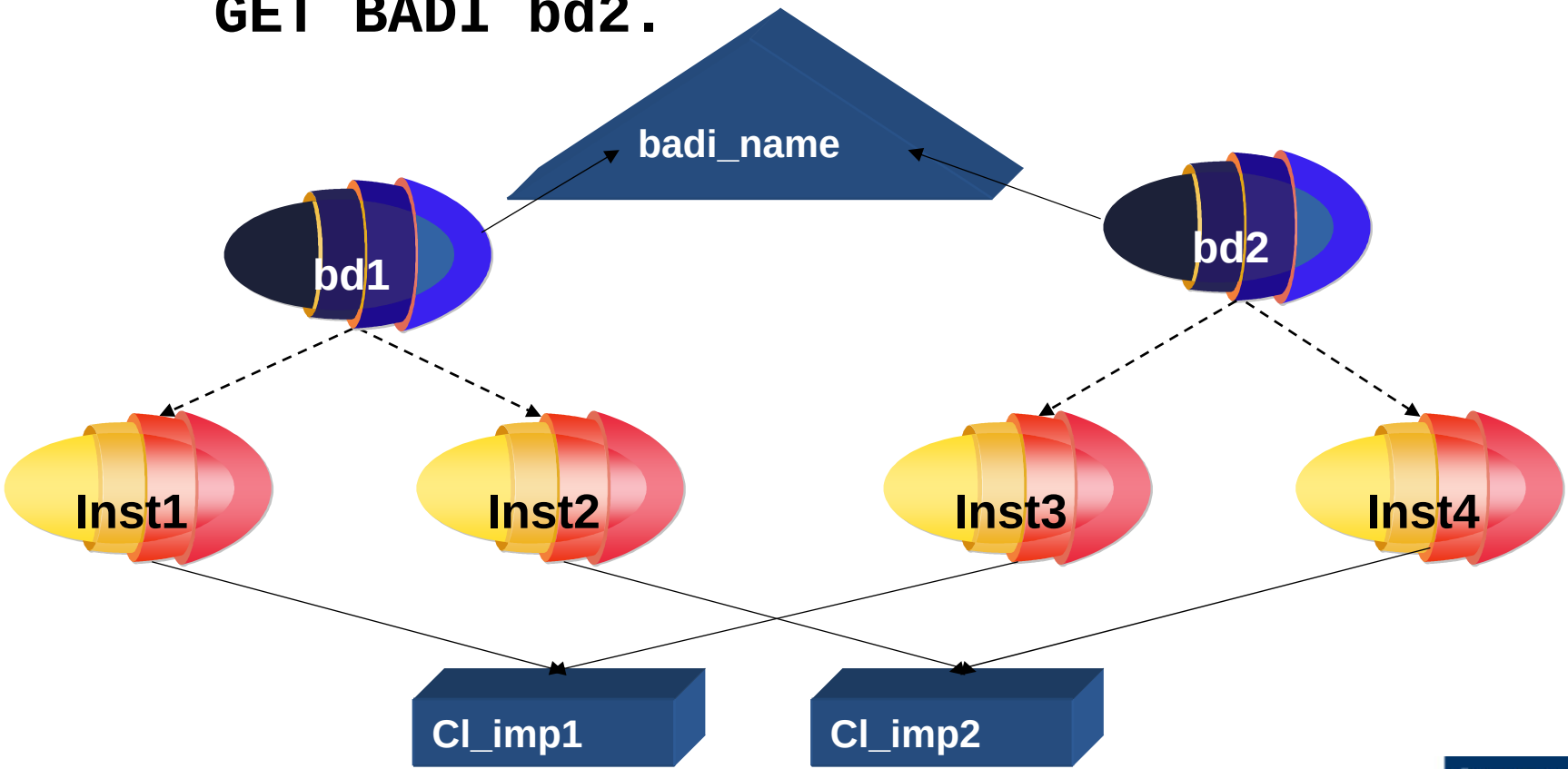
Instantiation With Option No Reuse

```
DATA  bd TYPE ref to badi_name.

      GET BADI bd1.
```



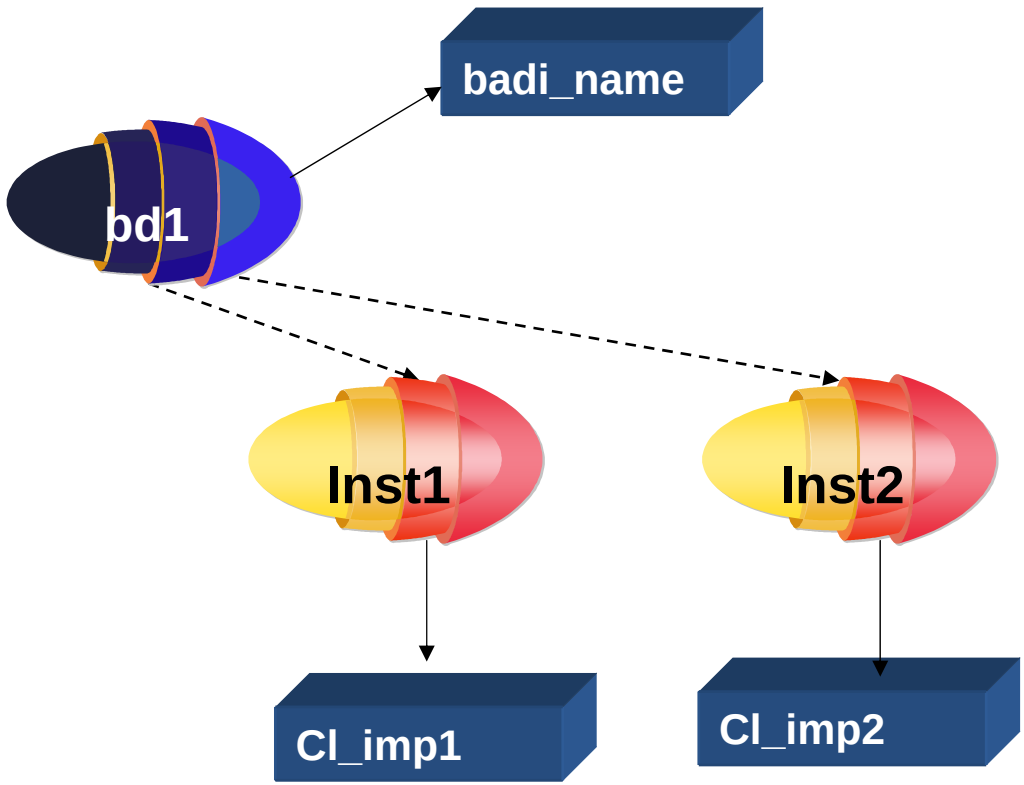
Instantiation With Option No Reuse

```
DATA: bd1 TYPE ref to badi_name,
      bd2 TYPE ref to badi_name.

      GET BADI bd1.

      GET BADI bd2.
```



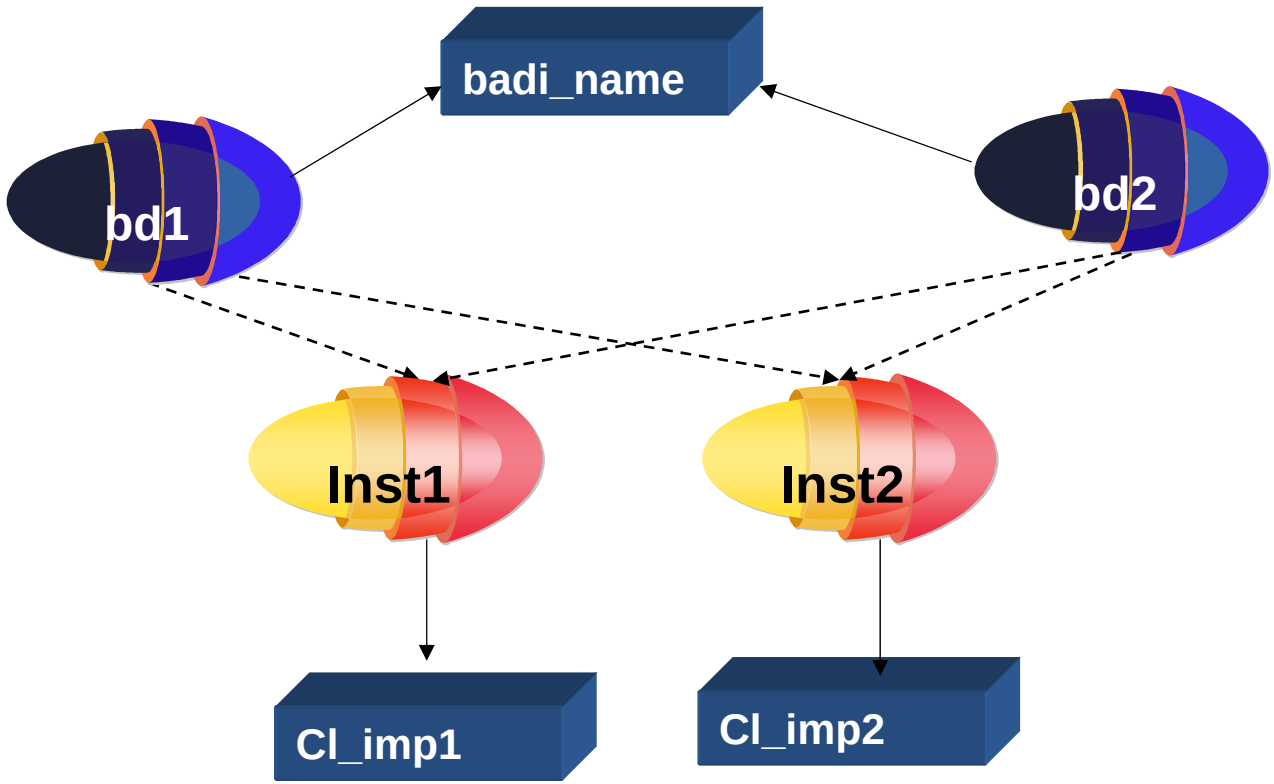
Instantiation With Option Reuse

```
DATA: bd1 TYPE ref to badi_name.
      GET BADI bd1.
```



Instantiation With Option Reuse

```
DATA: bd1 TYPE ref to badi_name,
      bd2 TYPE ref to badi_name.

      GET BADI bd1.
      GET BADI bd2.
```



Instantiations and Contexts (1)

With a context there is one instance per *context reference* and implementation class.

A *context class* is a class implementing `if_badi_context`, a *context reference* is a reference to an instance of a context class

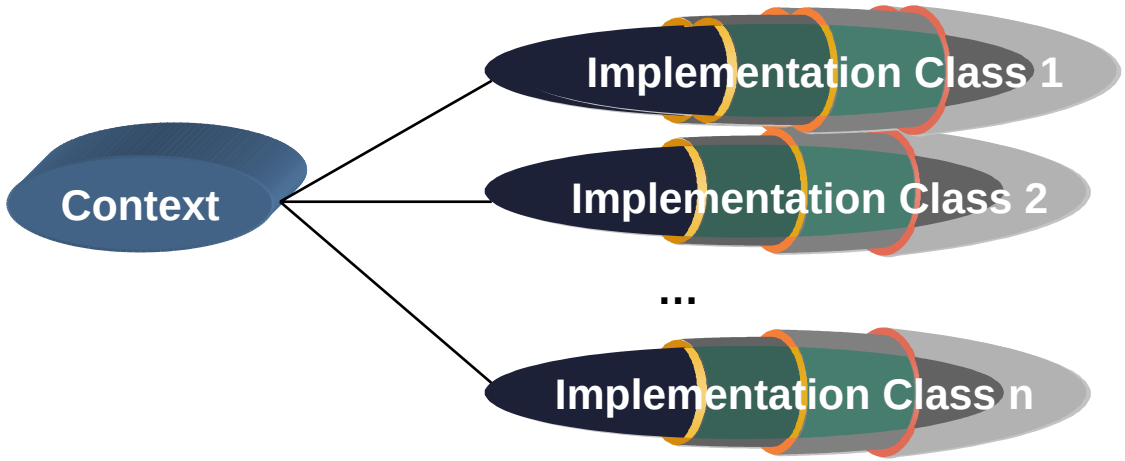
In ABAP:

```
DATA ctx TYPE REF TO cl_myctx.
 …
GET BADI bd CONTEXT ctx.
```

Passing the same context yields identical implementation instances

Instantiations and Contexts (2)

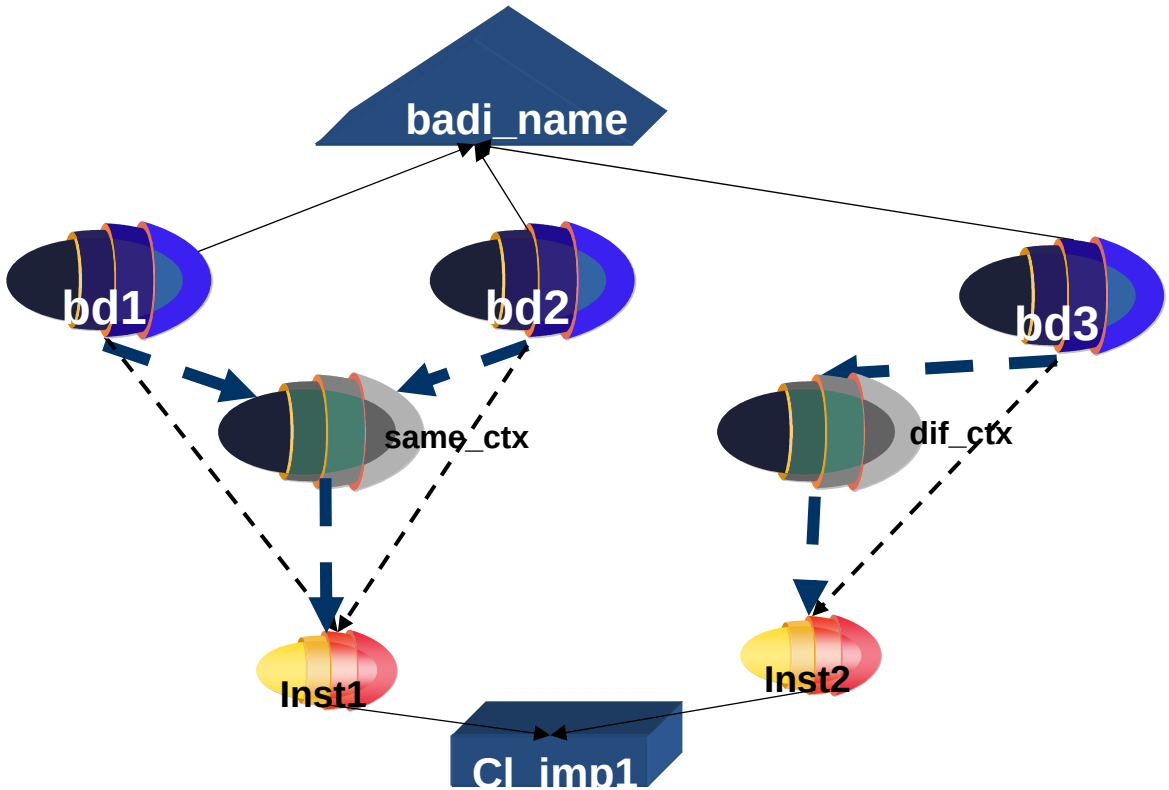
BAdI Implementations assign themselves to a given BAdI Context:



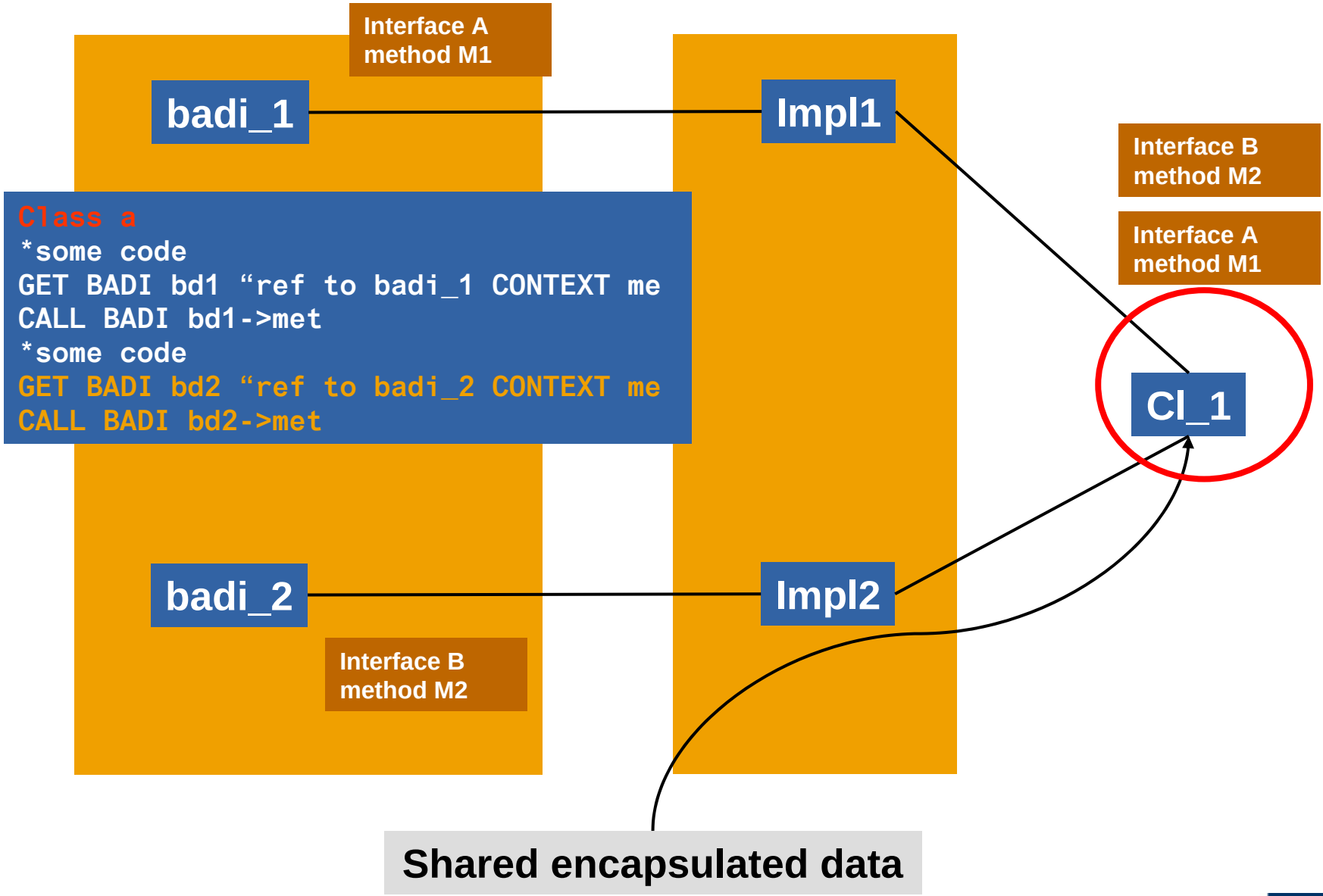
- **Lifetime of implementations is bound to the lifetime of the context.**
- **Whenever 'GET BADI' is called with the same context and a filter value which leads to the same implementation implementation class, the already created instance is used.**
- **Performance improvement**
- **Holds data over different method calls or even BAdIs.**

Instantiations With Context

```
DATA:   bd1 TYPE ref to badi_name,
        bd2 TYPE ref to badi_name,
        bd3 TYPE ref to badi_name.

        GET BADI bd1 CONTEXT same_ctx.
        GET BADI bd2 CONTEXT same_ctx.
        GET BADI bd3 CONTEXT dif_ctx.
```



Encapsulating Data for Two BAdIs Within One Class



BAdI Migration

Automatic migration by selecting utilities→migration from BAdI Builder (SE18)

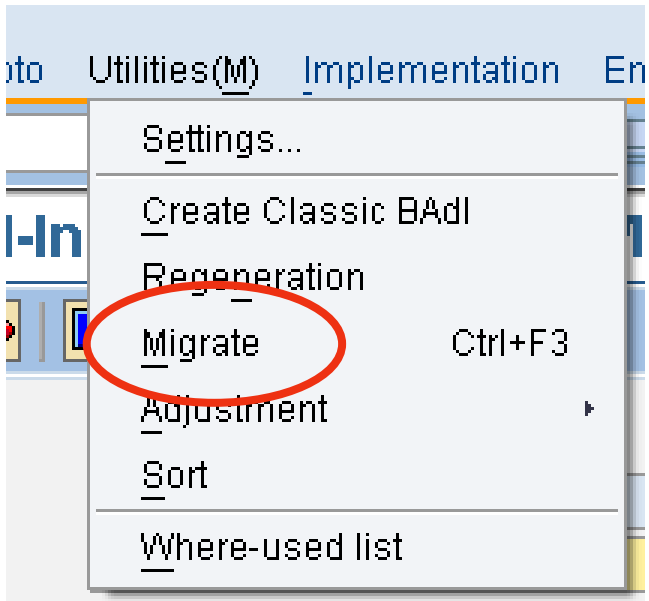
- Specify Enhancement Spot for BAdI Definition
- Specify Enhancement Implementation for BAdI Implementation

→ no special knowledge necessary

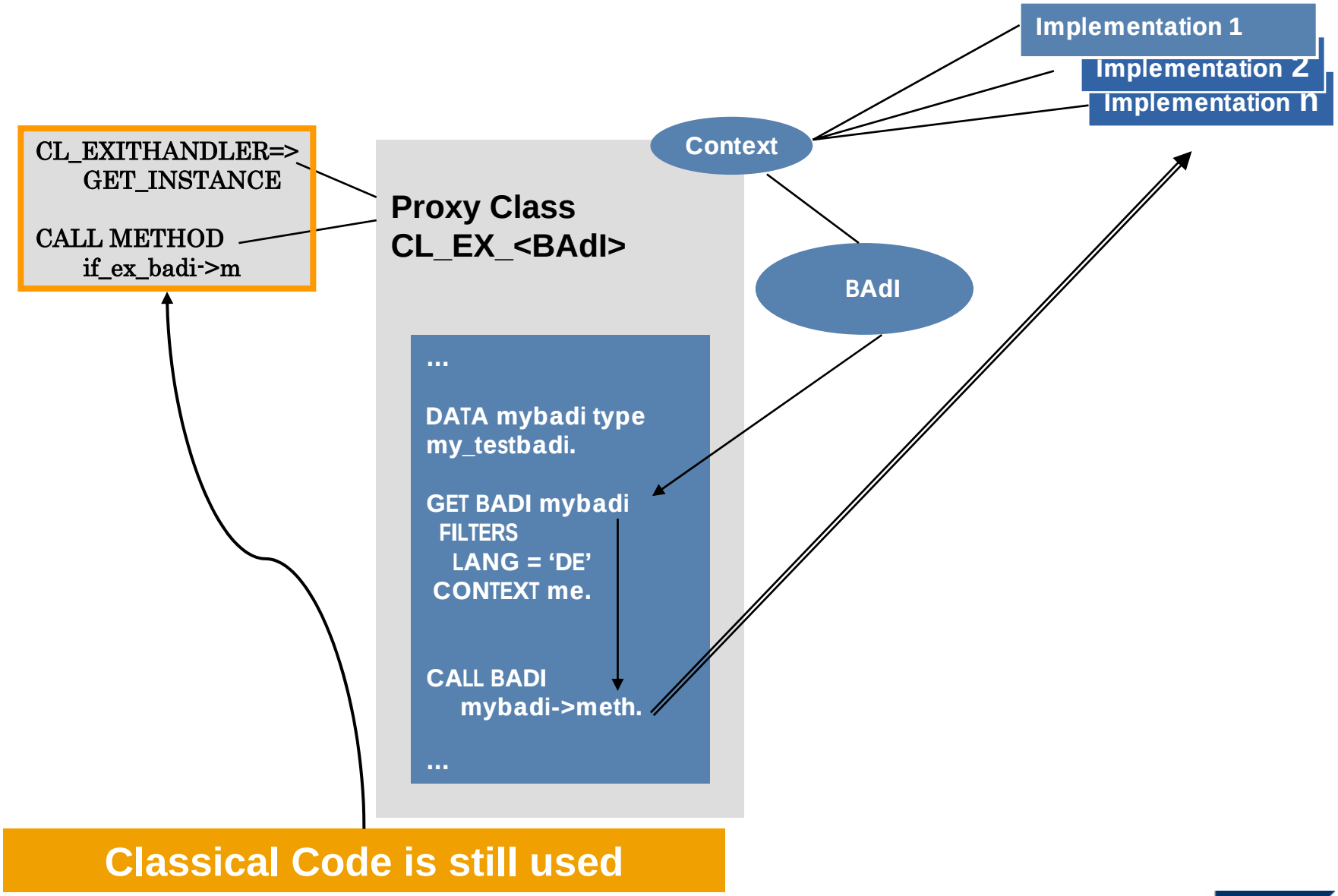
→ effort: 5 minutes per BAdI.

BAdI Migration

Automatic migration by selecting utilities → migration from BAdI Builder (SE18)



Automatic (Partial) BAdI Migration



BAdI Migration

Complete migration:

- Delete the classic BAdI.
- Change Context Settings if you need no context.
- Find all calls of the classic BAdI by GET_INSTANCE and reprogram the BAdI call using the new commands ‘GET BADI’ and ‘CALL BADI’.
- For BAdIs that are called more than once in one program the context settings may be changed to improve the performance.

➔ Expert knowledge of the application necessary

➔ Effort: from some minutes up to some days per BAdI

Performance Comparison

A BAdI call is ca. 7.5% slower than a method call!

The more implementations defined, the higher is the improvement on performance

Migration of Implementations

Normally BAdI definition and implementations are defined in different systems.

- **The after import method of a BAdI definition writes ‘Call transaction SPAU’ in the transport log.**
- **After import or upgrade perform a manual migration of BAdI implementations by using transaction SPAU.**

Dynamic BAdI (Next Release)

Syntax: bd is either a concrete BAdI-Handle or a generic BAdI-Handle (i.e. TYPE REF TO CL_BADI_BASE)

```
GET BADI { badi [FILTERS f1 = x1 f2 = x2 ...] }
         | { badi TYPE (name)
                  [ {FILTERS f1 = x1 f2 = x2 ...}
                           | {FILTER-TABLE ftab}] }
         [CONTEXT con].


CALL BADI { badi->meth parameter_list }
                  | { badi->(meth_name) {parameter_list
                  |parameter_tables} }.
```

Restricted Filter Values (Next Release)

BAdI is restricted to exactly one filter and the implementation only allows '=' and 'OR' in the filter definition.

Use case

1.BAdIs used in Frameworks often have only one key to select BAdI implementations e.g.: ESI

➔ A special implementation for these BAdIs will allow to keep efficiency even when thousands of BAdI-Implementations for one BAdI exist in a system

2. You want to program a registry application. Some registered code is called depending on a registry key.

➔Reuse the BAdI Framework for a fast registry implementation including upgrade support. A method for evaluation all implemented filter values is:
`CL_ENH_BADI_RUNTIME_FUNCTIONS=>GET_IMPLS_FOR_LIMITED_BADI(name)`

DEMO

Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

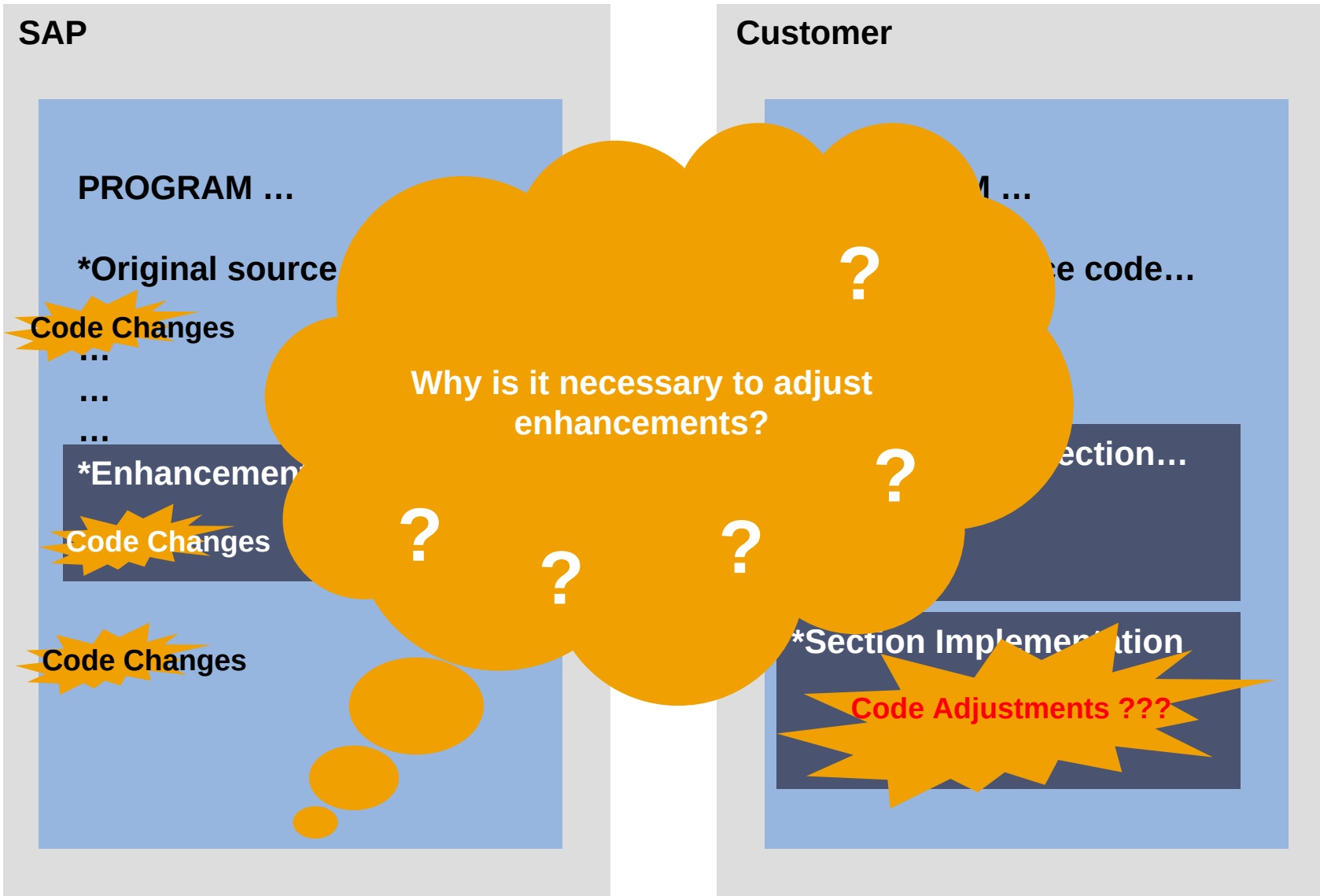
BAdI – Technology

Upgrade Adjustment

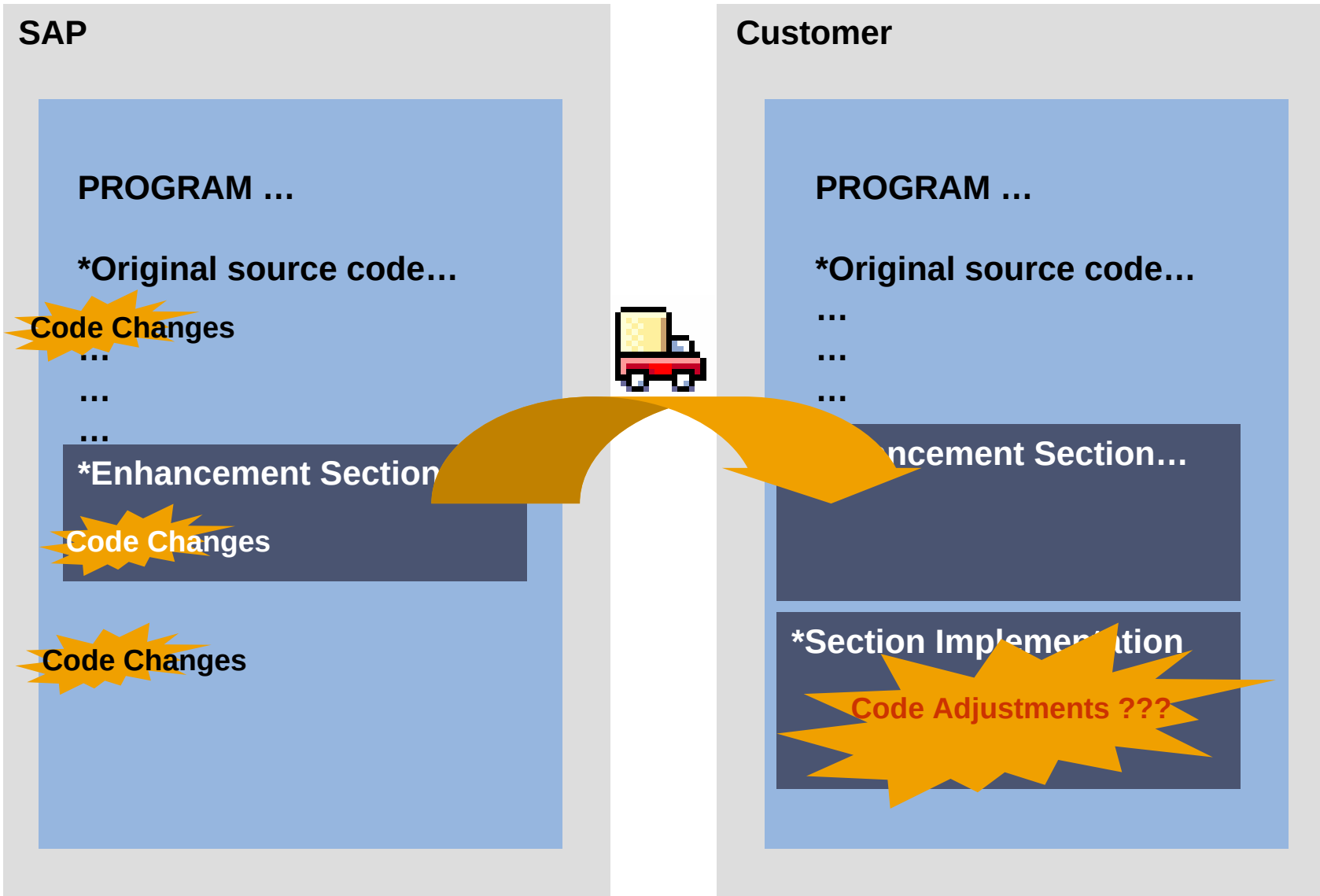
Switch Framework

Summary

Reasons for Upgrade Support I



Reasons for Upgrade Support II



Objects to be Adjusted

ABAP Source code with enhancements

- **The standard source code between ENHANCEMENT-SECTION and END-ENHANCEMENT-SECTION has changed.**

Function modules with enhancements

- **A new importing parameter was declared that has the same name as an importing parameter introduced by an enhancement.**

Classes and interfaces with enhancements

- **An enhanced method was deleted.**

BAdIs

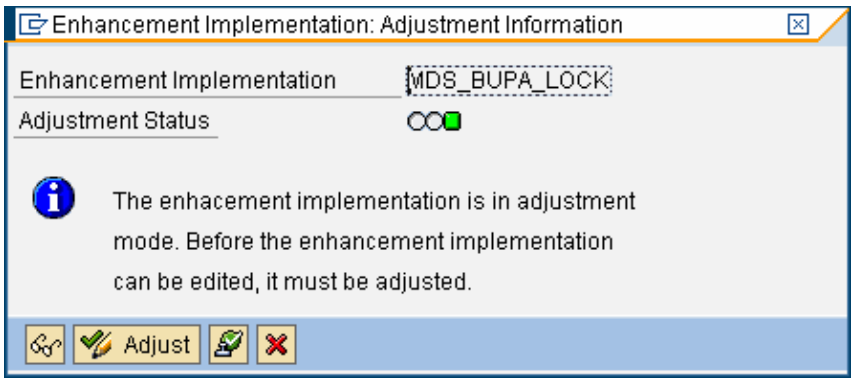
- **BAdI interface was changed or a filter was deleted**

How to Recognize Necessary Adjustments?

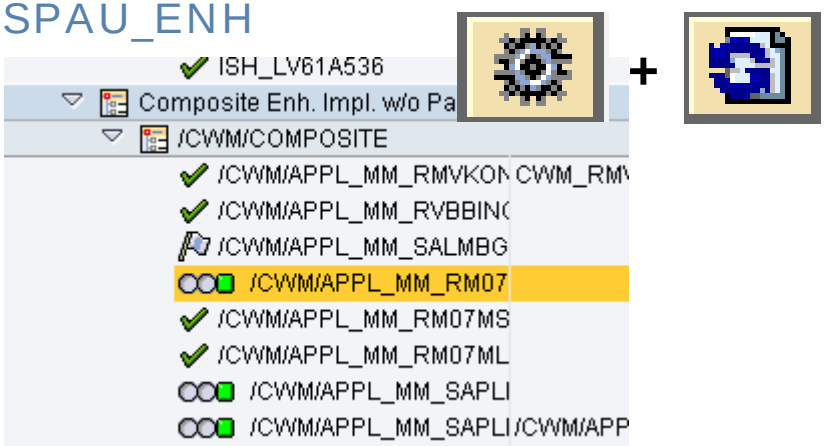
Import Log

```
Start Import LIMUREPSCD_TEST_ENH_UPGRADE2 ...
  1 conflict with enhancement objects occured (ENHOBJ PROG CD_TEST_ENH_UPGRADE2) - please call transaction SPAU_ENH
1 0
```

Editor



SPAU_ENH



Adjust Enhancements I

SPAU_ENH

1. **Double-Click on the Enhancement Implementation you want to adjust**
2. **Switch to change mode**

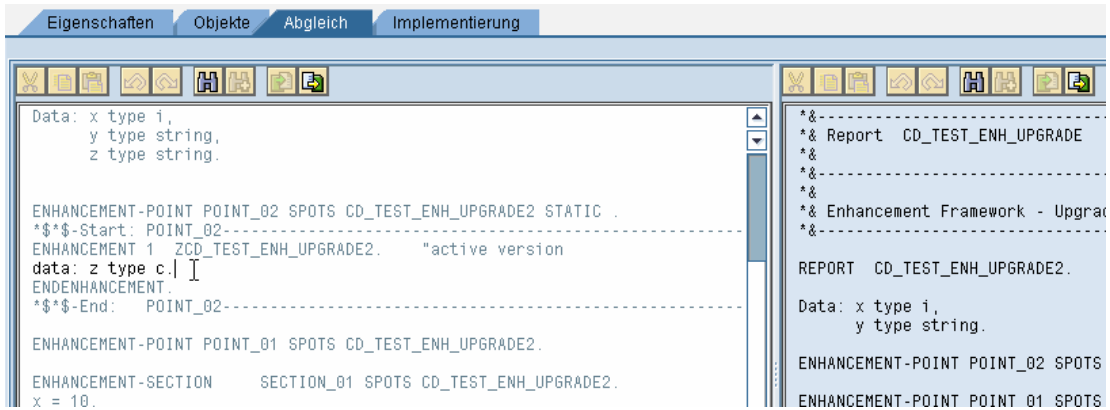
3. **Go to tab „Adjustment“**
4. **Double-Click on list entry to adjust it**

Status	Enhancement Fullname	Program name	Includename
	\PR:CD_TEST_ENH_UPGRADE2\EX:POINT_02\EI	CD_TEST_ENH_UPGRADE2	CD_TEST_ENH_UPG
	\PR:CD_TEST_ENH_UPGRADE2\EX:SECTION_01\EI	CD_TEST_ENH_UPGRADE2	CD_TEST_ENH_UPG

Adjust Enhancements II

SPAU_ENH

5. **Adjust Enhancement, e.g. in Splitscreen Editor**



6. **Set adjustment status to** 

7. **Press Button** 

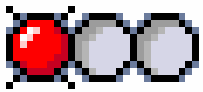
8. **Activate Enhancement** 

Adjustment States

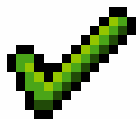
Semantic Changes

Automatic Adjustment

Tool-Aided Adjustment



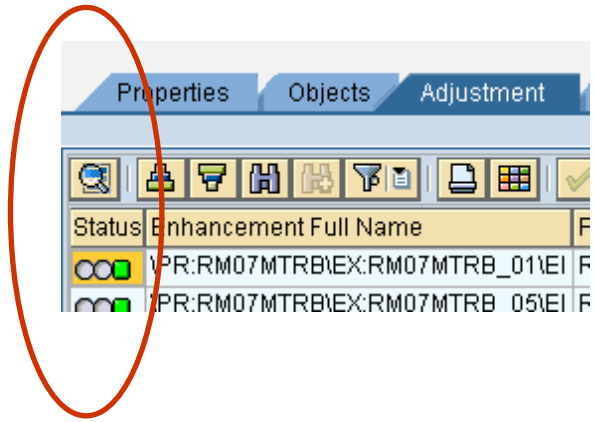
Manual Adjustment



Enhancement is adjusted

Enhanced Object was deleted

Implementation is empty



DEMO

Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

BAdI – Technology

Upgrade Adjustment

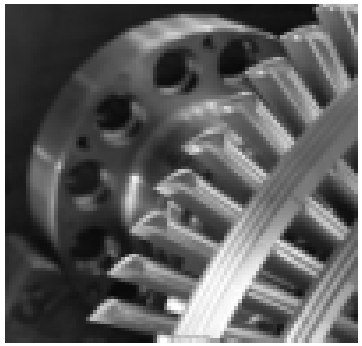
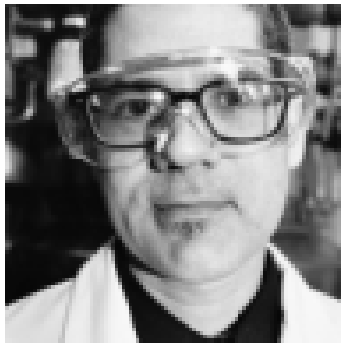
Switch Framework

Summary

Customer + SAP Status Quo: Industry Solutions

- **Shipment delay compared to core 6-12 months**
- **Latest technology stack can not be utilized**
- **Missing release synchronization leads to upgrade delay and implementation hurdles**
- **Delayed provision of legal requirements**
- **Industries are implemented as modifications**
- **No reuse of functionality of other Industry Solutions**

Goals and Benefits


Goals and Benefits

- **Reduce TCO for customer**
- **Timely provision of legal requirements**
- **Synchronization of release cycles and release planning**
- **Boost the attractiveness of ERP by providing already existing industry solutions to the entire SAP-Community**
- **Speed-up the introduction of SAP NetWeaver**

Synergies of Industry Solutions in mySAP ERP 2005

Exclusiv Activation	Multiple Activation
SAP ECC Industry Extension Healthcare 6.0	SAP ECC Enterprise Extension Consumer Products 6.0
SAP ECC Industry Extension Chemicals 6.0	SAP ECC Enterprise Extension Defense Forces & Public Security 6.0
SAP ECC Industry Extension Contract Accounting 6.0	SAP ECC Enterprise Extension Financials 6.0
SAP ECC Industry Extension Discrete Industries & Mill Products 6.0	SAP ECC Enterprise Extension FERC: Regulatory Reporting 6.0
SAP ECC Industry Extension Insurance 6.0	SAP ECC Enterprise Extension Financial Services 6.0
SAP ECC Industry Extension Media 6.0	SAP ECC Enterprise Extension Global Trade 6.0
SAP ECC Industry Extension Mining 6.0	SAP ECC Enterprise Extension Human Capital Management 6.0
SAP ECC Industry Extension Oil & Gas 6.0	SAP ECC Enterprise Extension Incentive & Commission Management 6.0
SAP ECC Industry Extension Public Services 6.0	SAP ECC Enterprise Extension Industry-specific Sales Enhancements 6.0
SAP ECC Industry Extension Retail 6.0	SAP ECC Enterprise Extension Joint Venture Accounting 6.0
SAP ECC Industry Extension Telecommunications 6.0	SAP ECC Enterprise Extension PLM 6.0
SAP ECC Industry Extension Utilities, Waste & Recycling 6.0	SAP ECC Enterprise Extension Public Sector Management 6.0
SAP ECC Industry Extension Catch Weight Management 6.0	SAP ECC Enterprise Extension Retail 6.0
	SAP ECC Enterprise Extension SCM 6.0
	SAP ECC Enterprise Extension Travel Management 6.0

Single Activation

Multiple Usage

Benefits

Switches control the visibility of repository objects at runtime

Benefits:

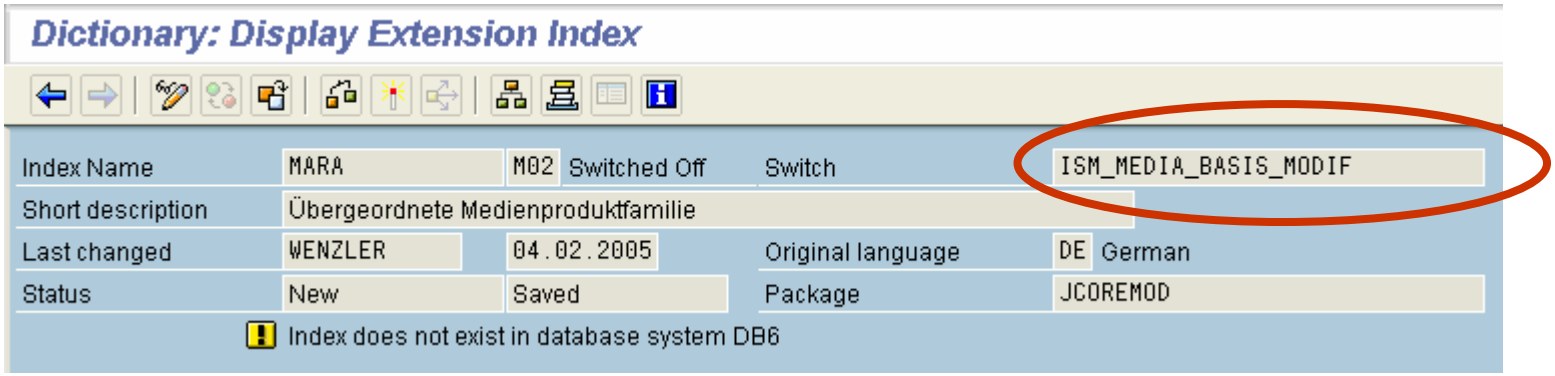
- **Industry Solutions are available with every release and SP without delay (i.e. timely provision of legal requirements), CRT's are no longer necessary for add-on systems**
- **Industry Solutions can be enriched by generic functions from other industries**
- **Synchronization of release cycles and planning**

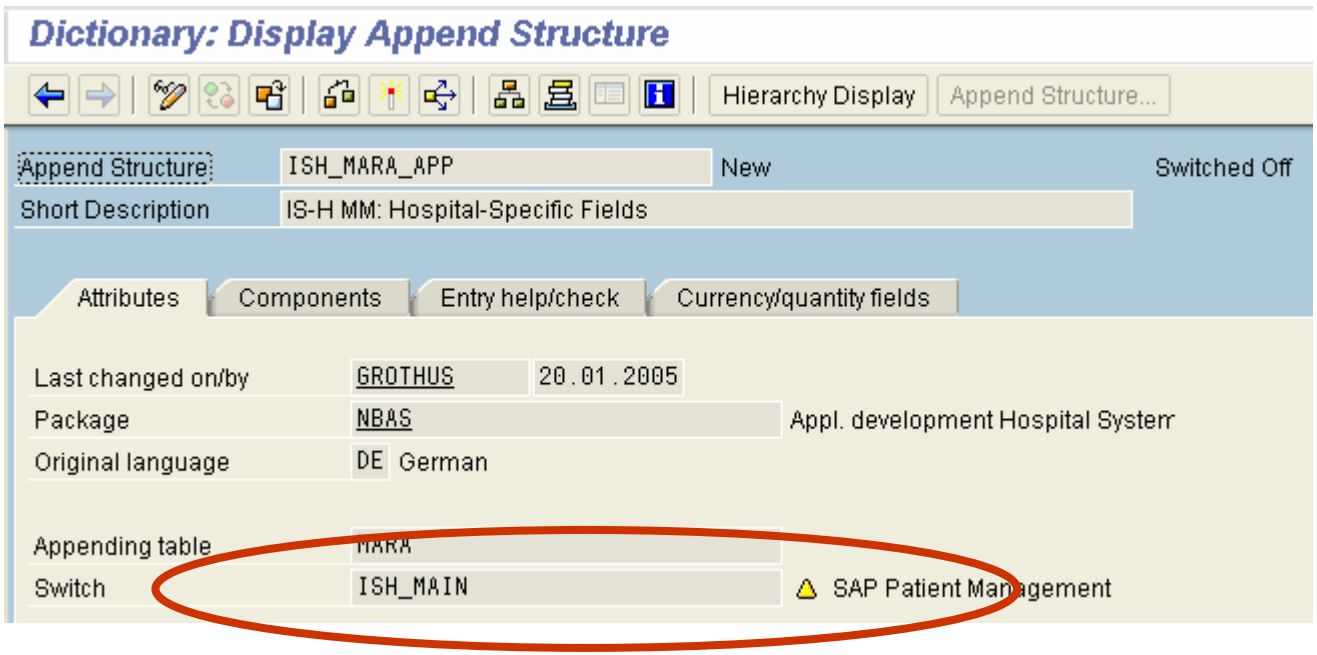
Switchable Objects...

...by package assignment

- **Appends, SI-, CI-includes for dictionary structures**
- **Fixed value appends to domains**
- **Secondary Indexes**
- **Append Search Helps**
- **Enhancement Implementations**
- **Switch Business Configuration Sets (Switch BC-Sets)**

How to Determine if an Object is Switched?





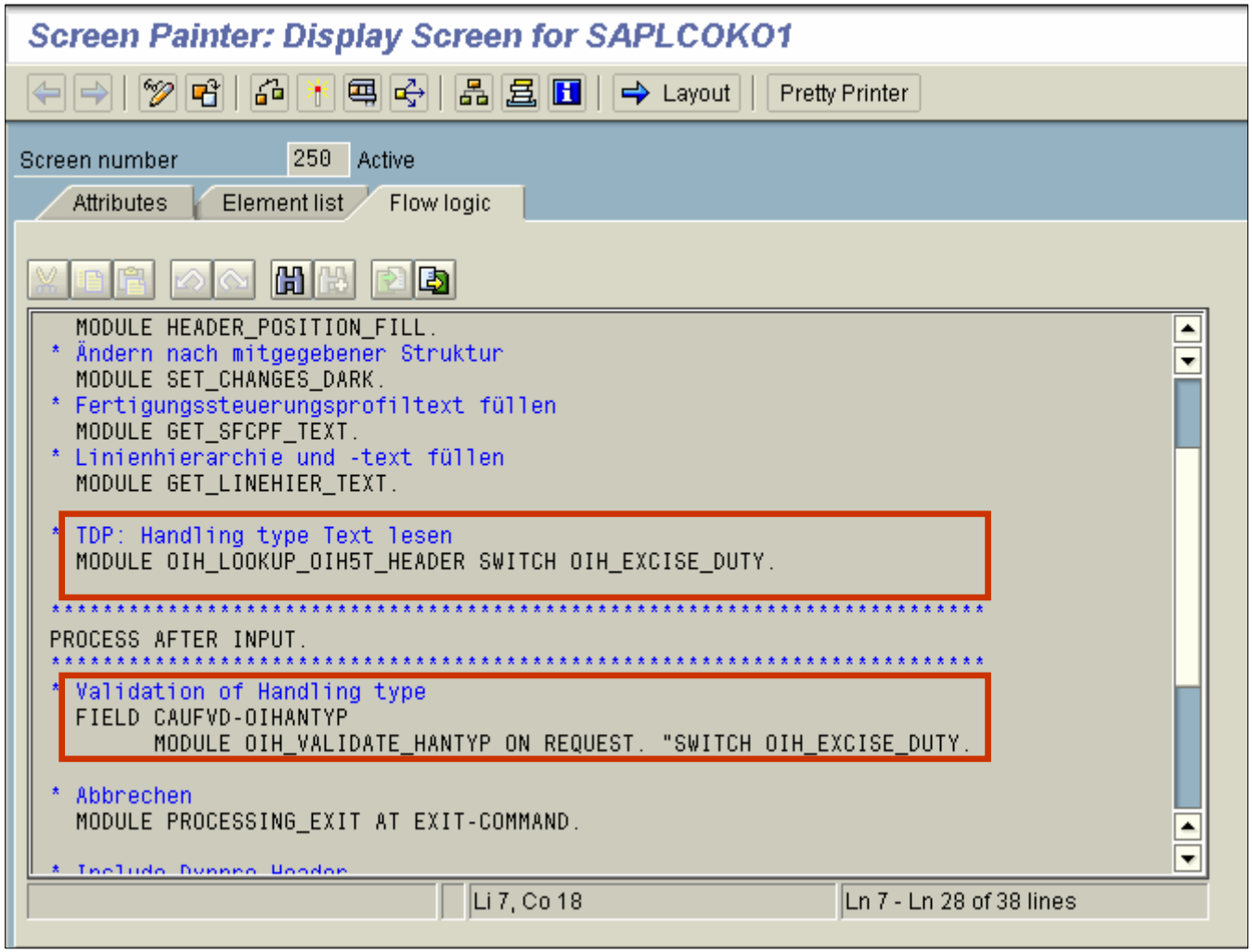
Switchable Objects II

Switchable Objects ...

... by direct assignment

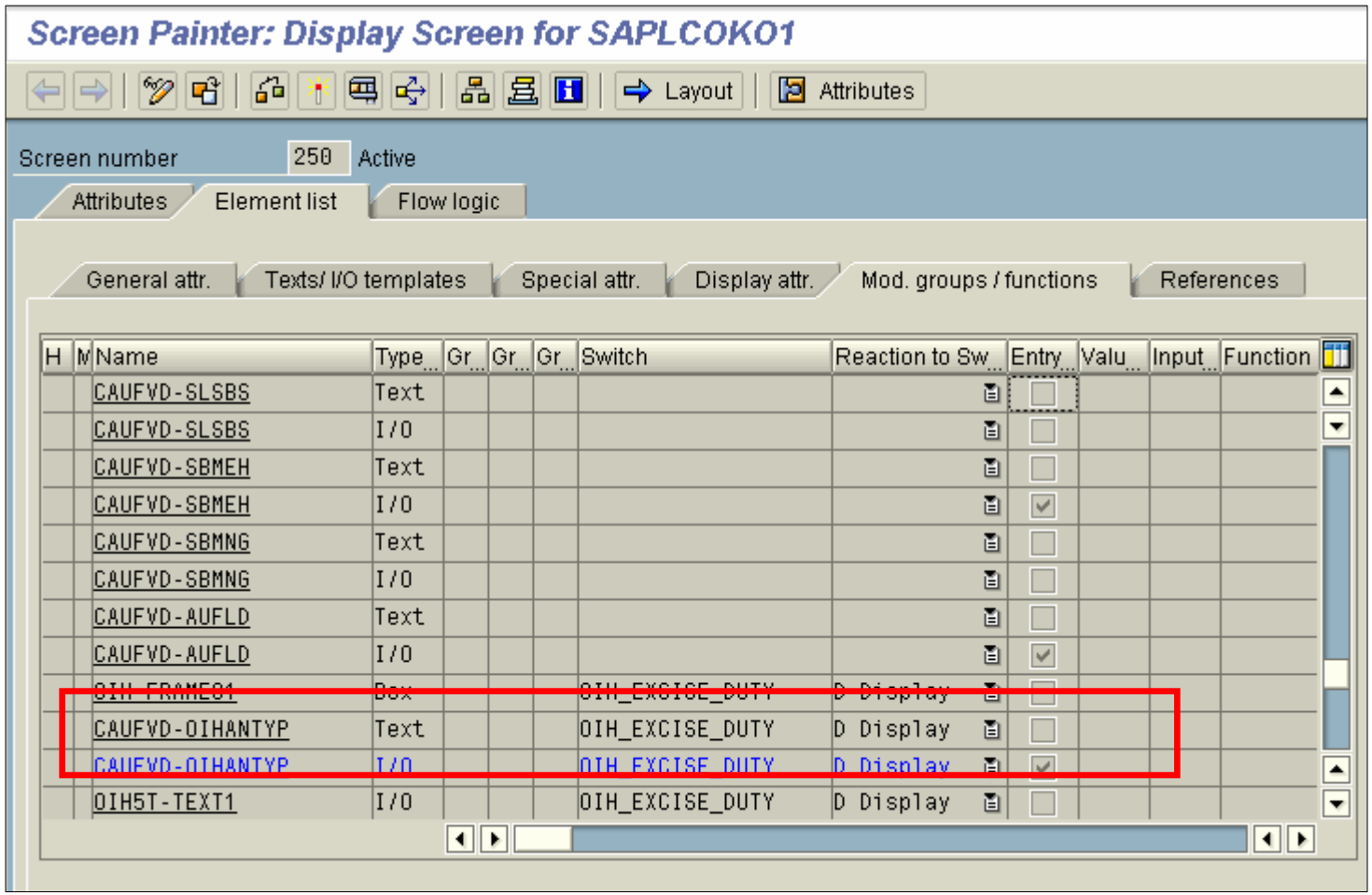
- **Screen elements & Flow logic**
- **Menu entries & functions**
- **IMG nodes**
- **Customizing**

Switch dependent module

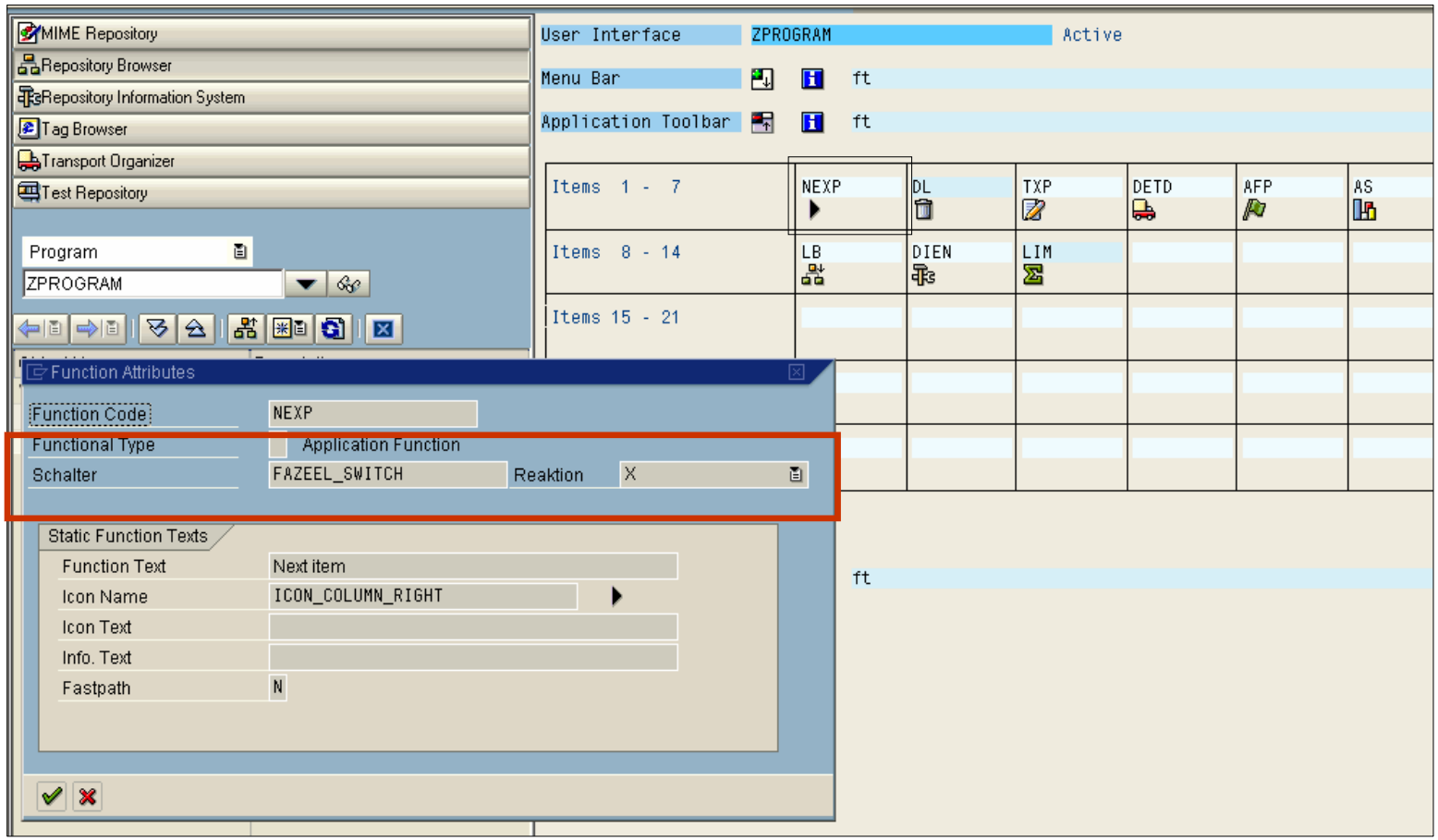


Screen Painter – Element List

Switch dependent screen elements in the element list – disabled elements are not visible at runtime.

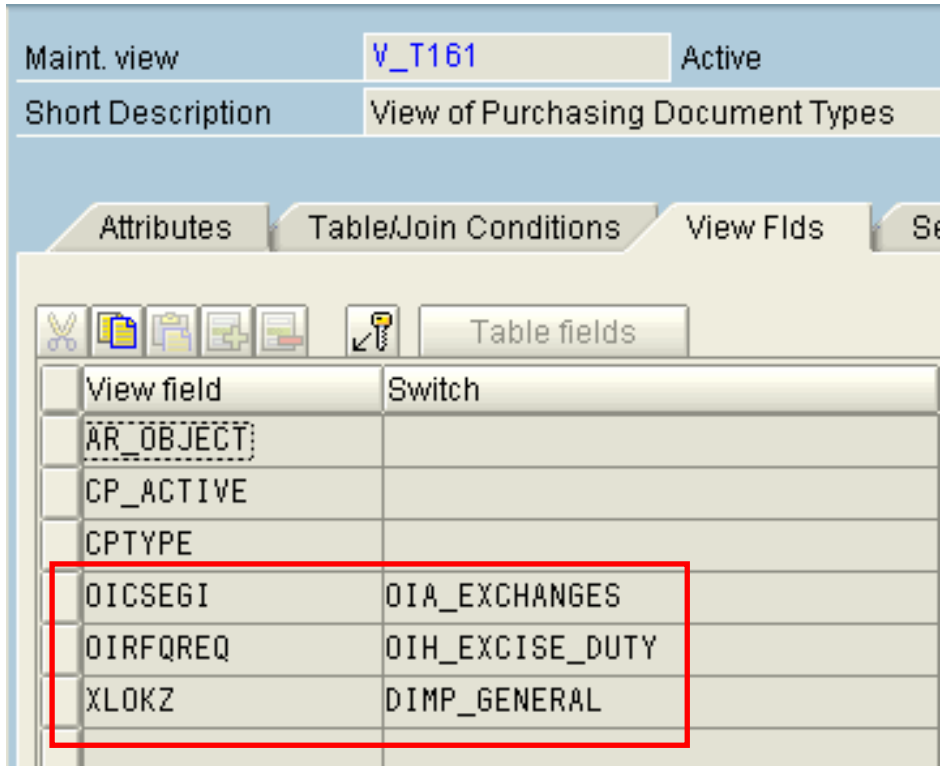


Switch dependent menu entries



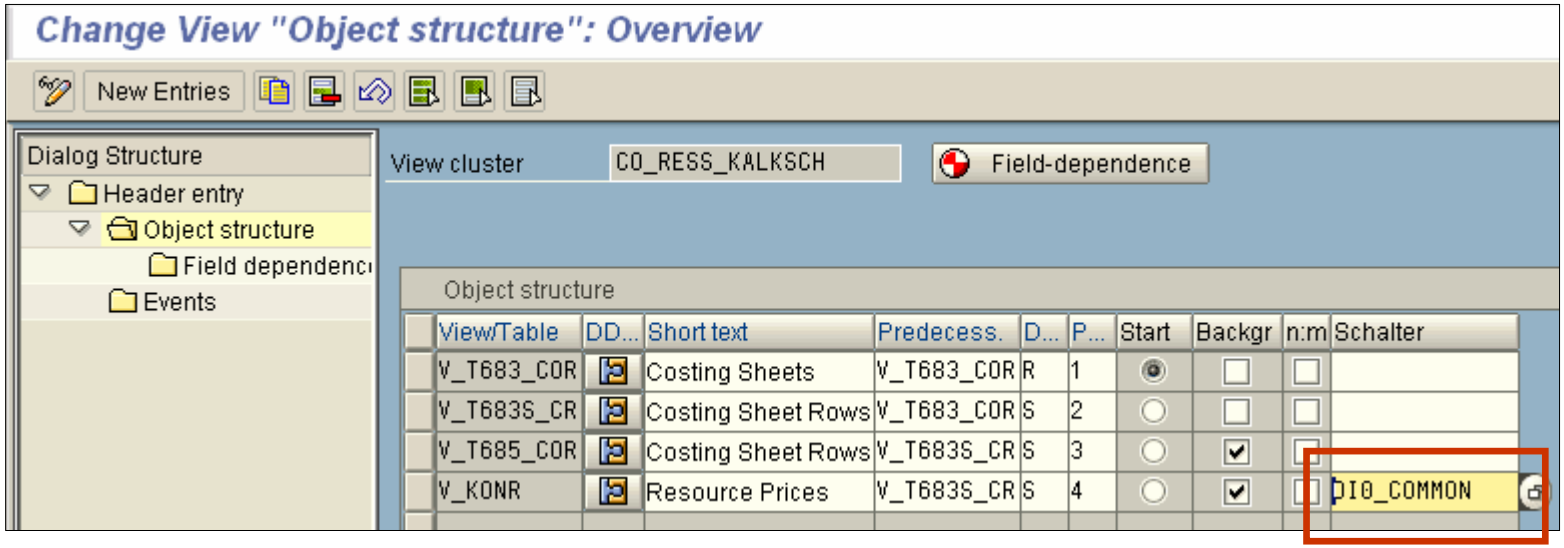
Maintenance Views

The visibility of the fields of a table control is determined by the switch state.



View Cluster

The visibility of a view within a view cluster can be controlled by switches.



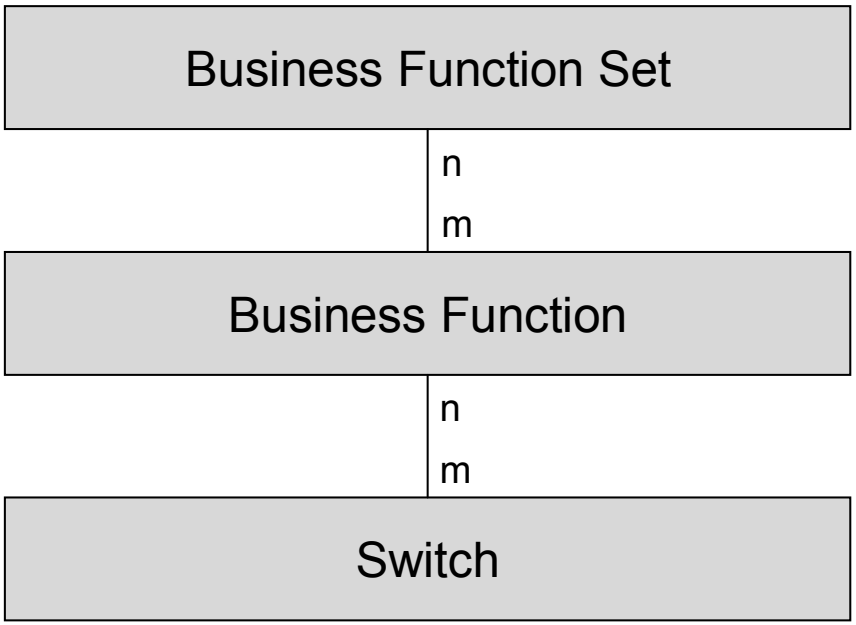
Switch dependent import of table content via BC-Sets

- **BC-Sets improvements:**
  - **Performance improvements**
  - **Support of deletion of customizing entries**
  - **Tool improvements like automatic recording, switch dependent definition and browsing**

Switch dependent IMG nodes

- **IMG nodes of disabled industries are not visible**

Business Function Set



Business Function Set

- **Pool of business functions**
- **Represents one industry solution**
- **Max. 1 can be active**
- **Use Transaction SFW5 to switch a on BFS.**
- **Transaction SFW3**

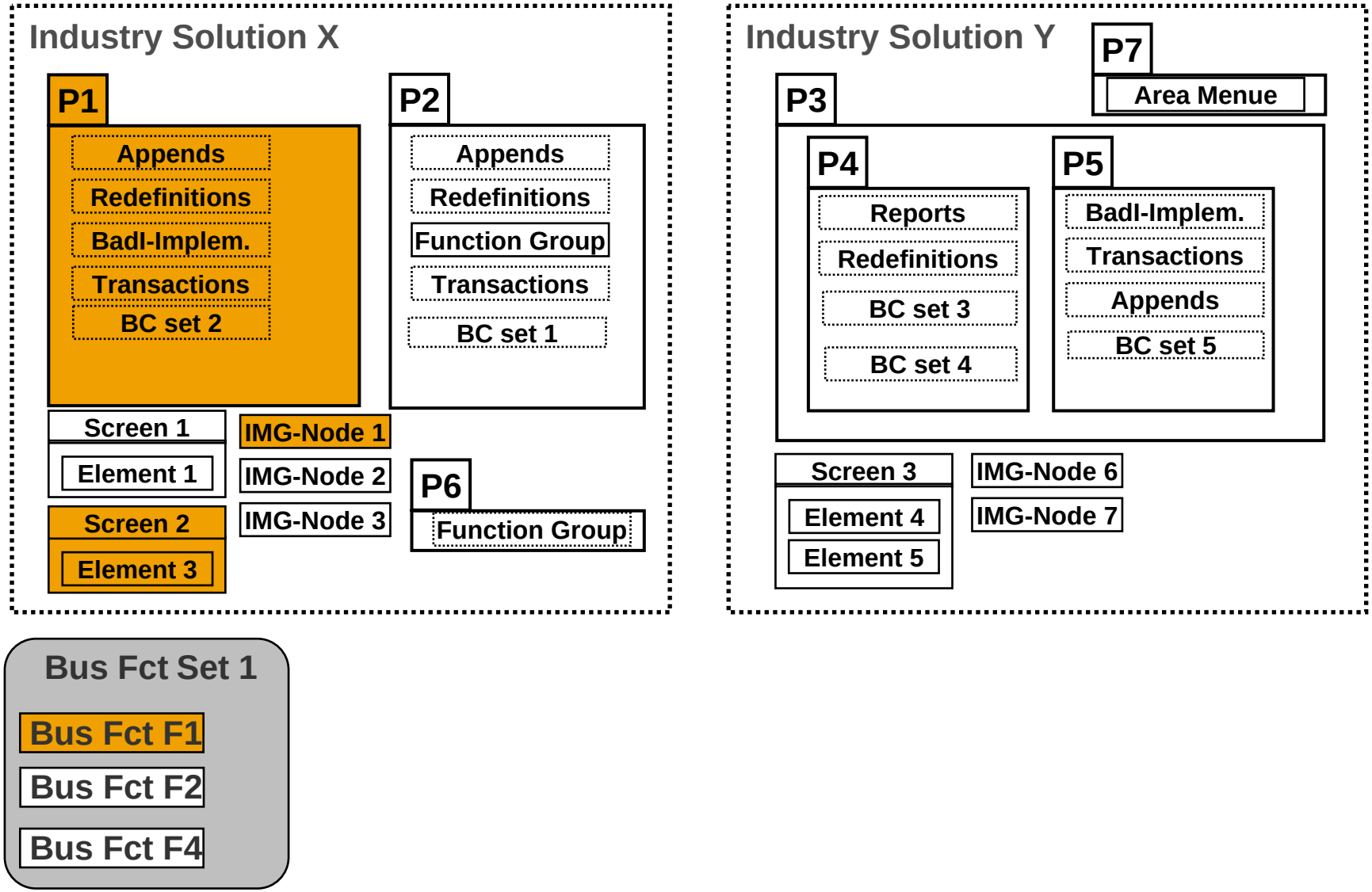
Business Function

- **Represents a piece of business functionality**
- **Contains switches**
- **Transaction SFW2**

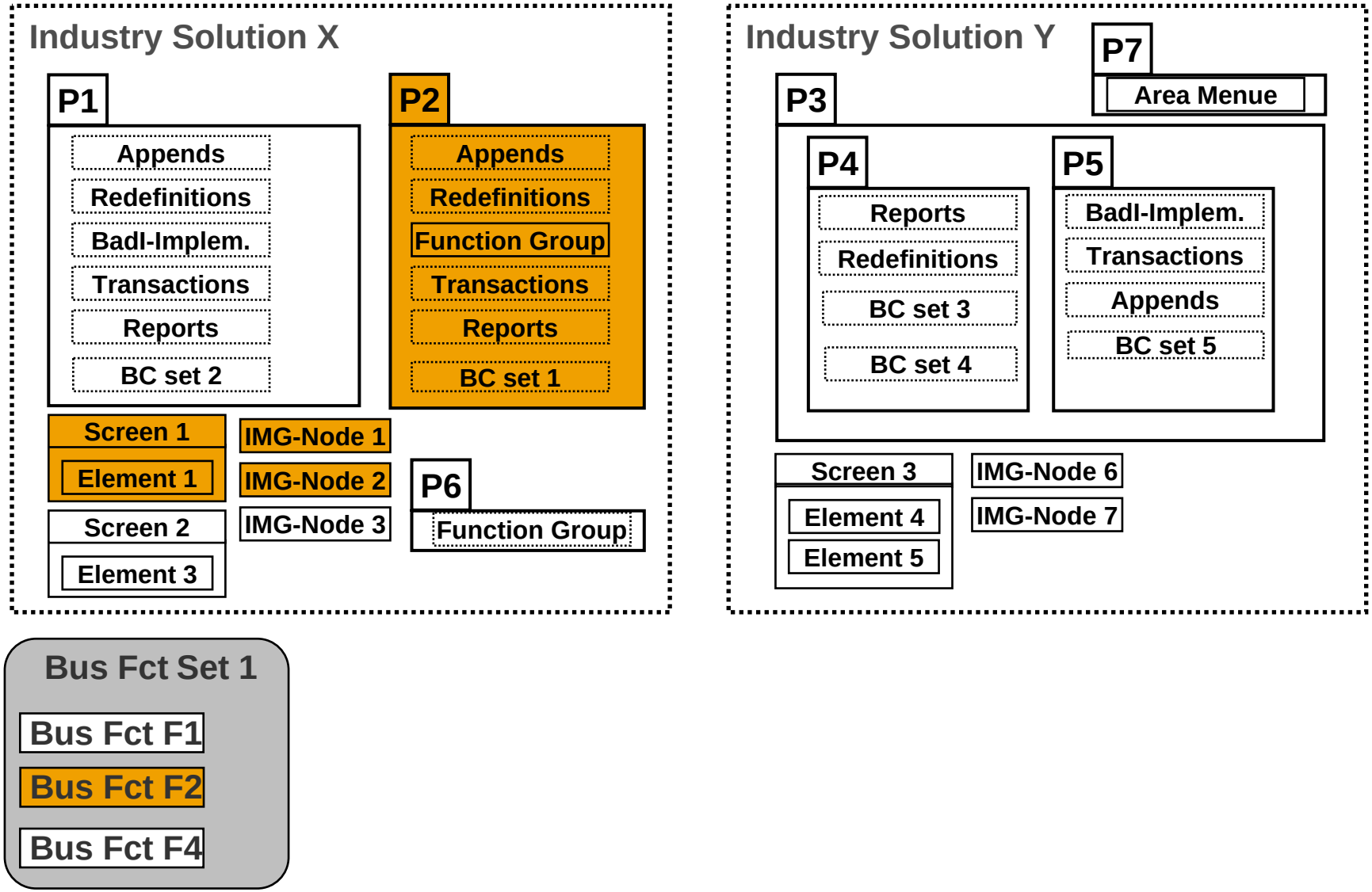
Switch

- **Calculated states: ON, OFF, STANDBY**
- **Transaction SFW1**

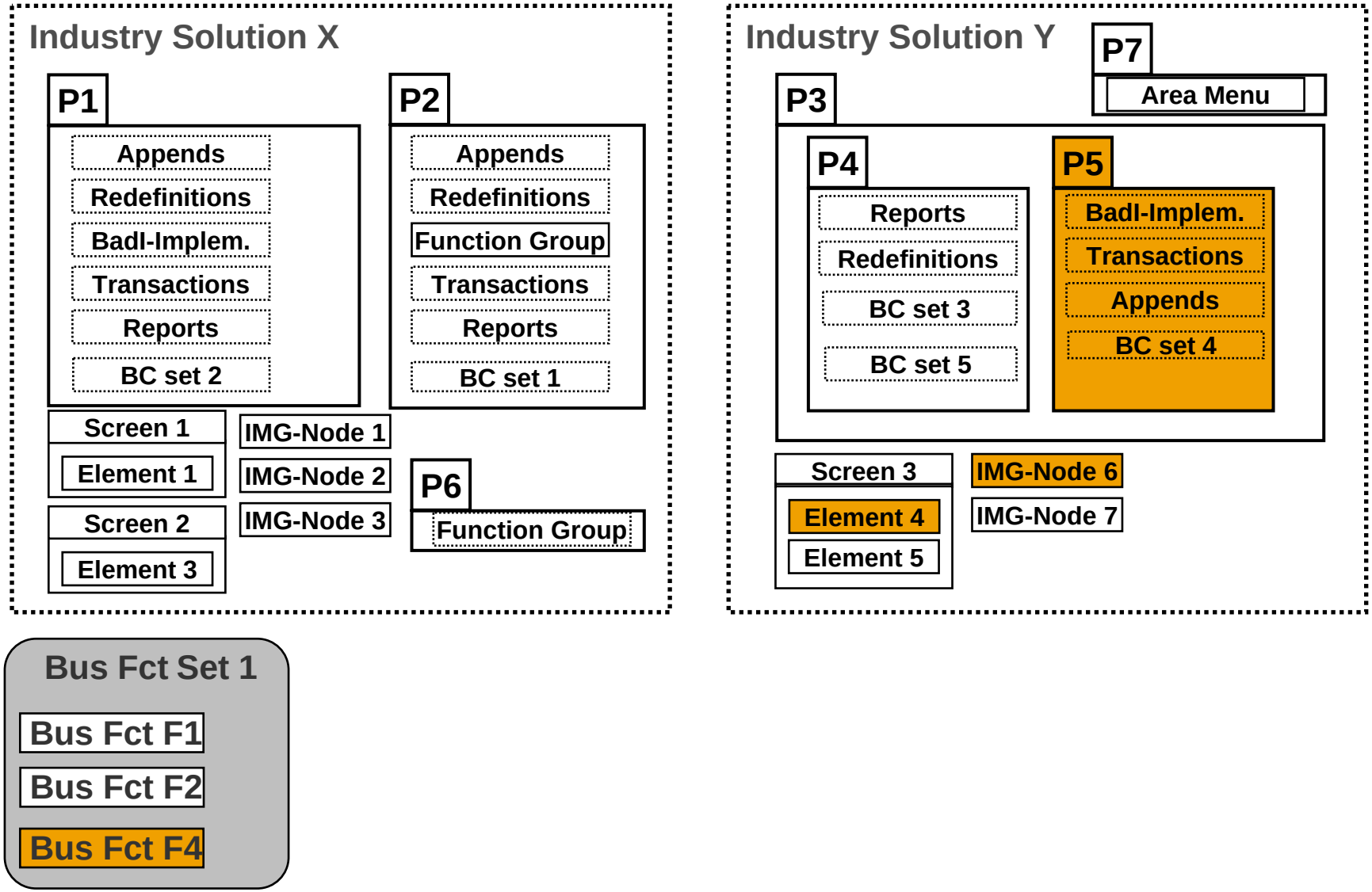
Mapping Switches and Architecture I



Mapping Switches and Architecture II



Mapping Switches and Architecture III



Activate a BFS

Log on to the Development System with authorization SAP_ALL

Disable all batch jobs

Close the system for all other users

Transaction SFW5

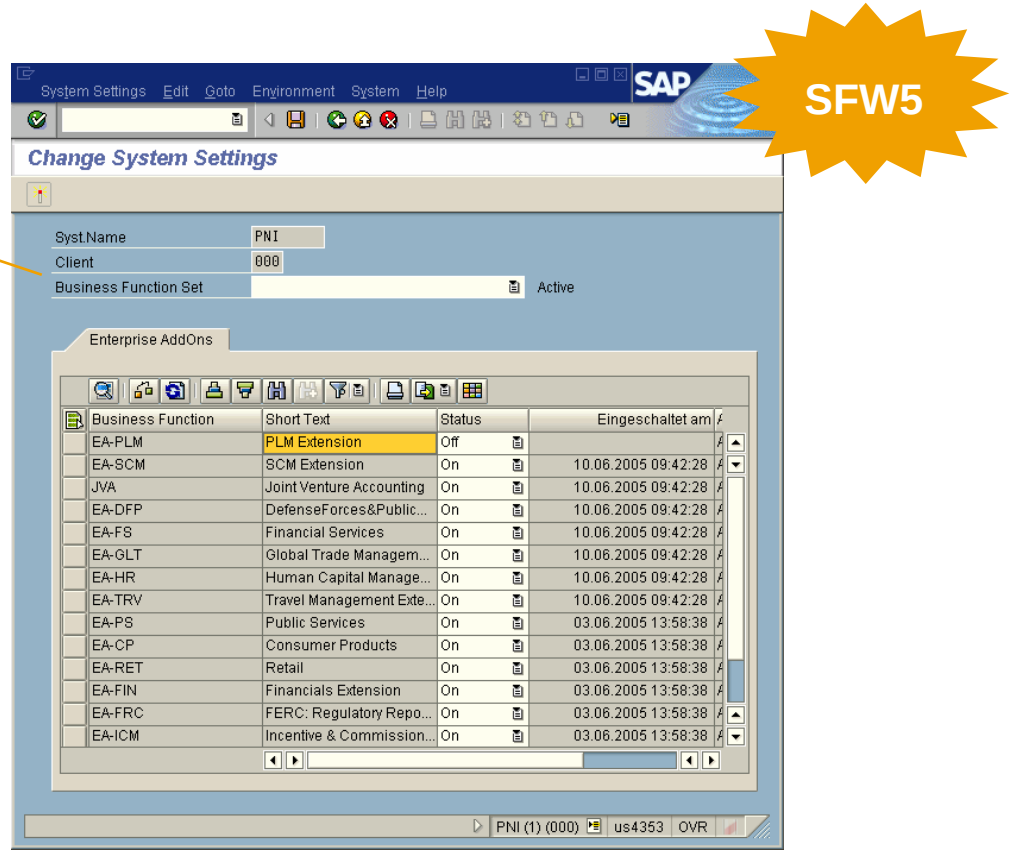
Choose the BFS in the drop down box

Select the required Business Functions that belong to the chosen BFS

Click on the button “activate”

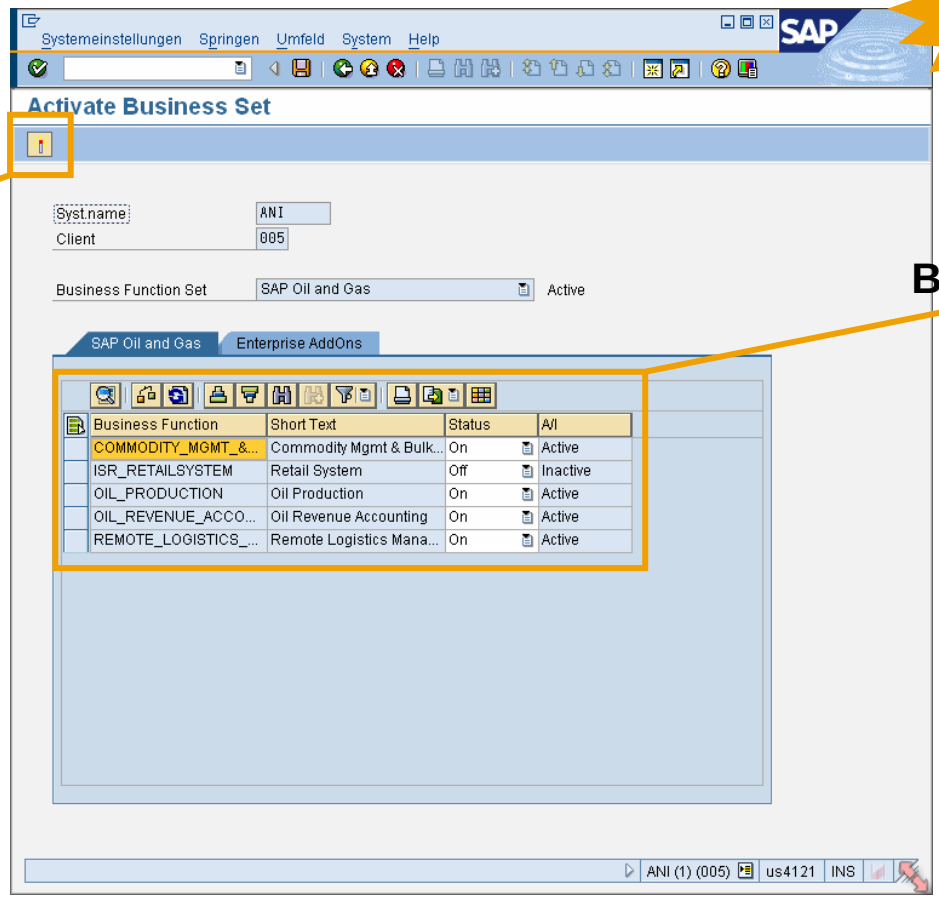
Choosing the Business Function Set (BFS)

I. Select the BFS



Activating the Business Function Set

SFW5



III. Press activate

II. Choose the Business Functions

DEMO

Activating a BFS II

- **The Batch Job for DDIC and BC-Set activation starts immediately.**
- **This job takes about 30-120 min**
- **The background activities are comparable to installing an Add On manually**
- **Reschedule batch jobs in the development system and unlock users**

Caution:
Only one Business Function Set can be activated

Caution:
There is no possibility, to deactivate a BFS

Enhancements for the objects of the Switch Framework

- **Switches**
  Packages can be added by enhancements. Beneficial for distributed development of switchable solutions. This enhancement is not switchable itself. When switching on a switch, all enhancements of this switch will be considered.

- **Business Functions**
  Switches can be added by enhancements.

- **Business Function Sets**
  Business Functions can be added by enhancements. Thus it is possible for ISVs to add their functionality to SAP Business Function Sets without modifying the original object.

- **Switching off Business Functions**
  For test purposes only

Enhancement Framework Overview

Source Code Plugin – Technology

Function Group Enhancement – Technology

Class Enhancement – Technology

BAdI – Technology

Upgrade Adjustment

Switch Framework

Summary

Summary

- **The Enhancement Framework offers new possibilities to extend the SAP Standard instead of modifying it.**
  - **Source Code PlugIns**
  - **Function Group Enhancements**
  - **Class Enhancements**
  - **New BAdIs**
- **The new BAdIs are more flexible and faster than the classic ones.**
- **The Enhancements offered by Enhancement Framework and several other object types can be switched by the Switch Framework.**

Further Information

Public Web

http://help.sap.com ➔ Documentation ➔ SAP Netweaver ➔ Application Platform ➔ ABAP technology ➔ ABAP Workbench ➔ Enhancement Framework

http://sdn.sap.com ➔ Weblogs by author Thomas Weiss

THANK YOU FOR YOUR ATTENTION !

QUESTIONS – SUGGESTIONS – DISCUSSION

Copyright 2006 SAP AG. All Rights Reserved