UV 911.411
Enterprise Computing
WS2025/2026

SW-Development with SAP-ABAP

Dr. Schwaiger Roland

NoR GmbH

Introduction

roland.schwaiger@nor.gmbh

The book to read (German)

Dr. Roland Schwaiger

Located
- Bad Dürrnberg, Hallein, AT

Background
- Mathematics (University Salzburg)
- Computer Sciences (University Salzburg, Bowling Green State University)
- Project & Process Management (SMBS – University of Salzburg Business School)

Profession
- Co-Founder and -CEO **NoR GmbH**
- SAP Technical Consultant (Cert. SAP Development Consultant, SAP Cloud Developer)
- SAP Trainer
- Project Coach (Cert. Scrum Master)
- Software Architect
- Software Developer (SAP AG, Walldorf, DE and Customer Development Projects)
- Author (check out Amazon and/or www.citeseer.com)
- Lecturer (University Salzburg, FH Salzburg)

Passion
- Learning

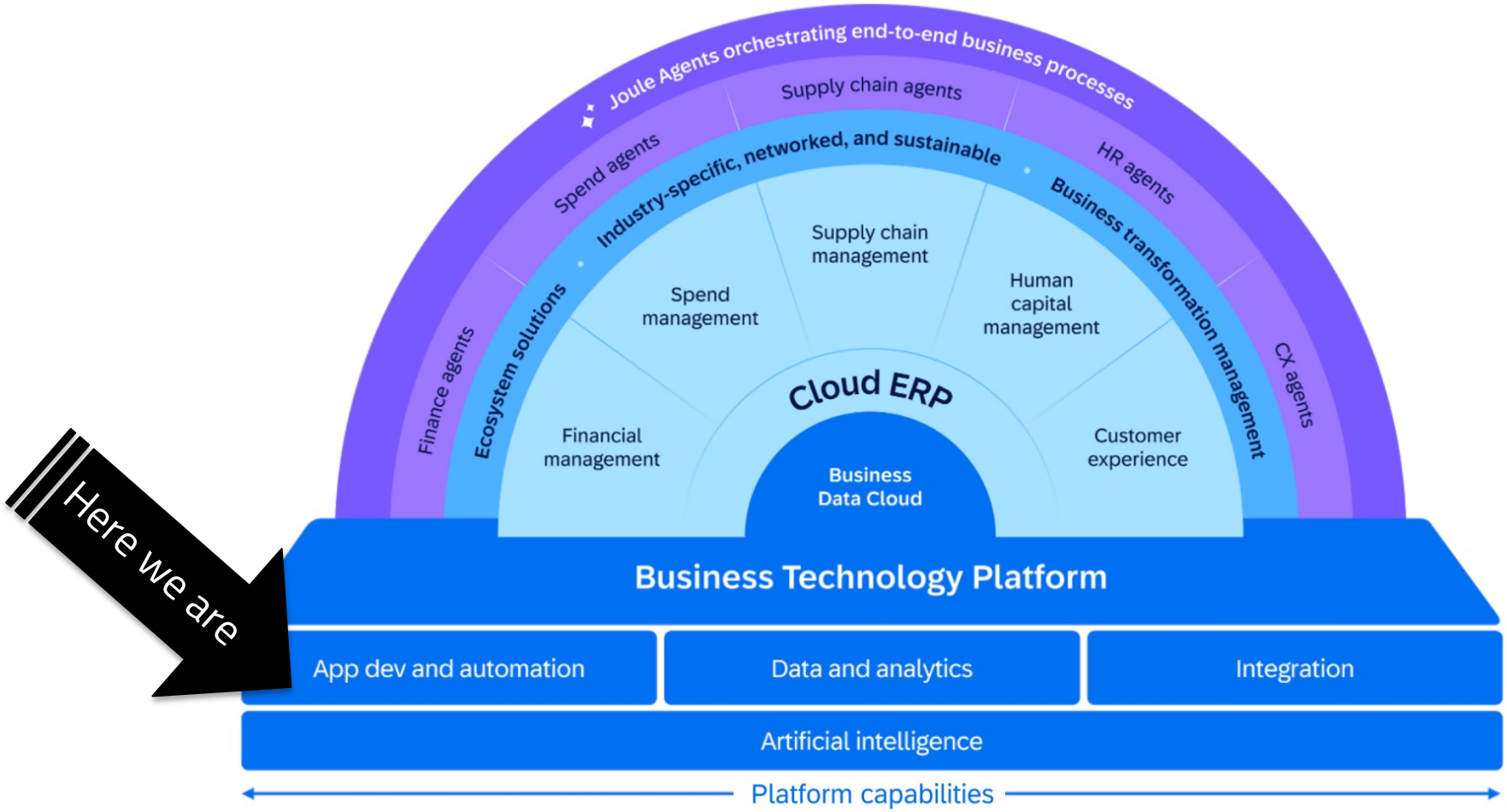
Definition

" **Enterprise computing** involves the development, deployment and maintenance of the information systems required for survival and success in today's business climate."

Thus SAP is one instance of the class of enterprise computing

Yen-Ping Shan & Ralph H. Earle,
Enterprise Computing with Objects ,
Addison-Wesley, 1998.

Motivation


Joule Agents orchestrating end-to-end business processes
Supply chain agents
Spend agents
HR agents
Industry-specific, networked, and sustainable
Business transformation management
Finance agents
Ecosystem solutions
CX agents
Supply chain management
Spend management
Human capital management
Cloud ERP
Financial management
Customer experience
Business Data Cloud
Here we are
Business Technology Platform
App dev and automation
Data and analytics
Integration
Artificial intelligence
Platform capabilities

Course Overview

Infrastructure BTP, Cockpit, Subaccount, Entitlements, Cloud Foundry, ABAP Environment

ABAP Cloud Program Package, OO Console Application

Dev Tools Eclipse ADT, Git, BAS, Class Builder, ABAP Editor, Dictionary

Data Types and Data Object DATA, TYPES, Tables, Structures, References

ABAP OO Class, Object, Method, Attribute, Exception

RAP Development CDS, Behavior, Service, Binding, UI Annotation

Database Create Tables, Read, Update, Delete

Finalizing the Course

The completion of the course takes place within the course block. This consists of:

- Implementation of repository objects as part of the course
- Final examination
  - Theoretical exam with ten questions covering the course content
  - Development of repository objects

Motivation

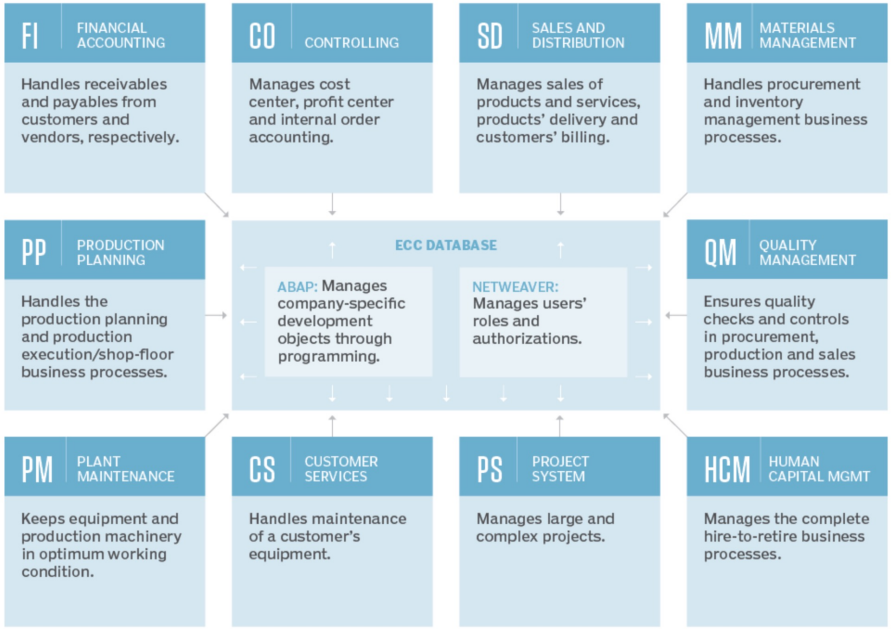
Motivation
What does SAP do?

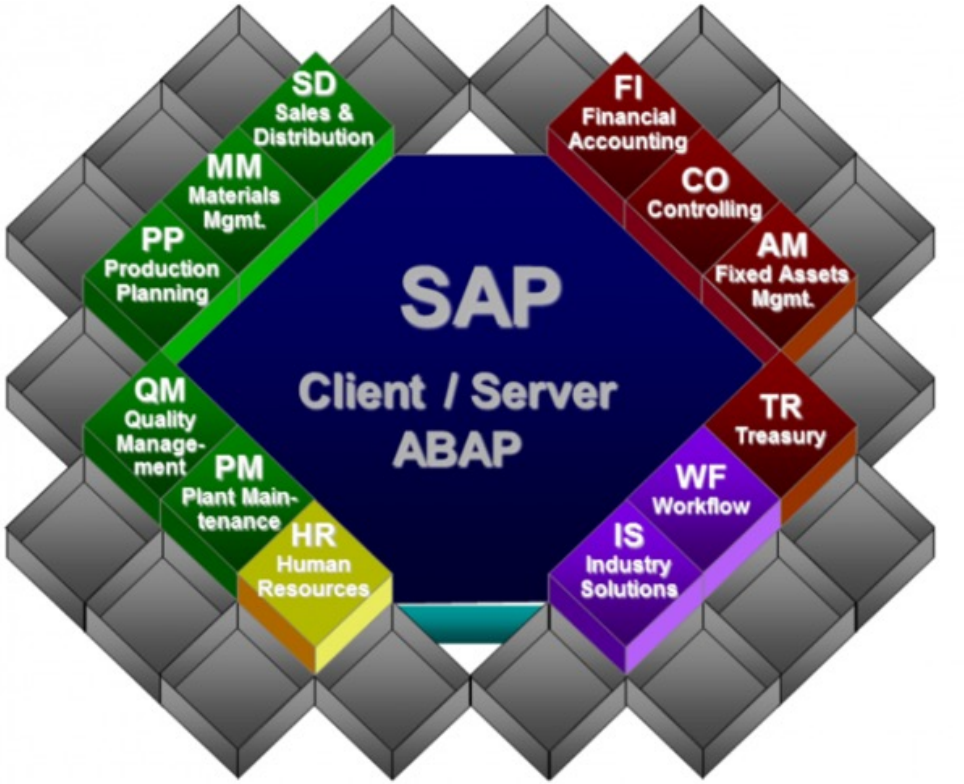
SAP helps companies and organizations of all sizes and industries run their businesses profitably, adapt continuously, and grow sustainably [Definition by SAP].

Motivation
SAP ECC

SAP ERP Central Component is the umbrella term for SAP's functional and technical software components that make up the ERP system.
ECC is also known as SAP ERP.

https://erproof.com/how-does-sap-work/



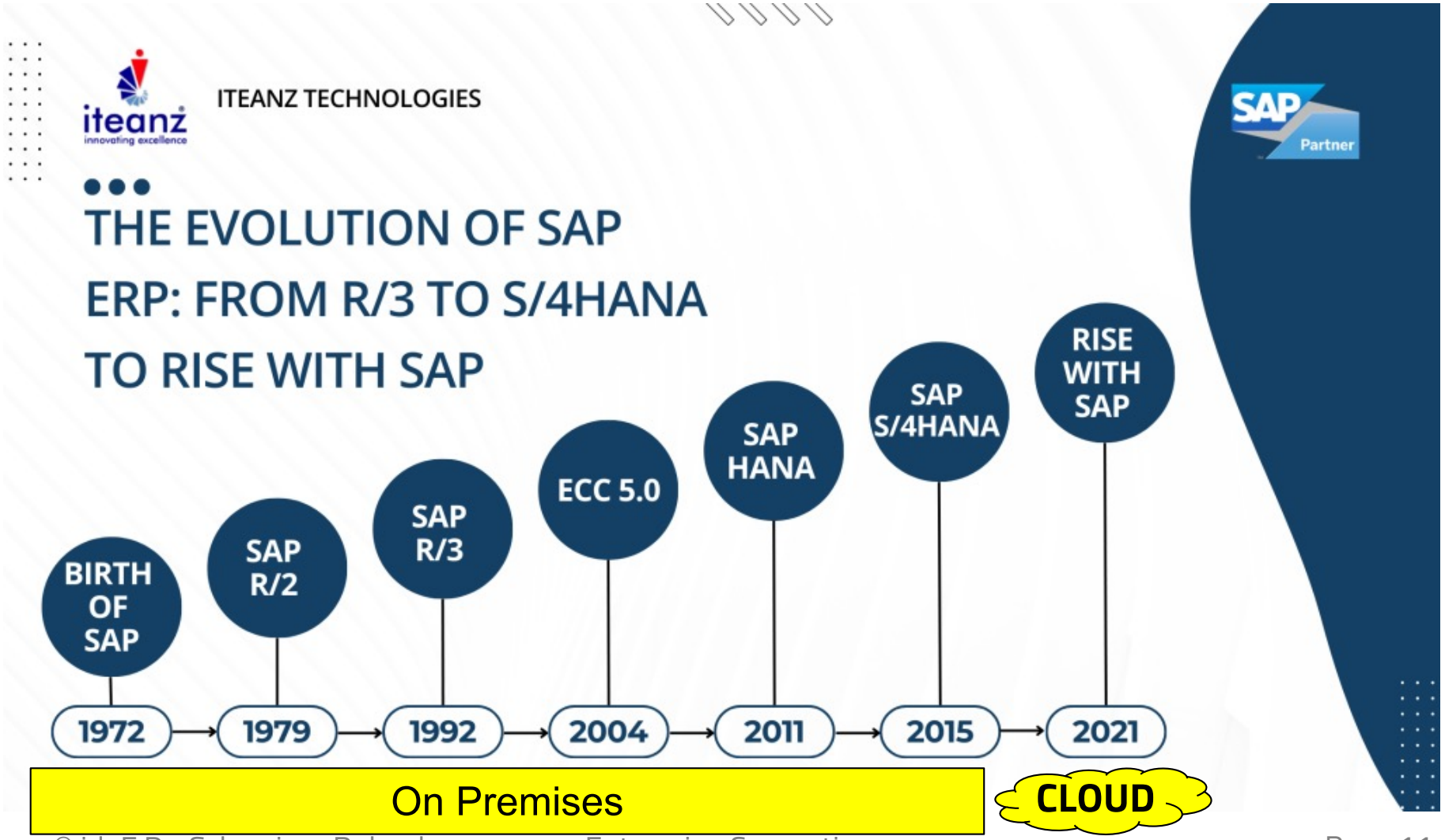


https://www.techtarget.com/searchsap/definition/SAP

Motivation
Who is using SAP? e.g.

https://www.thomsondata.com/customer-base/sap.php

Evolution of SAP



Evolution of SAP

- The first version of SAP's **enterprise software** was a financial accounting system named **RF** or better known as R/1 . (The "R" was for "Real-time data processing ")
- This was replaced by R/2 at the end of the 1970s. **SAP R/2** was a mainframe based business application software suite that was very successful in the 1980s and early 1990s
- The client server based solution SAP R/3 was official launched on July 6 , 1992. (The 3 stands for the three layer architecture)
- In 2008 SAP HANA, an in-memory, column-oriented RDBMS was introduced .

Motivation
SAP

SAP history

- “SAP system analysis and program development” was founded in 1972 by 5 former IBM engineers
- In 1976 the SAP GmbH was founded, they moved to Walldorf
- In 1988 the SAP Aktiengesellschaft *Systems, Applications and Products in Data Processing* was founded

Motivation
SAP History

Metric	Updated SAP Figures (approx.)
Global ranking	Largest software company in Europe; among top globally
Employees (total)	~109,973 (2024)
Revenue (IFRS)	~€34.18 bn (2024), ~€36.8 bn (2025 est.)
Customers	~425,000 – 440,000 worldwide
Countries/Subsidiaries	Operations in 180+ countries
Development Centers	~19 major SAP Labs centers globally

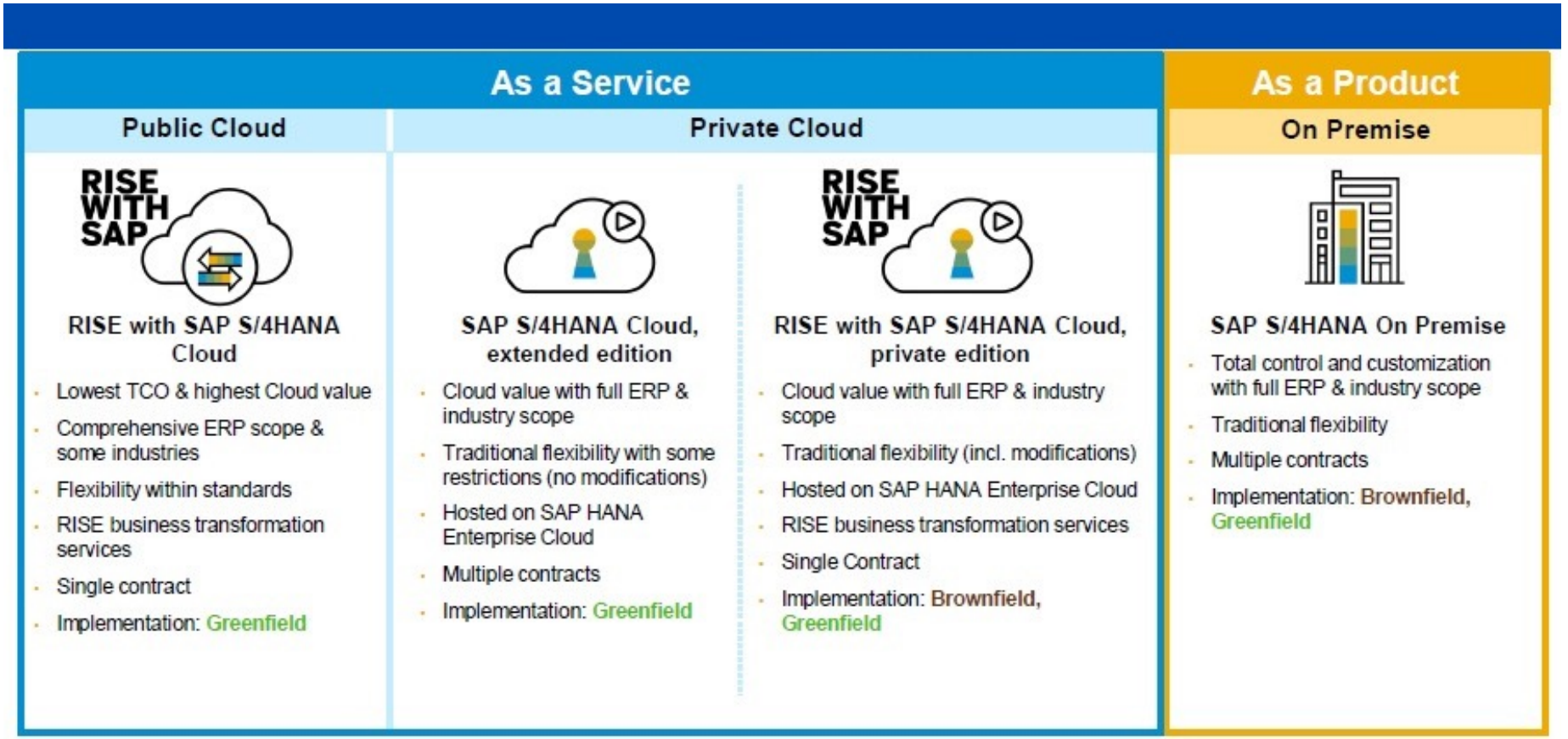
Motivation
SAP

SAP is...

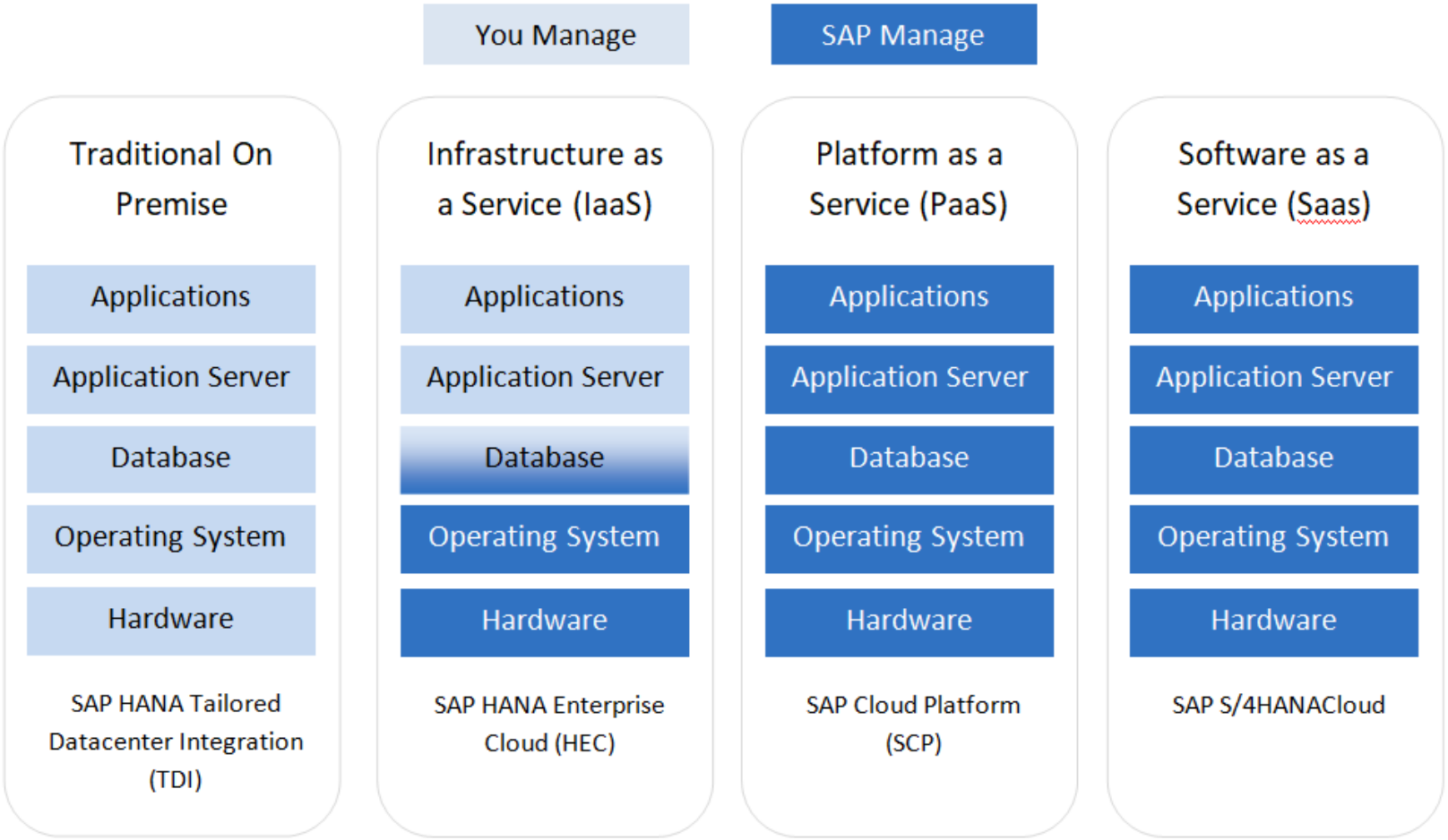
- the corporation SAP
- the family of products of the SAP, referred to as **THE SAP SYSTEMS**

Terms

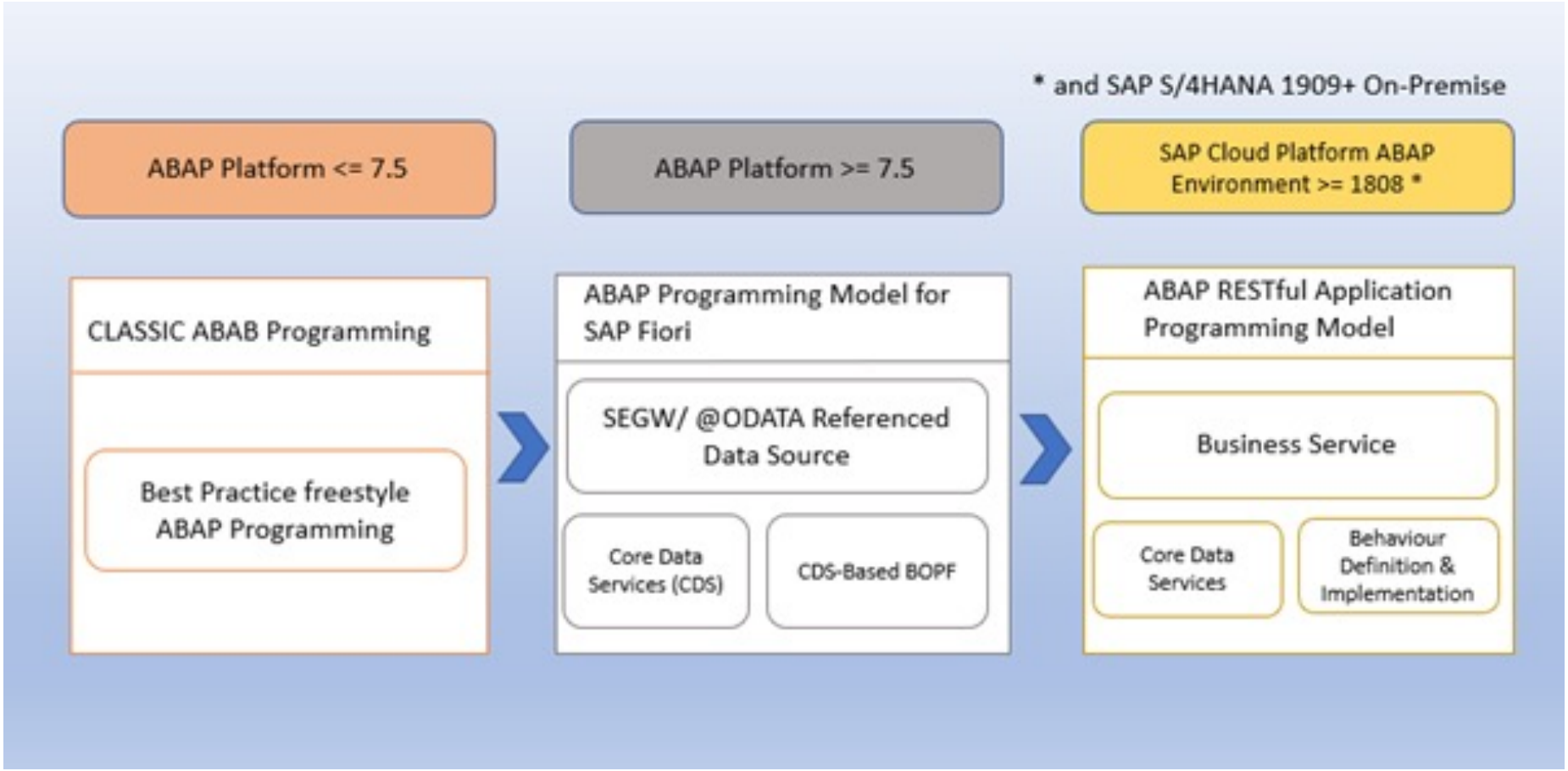
Talking about Cloud



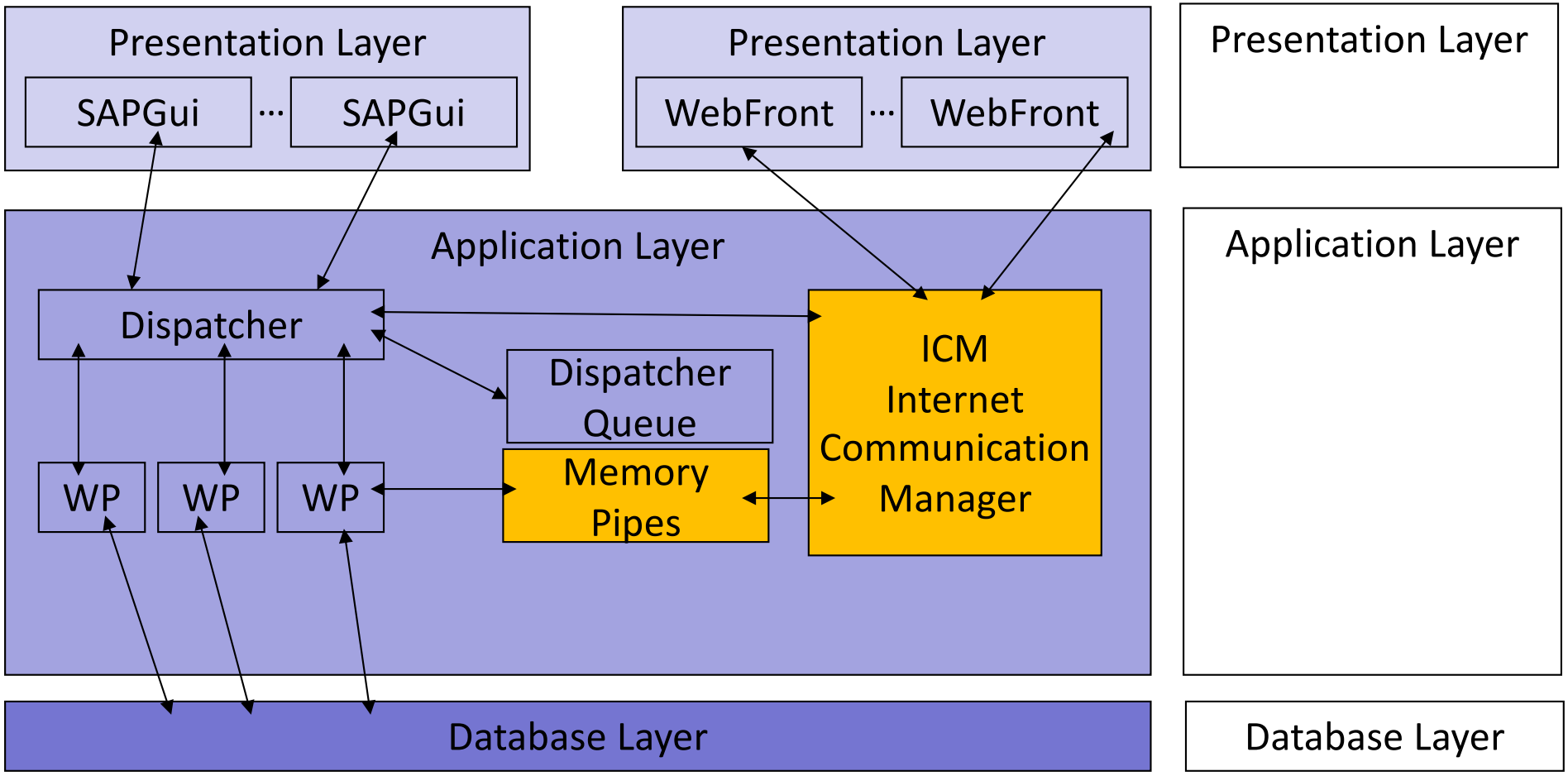
Talking about Responsibility



Development Model Evolution

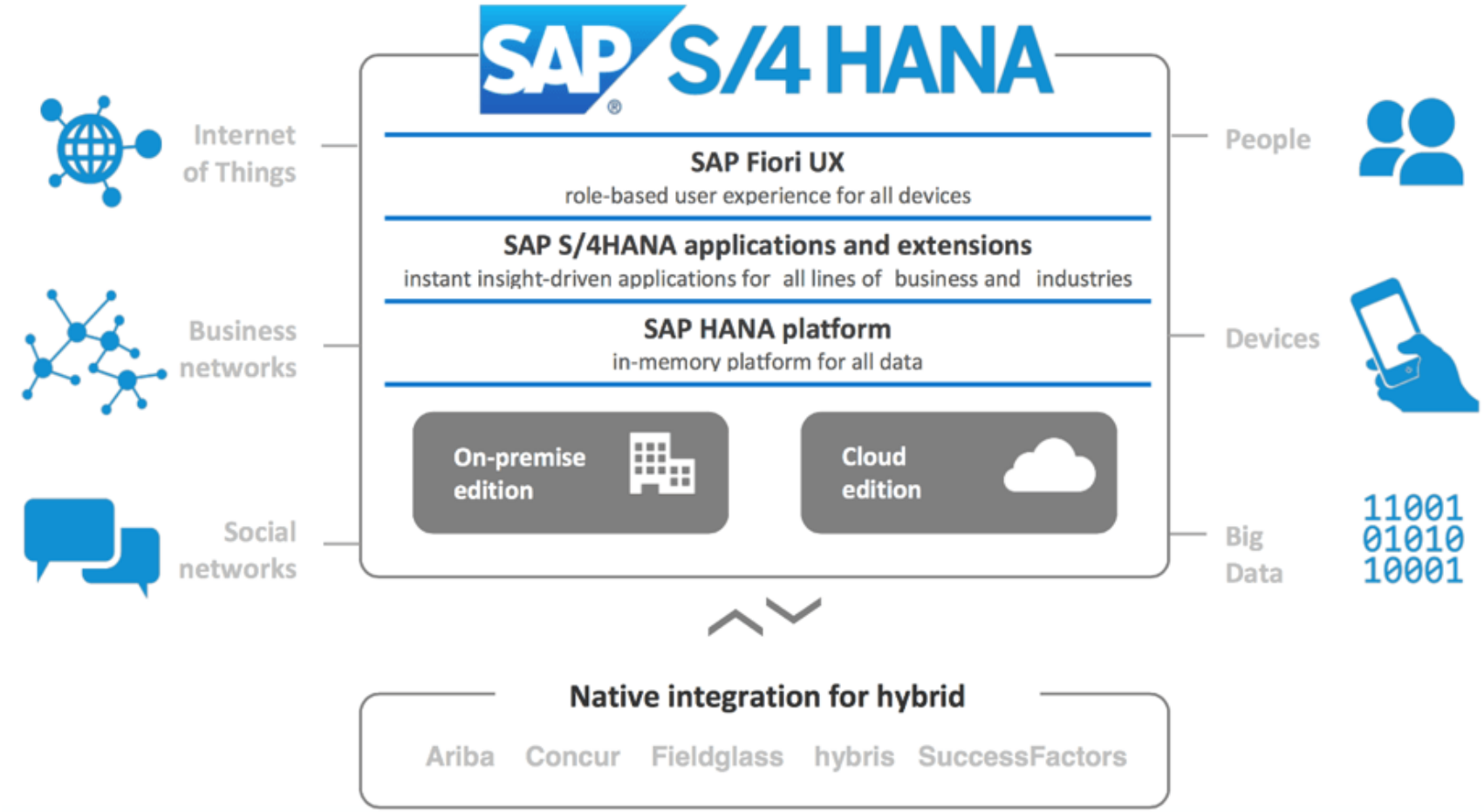


Terms
Application Platform ABAP onPremises



Terms
SAP System Examples

NoR
EMPOWER THE EVOLUTION OF NOW



© SAP

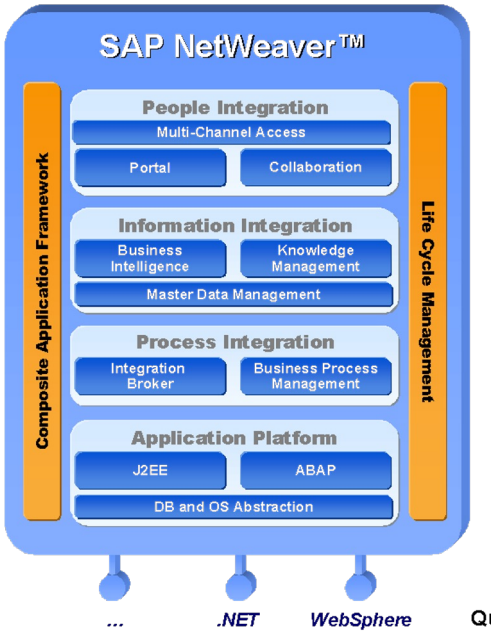
Terms
SAP integration options

OnPremises

Classic
- Java Connector (Jco)
- Flash Islands (Flex)
- Silverlight Islands (MS)

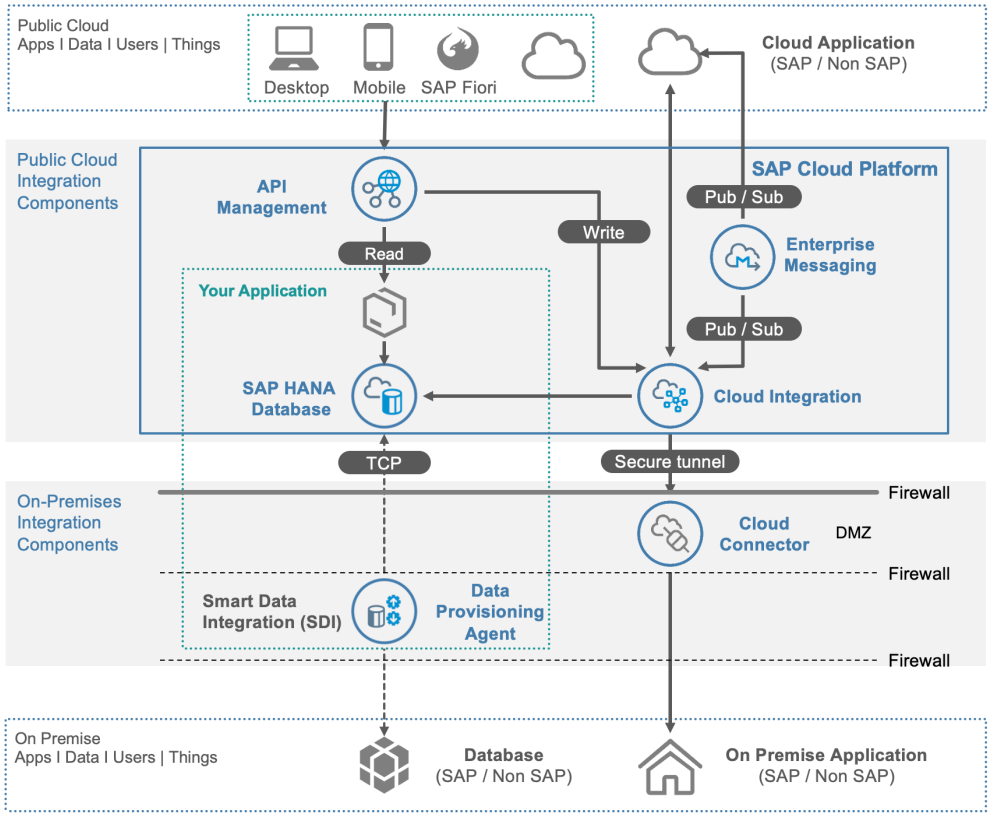
Current
- Web Services
- OData Rest Services
- …

Further
- Operation mode server or client



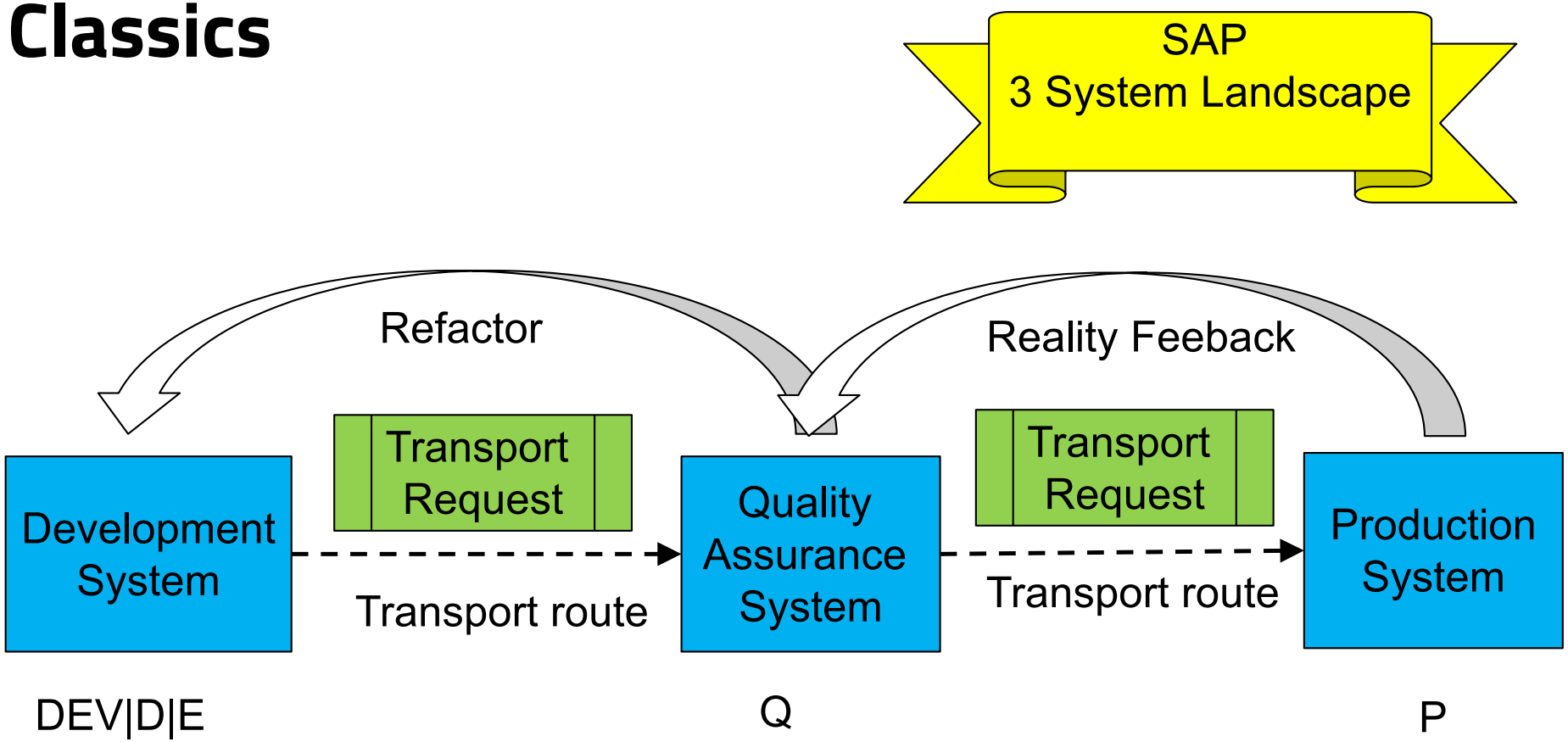
Quelle: [7]

Cloud

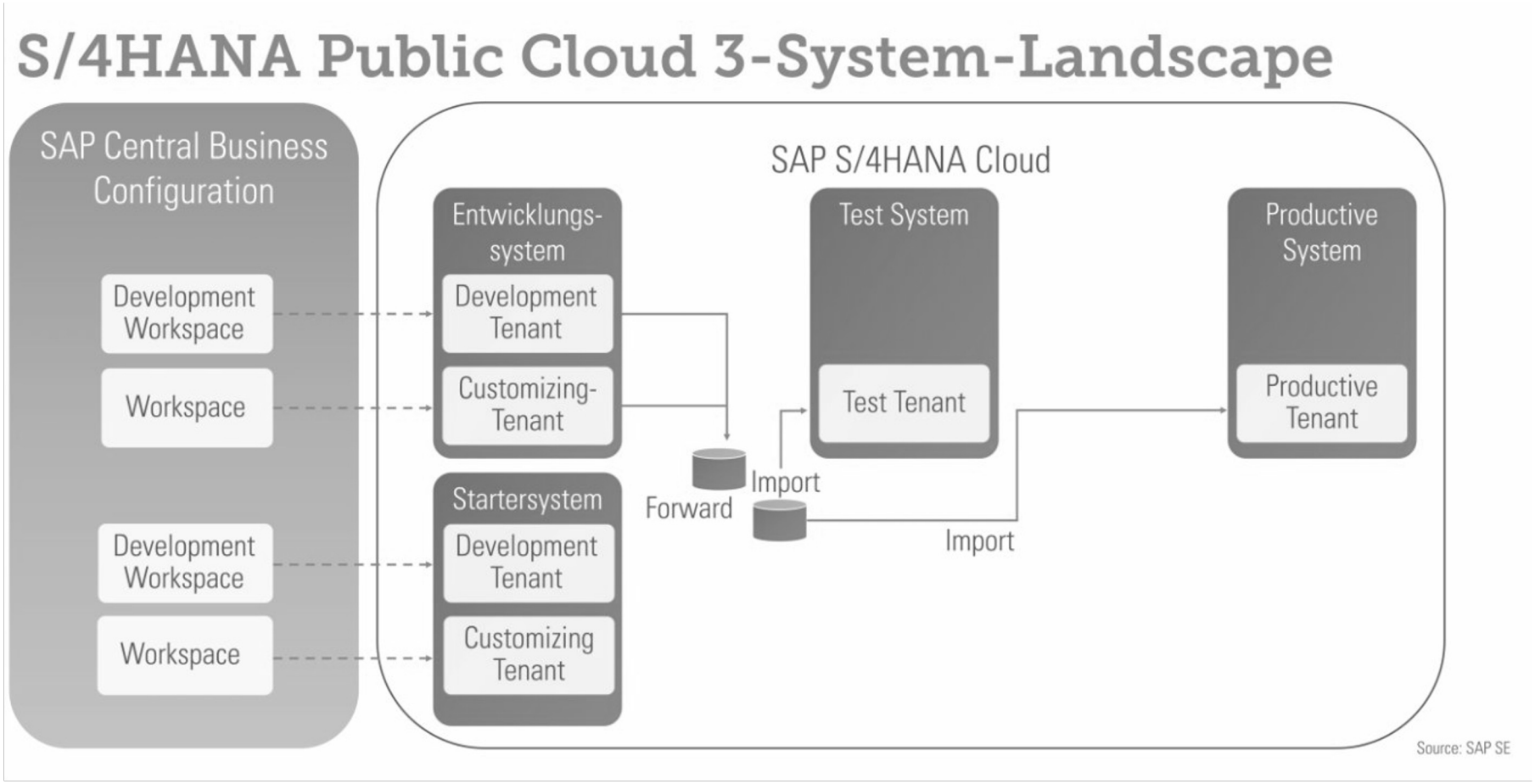


© SAP

Terms
From Development to Production
Classics

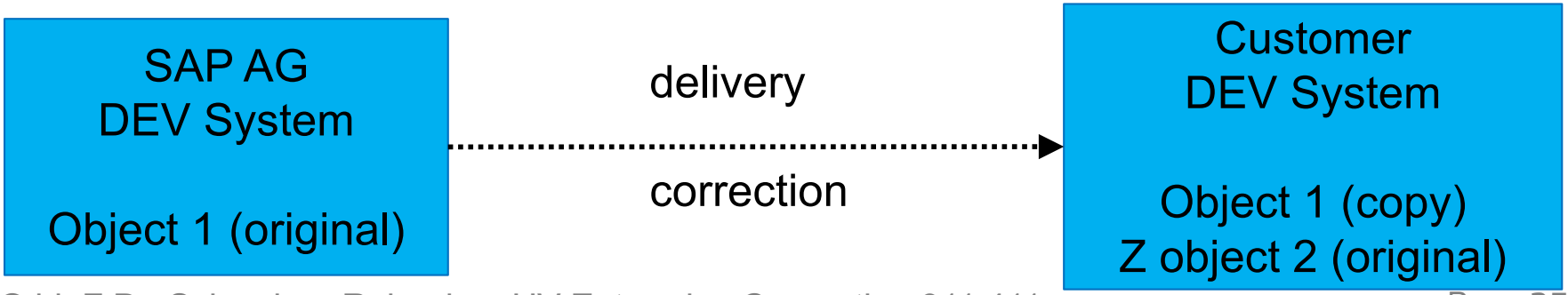


Terms
From Development to Production



Terms
SAP upgrade

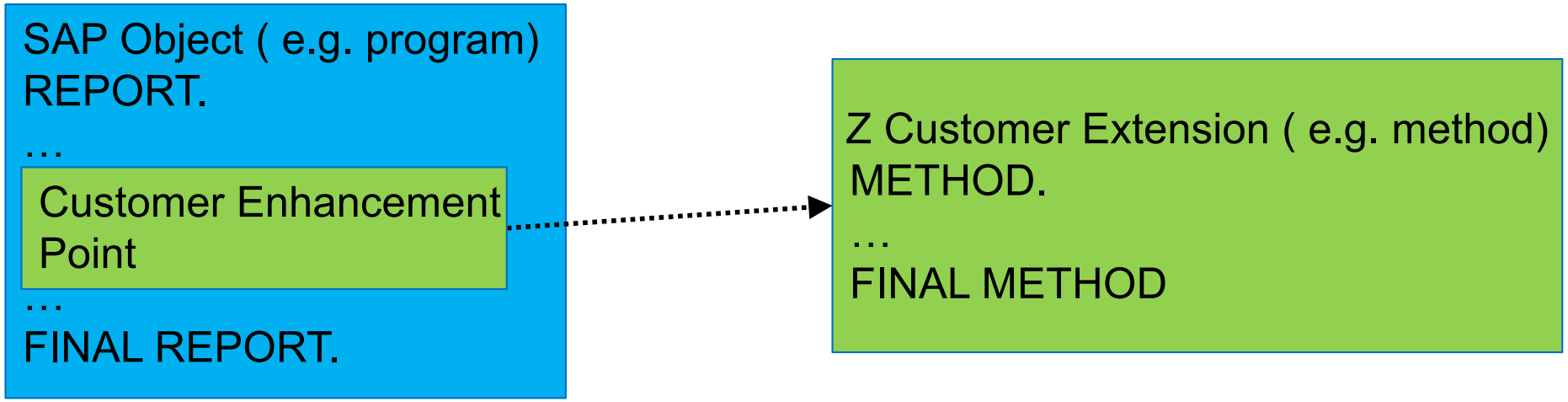
- Original (from SAP or customer)
- Copy (Is the original in the subsequent system)
- Modification (change to the copy of SAP)
- Modification Adjustment (comparison of modifications with the new SAP deliveries)



Terms Extensibility

SAP Extensibility

- Principle (SAP prepares an extension possibility for customer implementation = customer development object in the customer namespace, like Z*)
- Enhancement Framework



Terms Extensibility

NoR
EMPOWER THE EVOLUTION OF NOW

SAP S/4HANA Cloud Public Edition offers support for the following three main extensibility options:

- Key User Extensibility
  - Use key user extensibility for quick, simple extensions.
  - Ensure key users are familiar with the business processes to maximize efficiency.

- **Developer Extensibility**
  - Utilizing the SAP S/4HANA Cloud ABAP Environment and standard ADT tools, it allows for the development of cloud-ready, upgrade-stable custom ABAP code.
- **Side-by-Side Extensibility**
  - Side-by-side extensibility lets you build custom applications or extend existing ones, using the SAP Business Technology Platform (BTP).
  - On SAP BTP, you can develop using your preferred runtime or language, such as ABAP, Java, or Node.js.
  - Use side-by-side extensibility for loosely coupled scenarios.
  - Ensure seamless integration using SAP API Business Hub.
-

Terms
Standard vs. Custom

NoR
EMPOWER THE EVOLUTION OF NOW

SAP standard vs. customer development

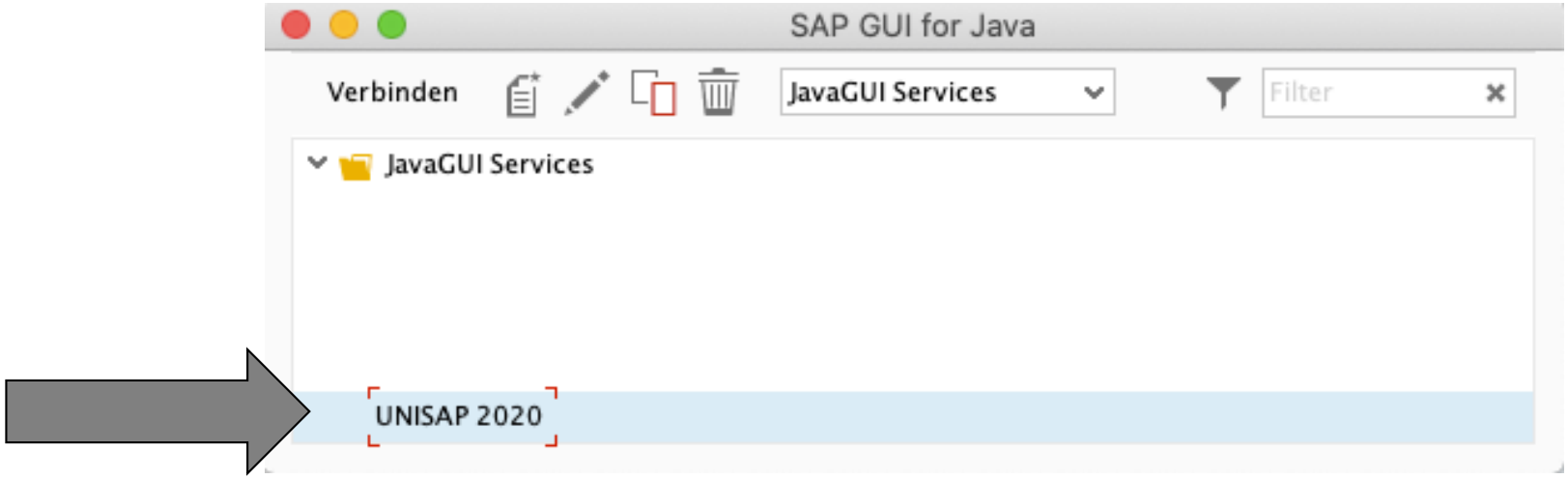
- SAP delivers standard functionality
- Customers implement their own processes to realize a competitive advantage, i.e. customer development (custom development)

Terms
SAP GUI/Logon

GUI for User & Developer

- Open SAP Logon
- Double Click your System to open SAP GUI

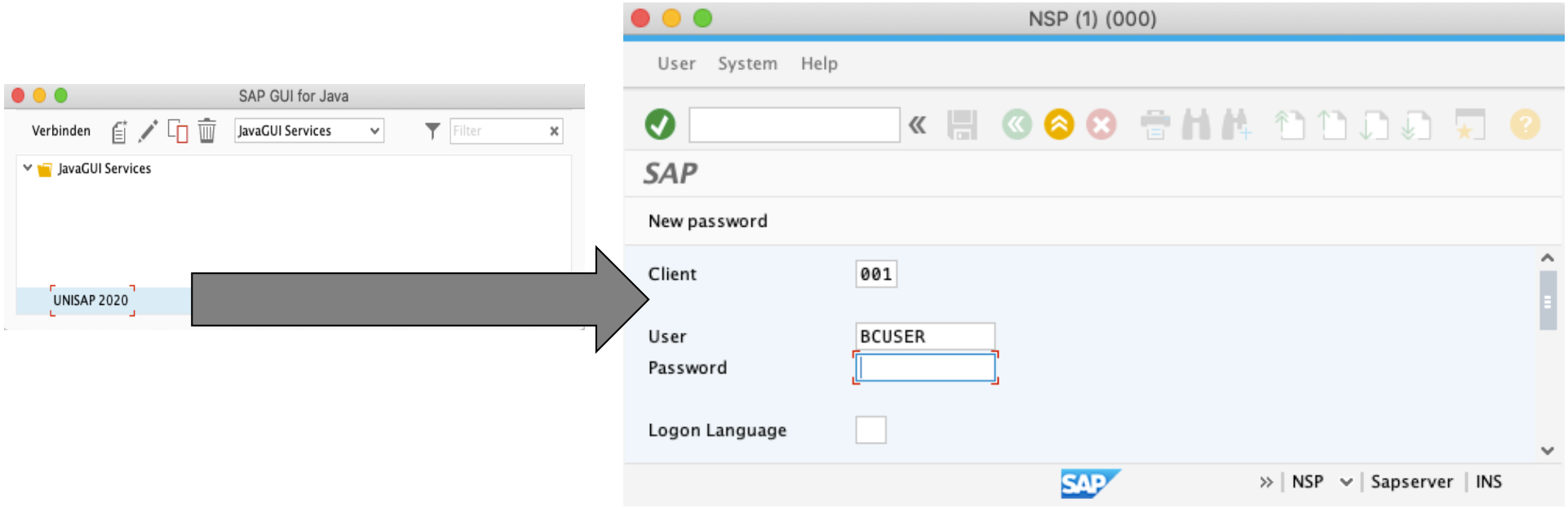
SAPLogon



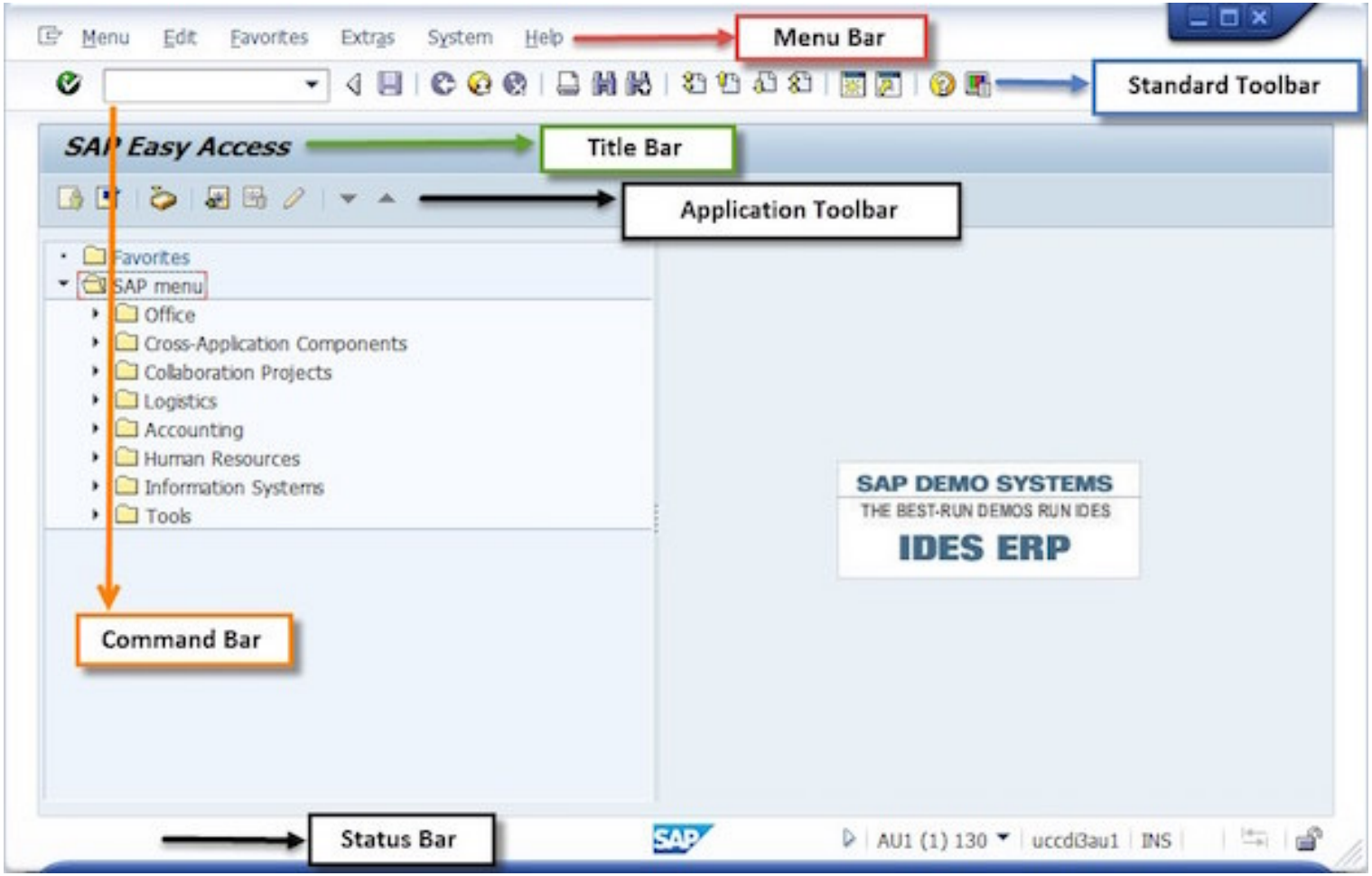
https://developers.sap.com/trials-downloads.html

Terms
SAP GUI Logon

SAP Logon via SAP GUI



Terms
SAP GUI



© SAP

Terms
Transaction Code

NoR
EMPOWER THE EVOLUTION OF NOW

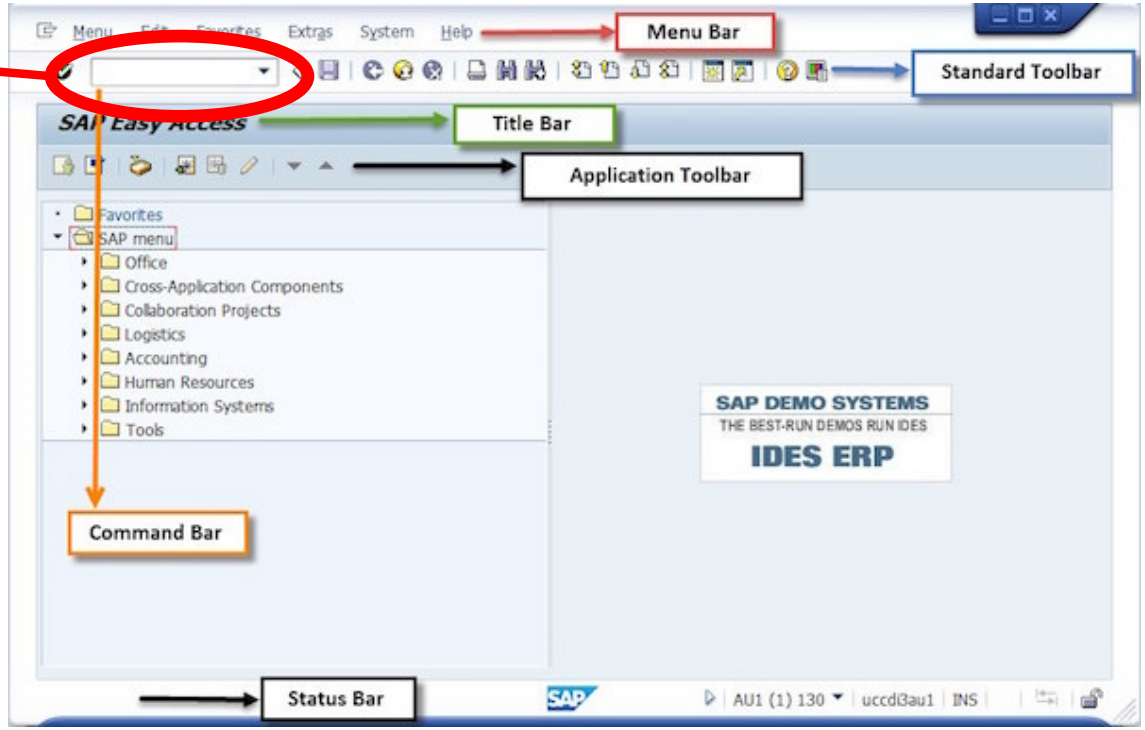
- A transaction code in SAP is a shortcut to functionality, like SE80.
- It has nothing to do with a classical transaction on DB level
- You insert the transaction code in the Command Bar.

Terms
SE80

THE most important transaction code :

SE80

Object Navigator



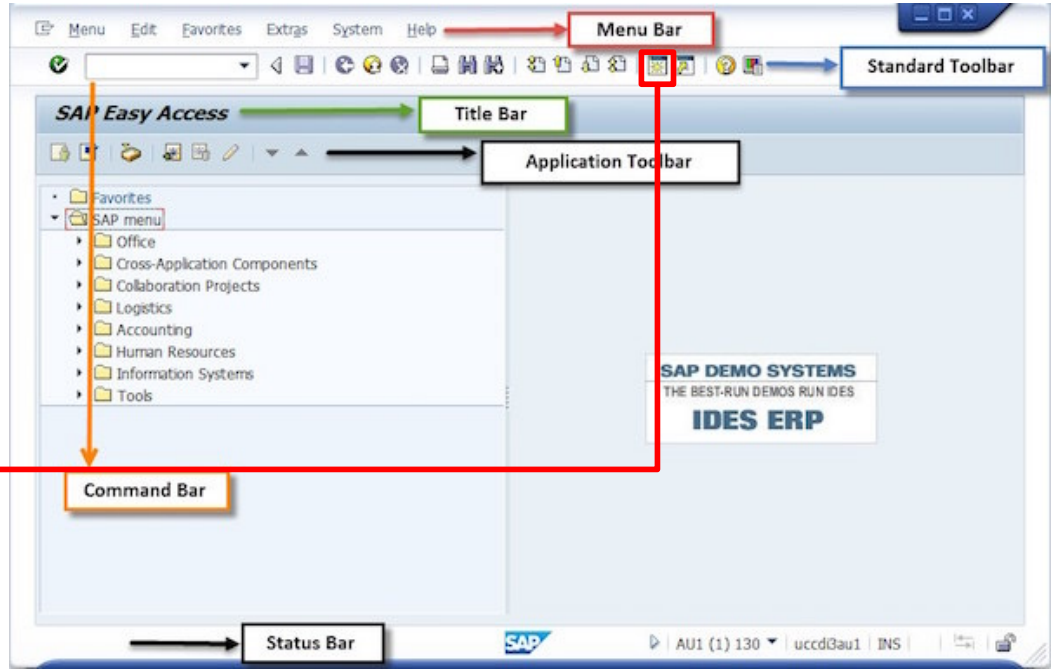
Terms
Prefix transaction code

Prefix for the transaction code :

Prefix + transaction,
e.g. /nSE80 = Object Navigator

Prefix:
- /n = clear actual window and start transaction
- /o = new window

Direct Input:
- /nex = close all windows

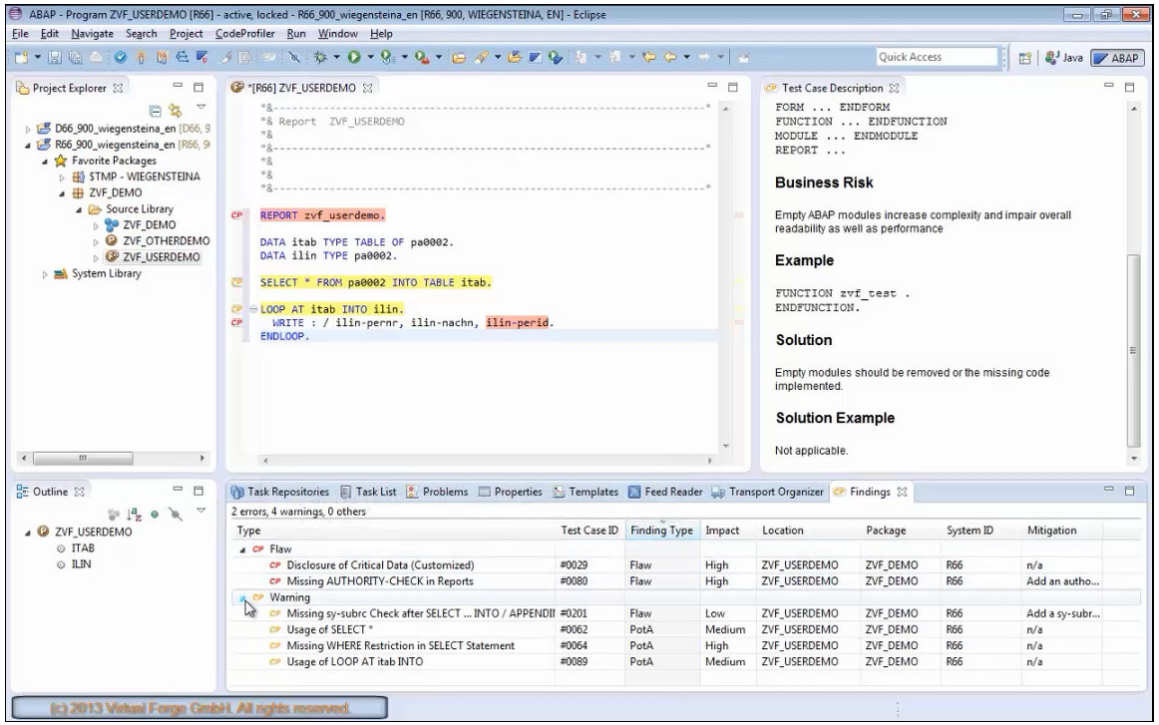


Terms
Eclipse ADT

NoR
EMPOWER THE EVOLUTION OF NOW

- Additional development environment: Eclipse
- Perspective: ABAP Development Tools (ADT)

https://tools.hana.ondemand.com/ _ _

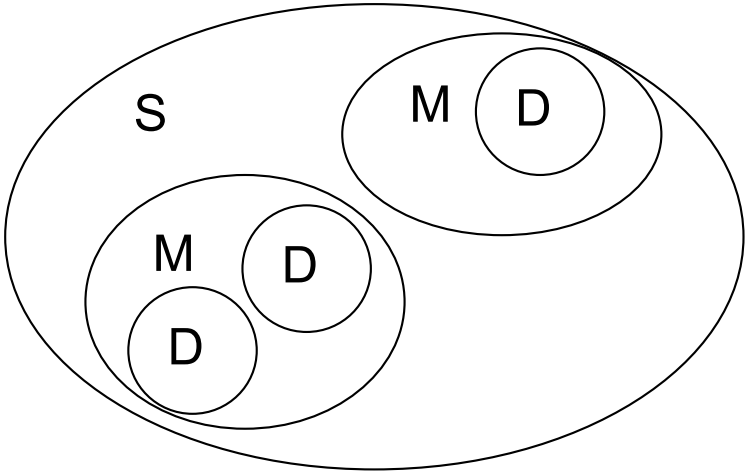


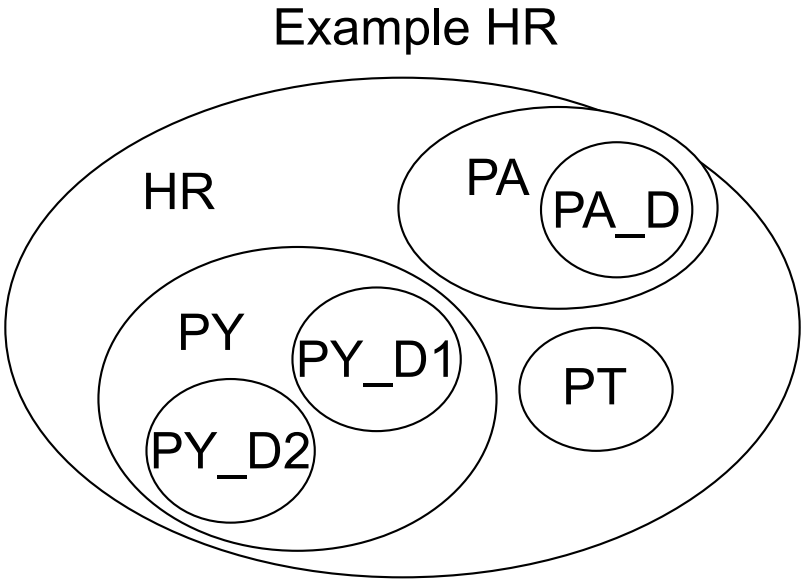
Terms

Package

Development organization packages (~Java Package)

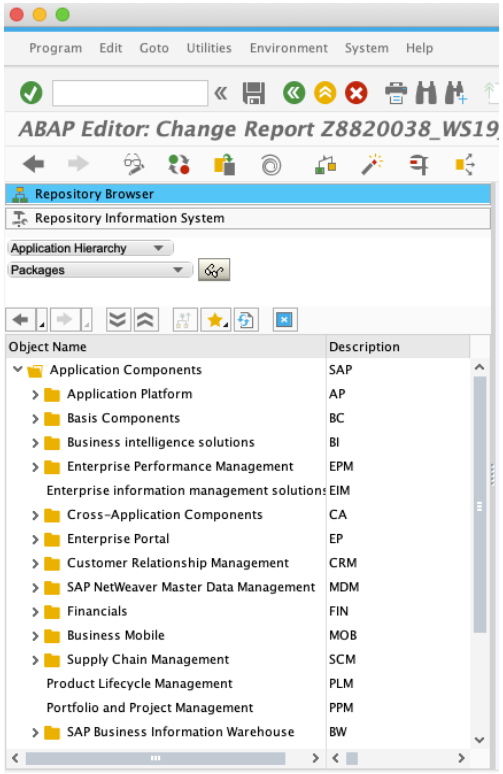
- Structural package (S)
- Main package (M)
- Development package (D)

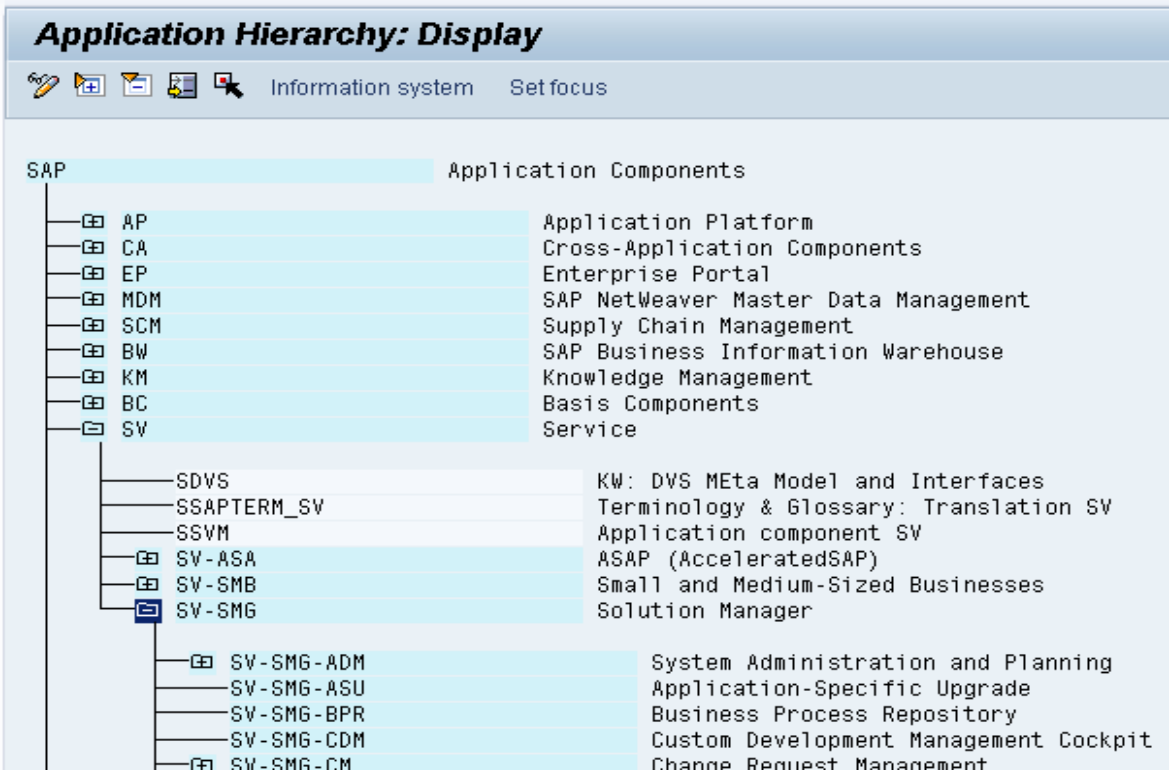




Terms
Enterprise business area

Development organization application component ~ Enterprise Business Area





Terms
Software component

Development organization software component

This describes a set of **development objects** that can only be **delivered together** .

Exception : Packages that are not to be delivered to customers must, however, be assigned to the

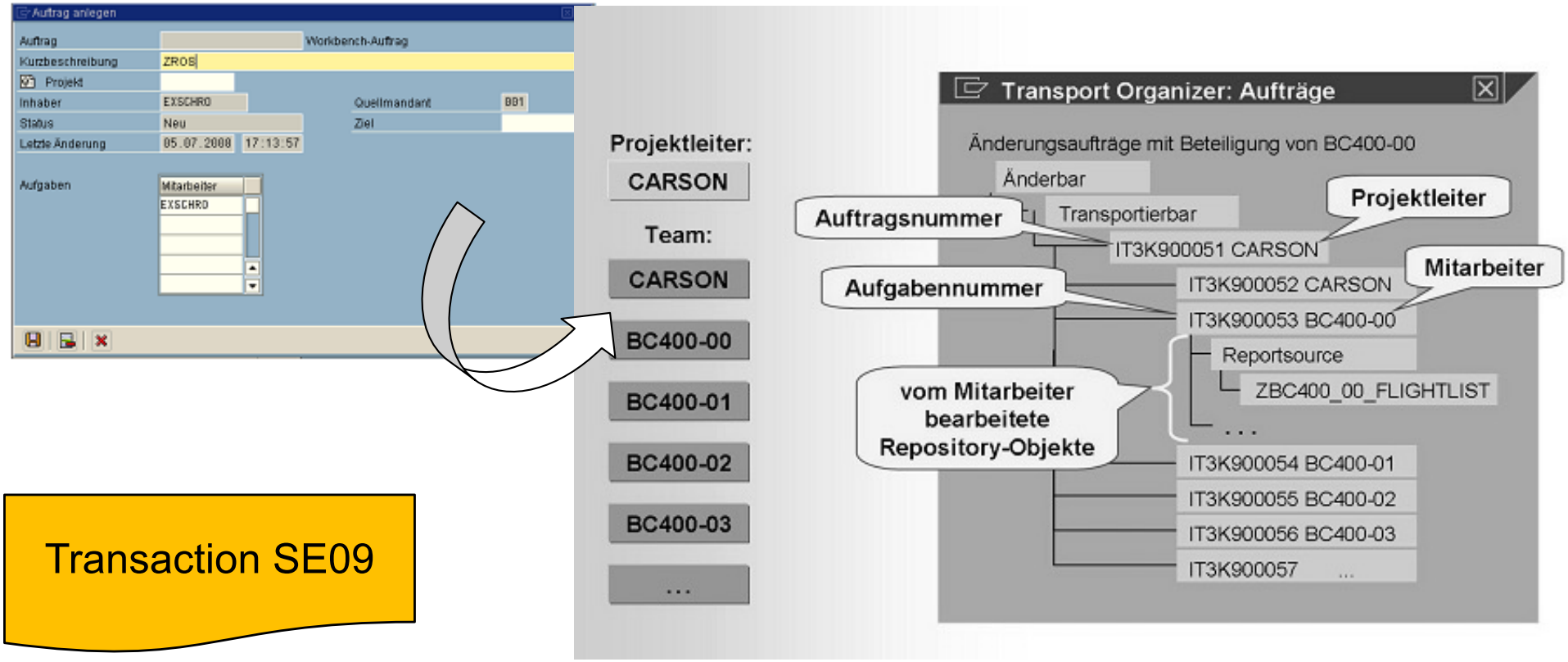
Terms
Transport layer

Development organization transport layer

All development projects that are carried out in an SAP system and transported on the same transport routes are combined into a **transport layer** .

Terms
Transport request

Development organization transport request

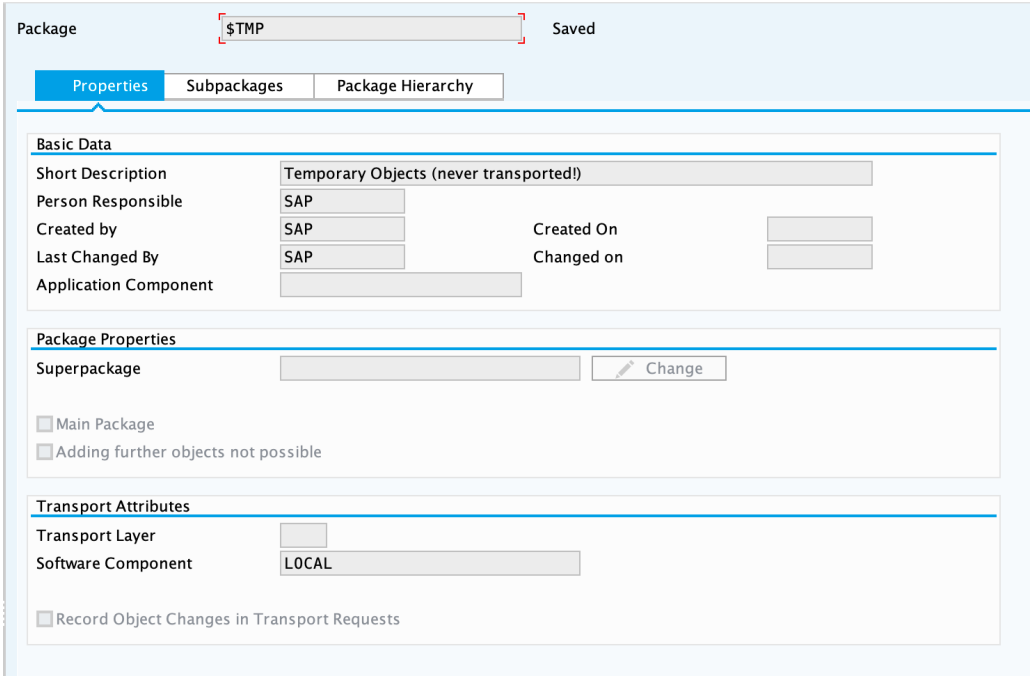


Transaction SE09

Terms
Packages

Development organization : Based on package,

e.g. Local Package $TMP

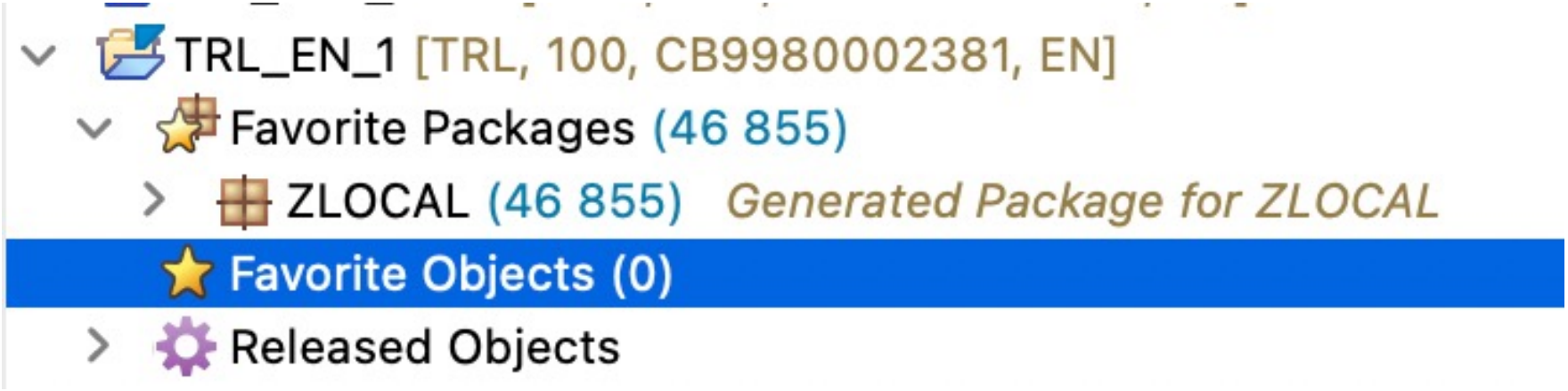


Terms
Packages

NoR
EMPOWER THE EVOLUTION OF NOW

Development organization : Based on package,

e.g. Local Package ZLOCAL

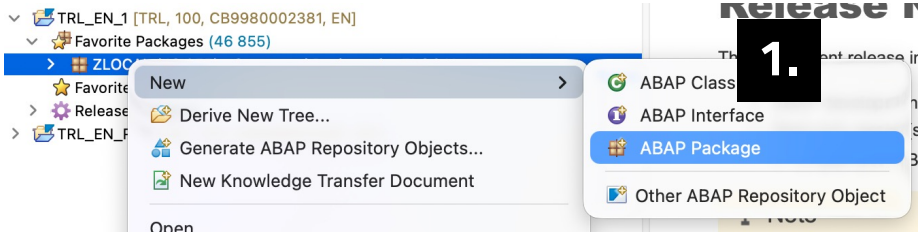


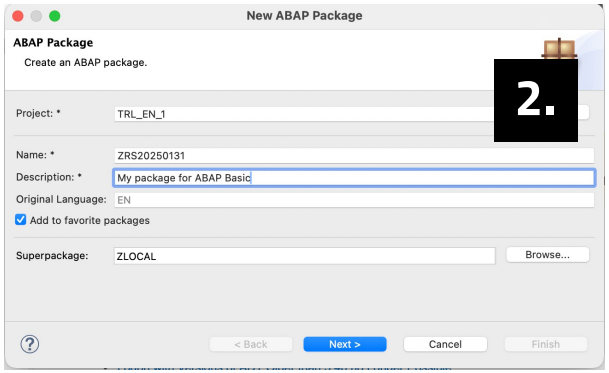
Terms
Do it yourself!

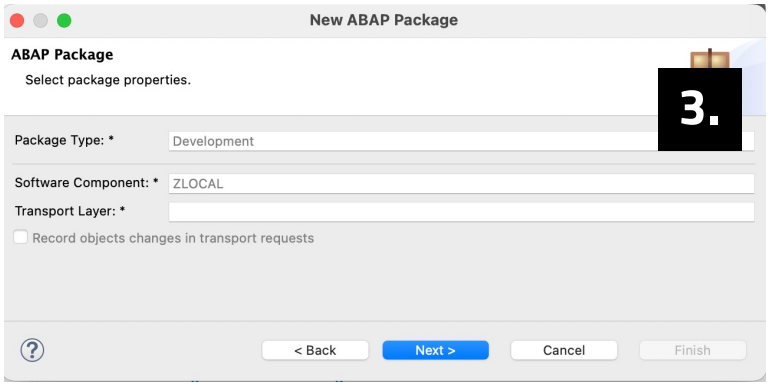
NoR
EMPOWER THE EVOLUTION OF NOW

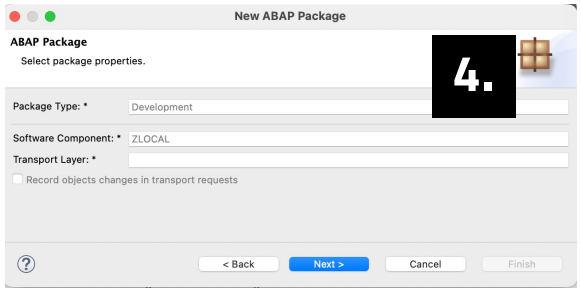
Create a development package below ZLOCAL: Name = Z<MATNR>

1. Use the context menu in ADT on ZLOCAL
2. Fill out the creation form
3. Save

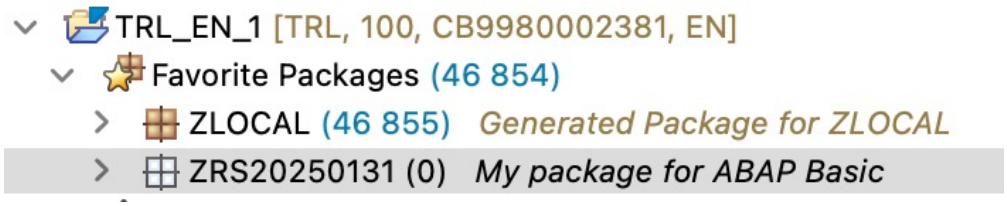


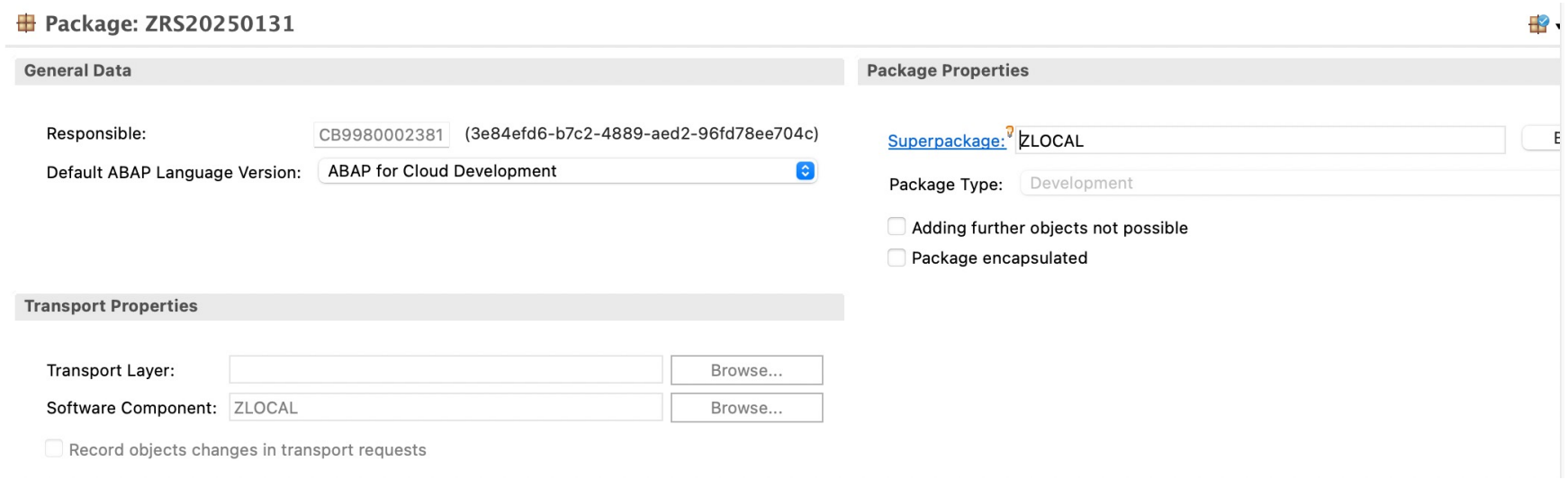






Terms
Do it yourself!





Terms
Namespaces

SAP delivers its own software developments to the customer. The customer can develop their own software. In principle, there can be name overlaps between SAP objects and customer objects .

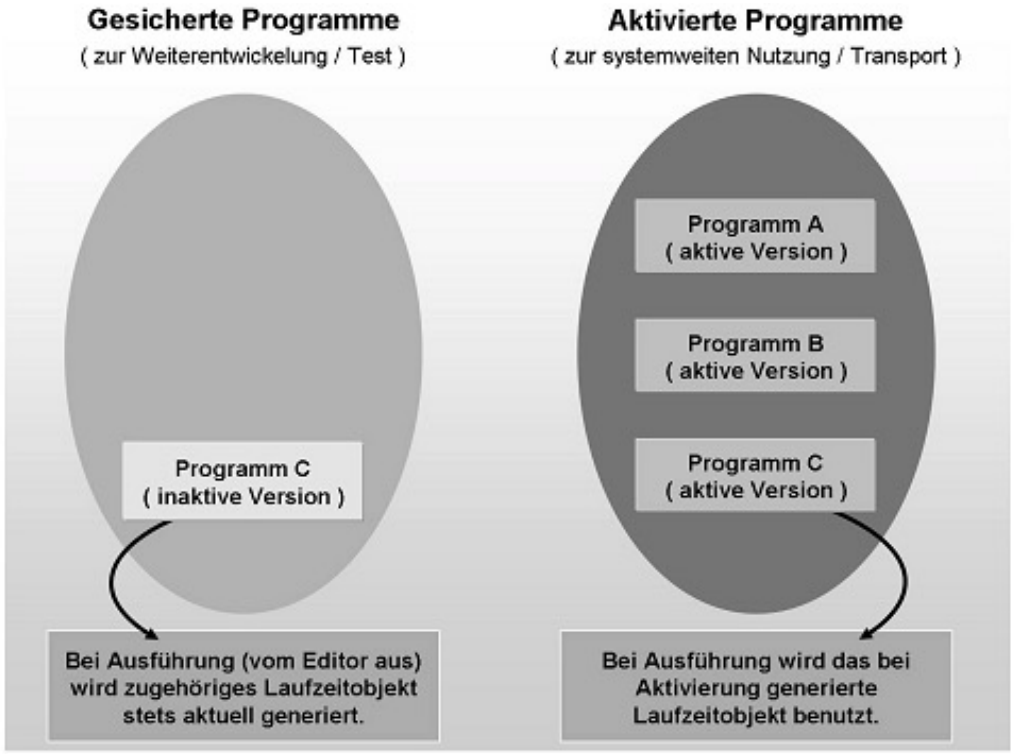
There are two variants (namespaces) to protect customer objects.

1. If development objects **begin with Z or Y** , they are implicitly protected because SAP does not deliver objects beginning with Z or Y (exception: customer exits)

2. **Reserved namespace** from SAP . This has 10 digits and begins with "/" and ends with "/". This namespace must be prefixed to the development objects. This variant is particularly interesting for third-party providers.

Terms
Activation State

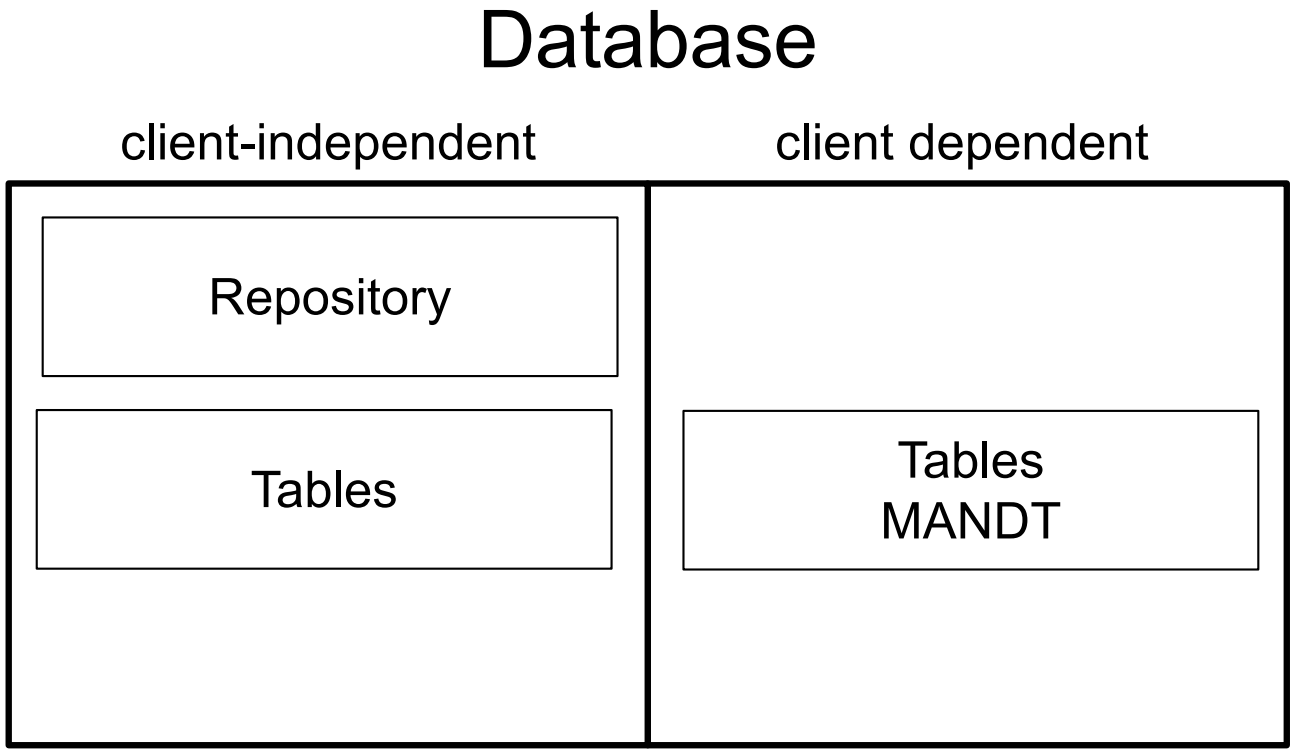
- **Repository objects**
  - Activity States
  - Program Types
  - Function Modules
  - …
- **Dictionary objects**
  - Activity States
  - Domain
  - Data Elements
  - Structure
  - Transparent Table



© SAP

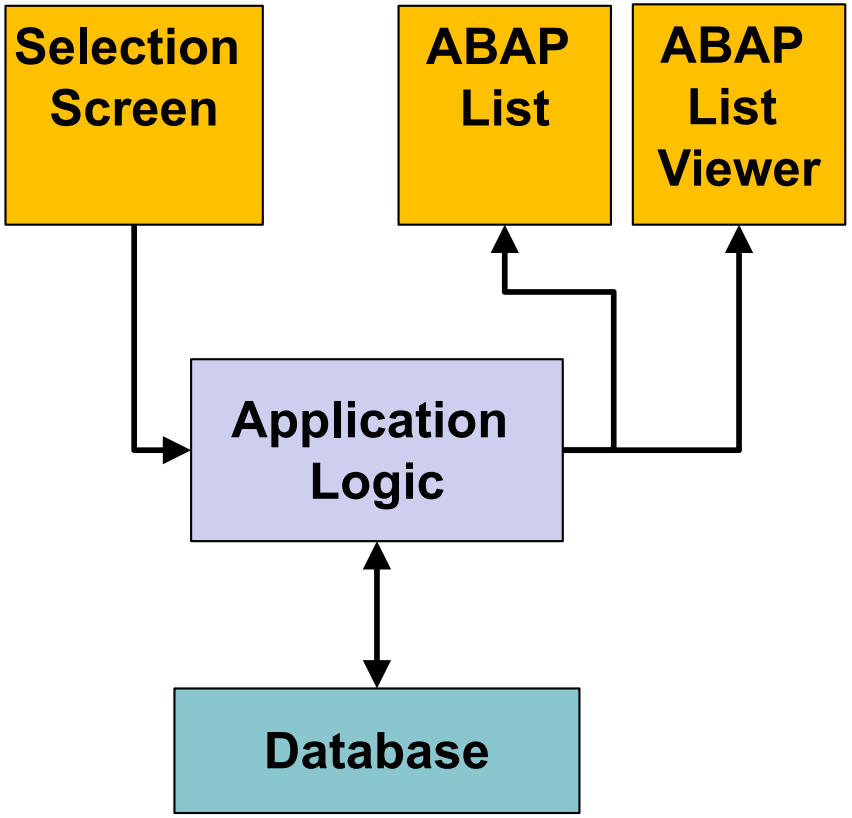
Terms
Repository Object

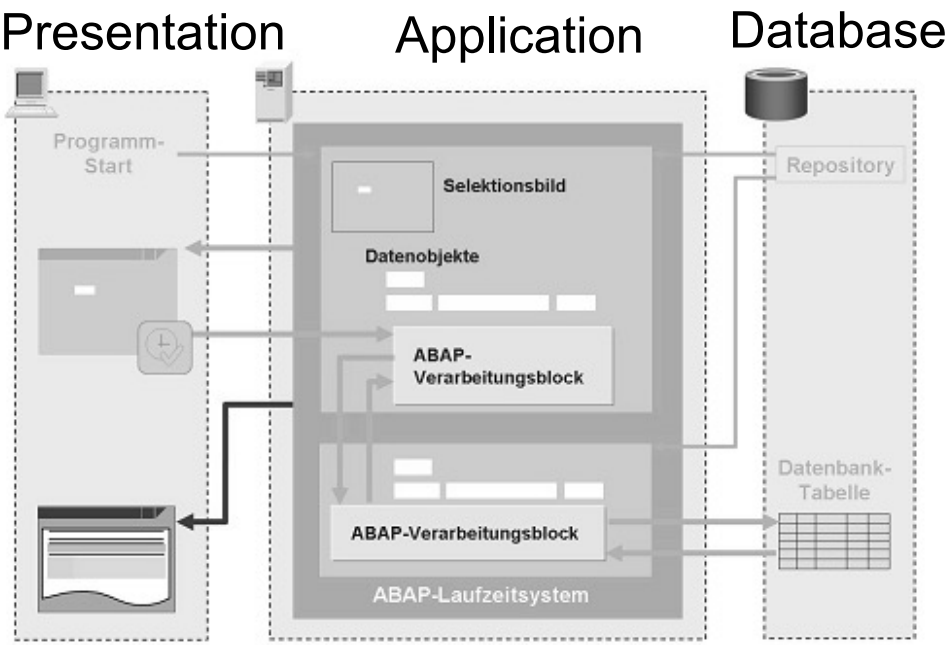
- **Repository Objects** (repository structure)



Terms
Processing Sequence

- Example program with exemplary sequence





Example

```
*&---------------------------------------------------------------------*
*& report Z8820038_WS19_HELLO_WORLD
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

REPORT z8820038_ws19_hello_world.

* Airline input field
PARAMETERS: pa_car TYPE scarr-carrid .

* Target data object for FG name
DATA gd_carrname TYPE scarr-carrname .

***
INITIALIZATION.
* Initialization for input field
  pa_car = 'LH'.

LOAD OF PROGRAM.

START OF SELECTION.

* Reading the FG from the DB table SCARR
SELECT SINGLE carrname FROM scarr INTO gd_carrname
WHERE carrid = pa_car .


END OF SELECTION.

* Output to standard list
WRITE ' Hello World!'.
* New line
  new line .

* Output of the name of the FG
WRITE: / 'FG name: ', gd_carrname .
```

Terms: Sample program MVC pattern

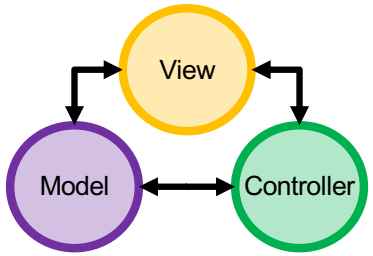

```
REPORT z8820038_ws19_hello_world.

* Airline input field
PARAMETERS: pa_car TYPE scarr-carrid .

* Target data object for FG name
DATA gd_carrname TYPE scarr-carrname .

***
INITIALIZATION.
* Initialization for input field
  pa_car = 'LH'.

LOAD OF PROGRAM.

START OF SELECTION.

* Reading the FG from the DB table SCARR
SELECT SINGLE carrname FROM scarr INTO gd_carrname
WHERE carrid = pa_car .

END OF SELECTION.

* Output to standard list
WRITE ' Hello World!'.
* New line
  new line .

* Output of the name of the FG
WRITE: / 'FG name: ', gd_carrname .
```

Selection screen = View

Business data = model

Business logic = model

Results list = View

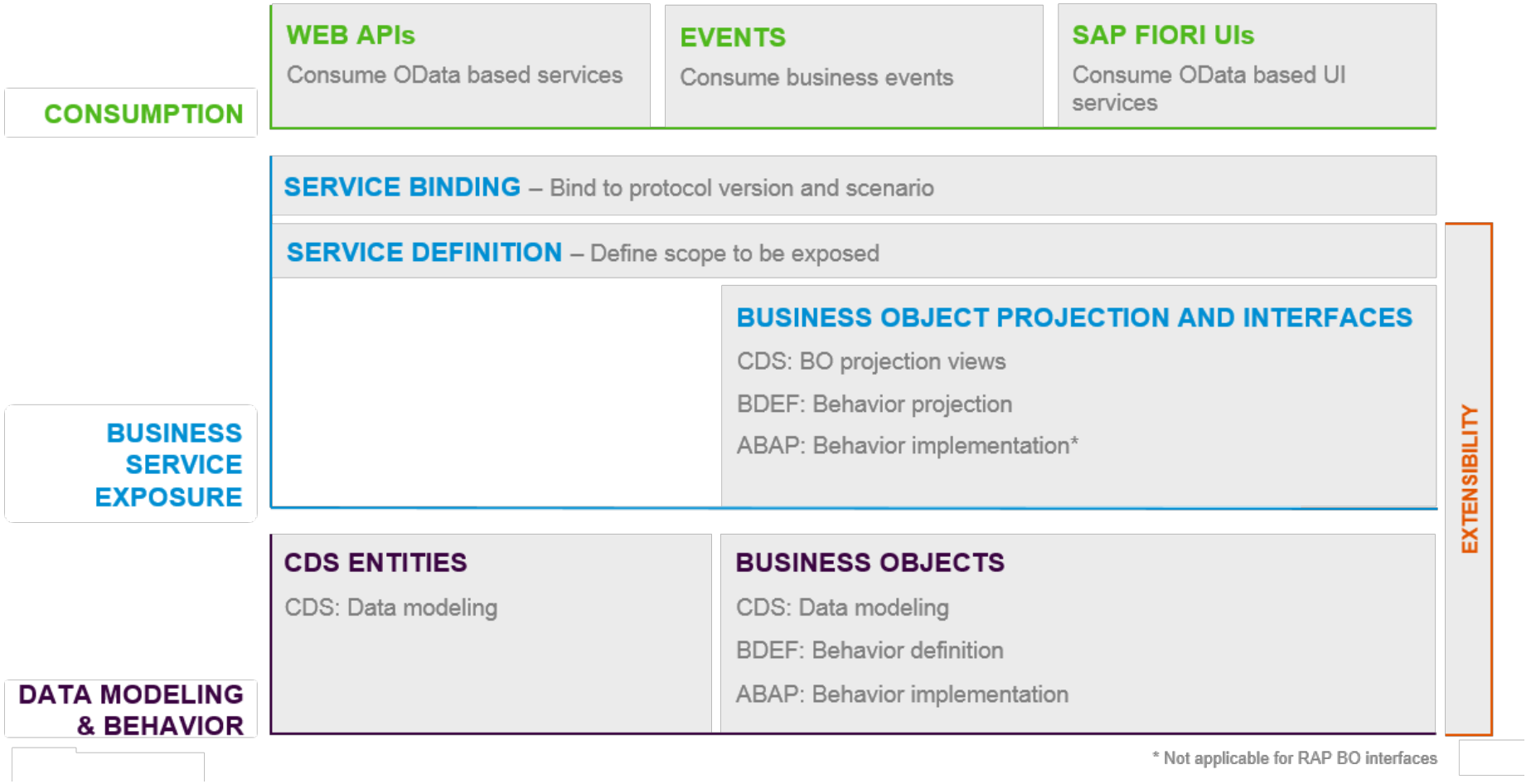


Controller

Development Model RAP

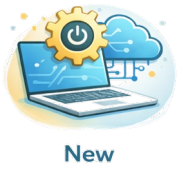
NoR
EMPOWER THE EVOLUTION OF NOW

RAP - The big picture

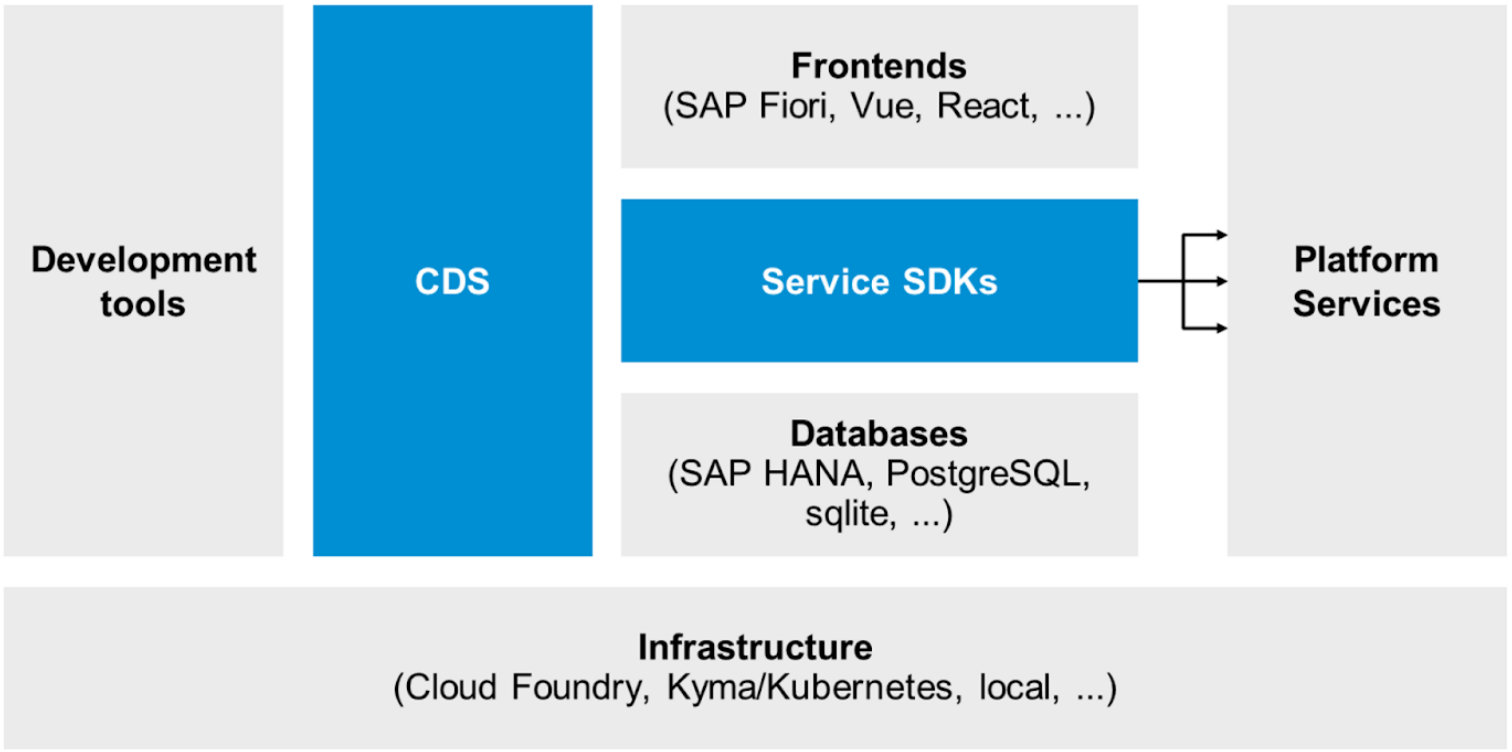


(c) SAP

Development Model CAP



NoR
EMPOWER THE EVOLUTION OF NOW

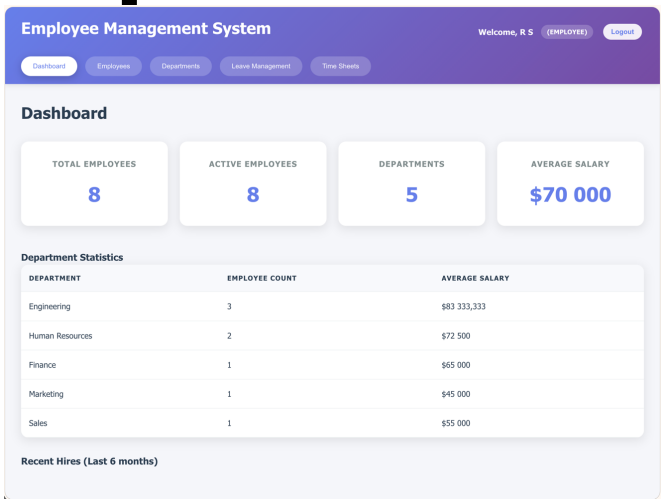


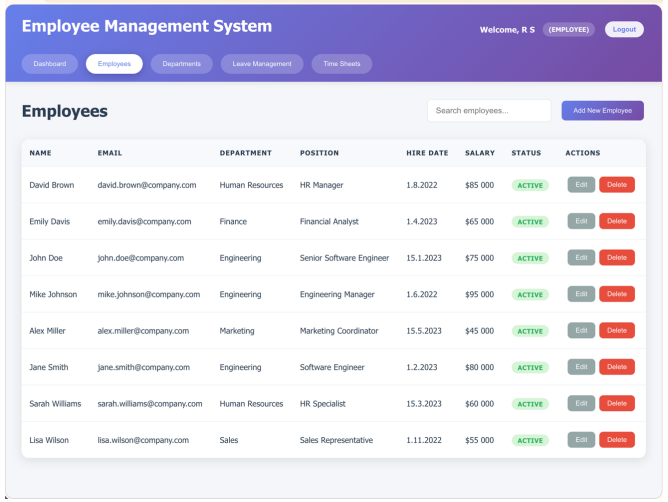
node express Java POWERED spring

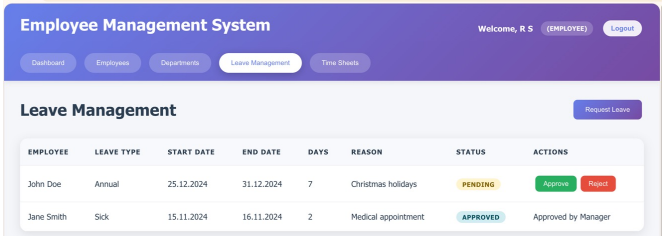
(c) SAP

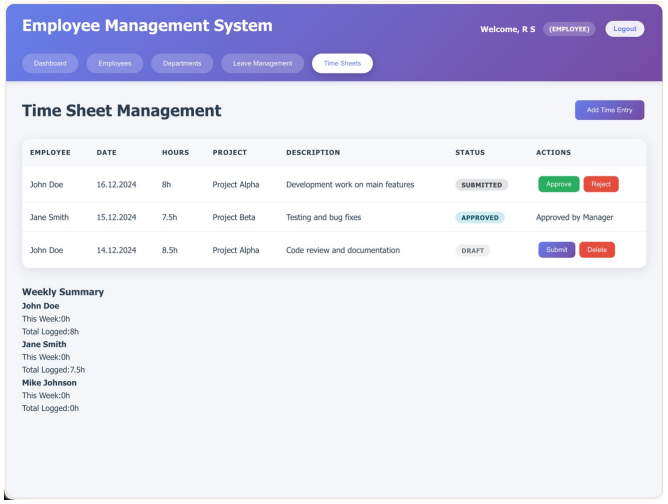
Development Model CAP Example

NoR
EMPOWER THE EVOLUTION OF NOW





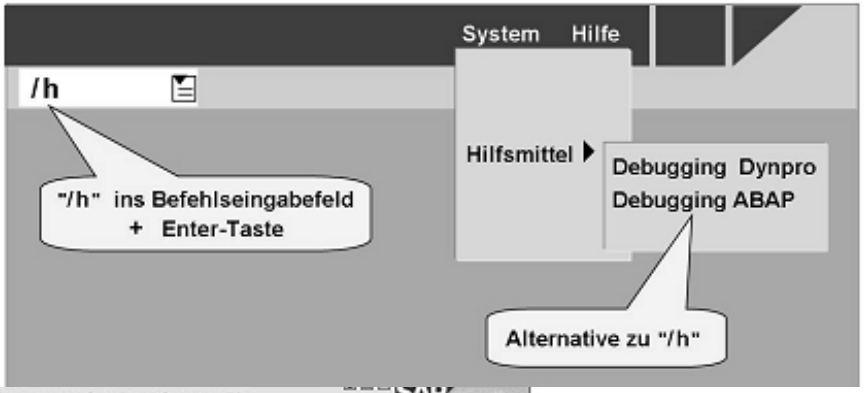


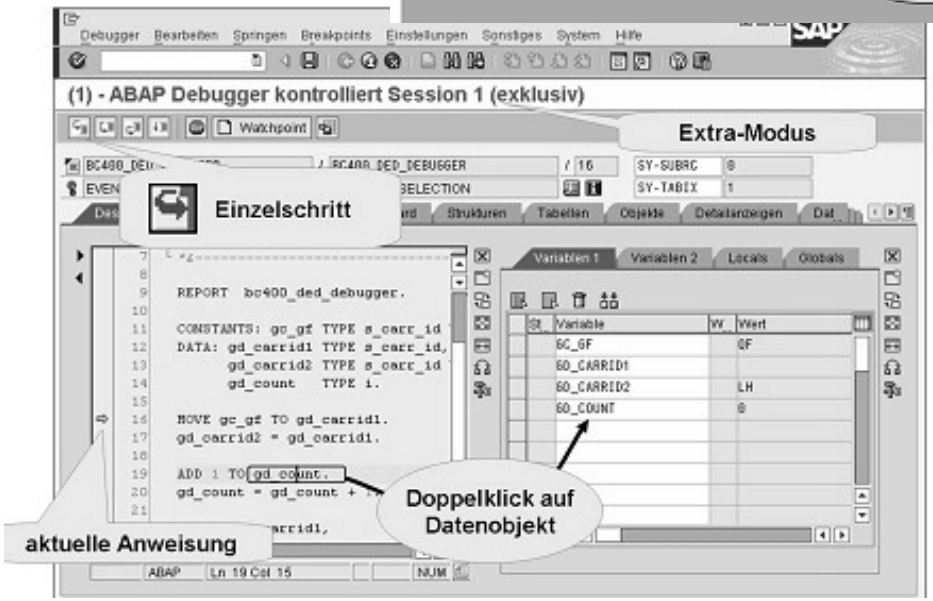


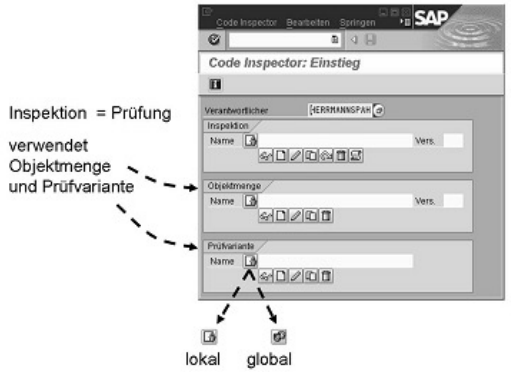
Terms
Code analysis

Dynamic vs. Static Analysis

- Debugger (/h)
- Code inspector (TA SCI)
  - Test Variant
  - Object Set
  - Inspection





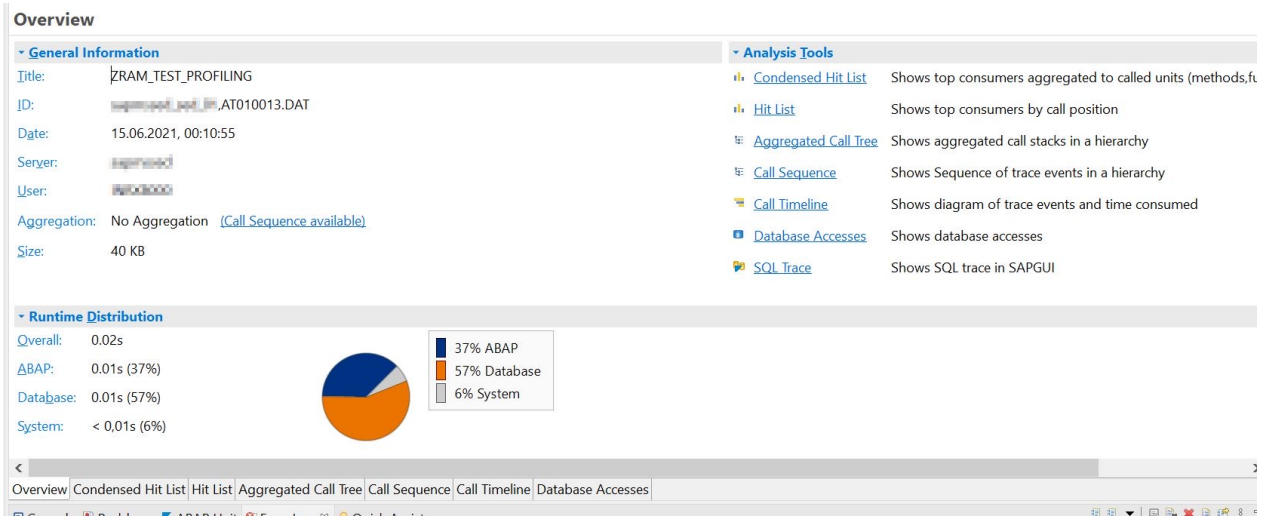


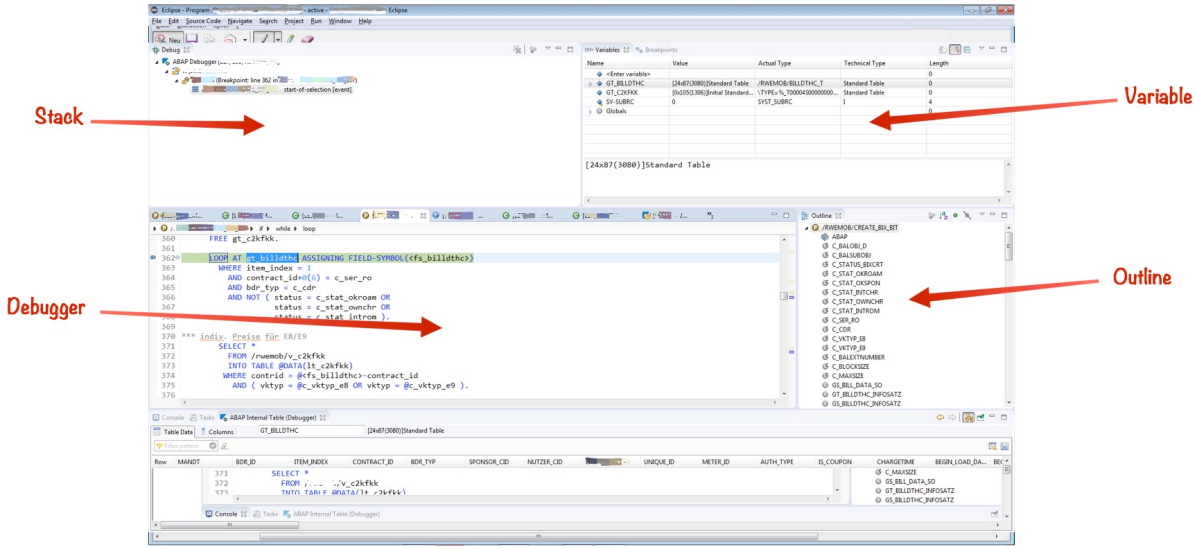
Terms
Code analysis

NoR
EMPOWER THE EVOLUTION OF NOW

- **ABAP Test Cockpit (ATC) (Static Analysis):** Identifies syntax errors, coding standard violations (e.g., naming conventions), and potential security vulnerabilities. It supports remote analysis for system conversions (e.g., S/4HANA readiness).
- **ABAP Profiler (Dynamic/Performance Analysis):** Integrated into ADT to trace code execution, analyzing runtime, database accesses, and performance bottlenecks. It provides views like call trees and timelines to pinpoint slow code.
- **ABAP Debugger:** Allows interactive analysis of code at runtime, including breakpoints, stepping through code, and inspecting variable values, including specialized views for internal tables.
- **Code Inspector (SCI):** Traditional static code analysis tool integrated into ADT for checking compliance with development standards.

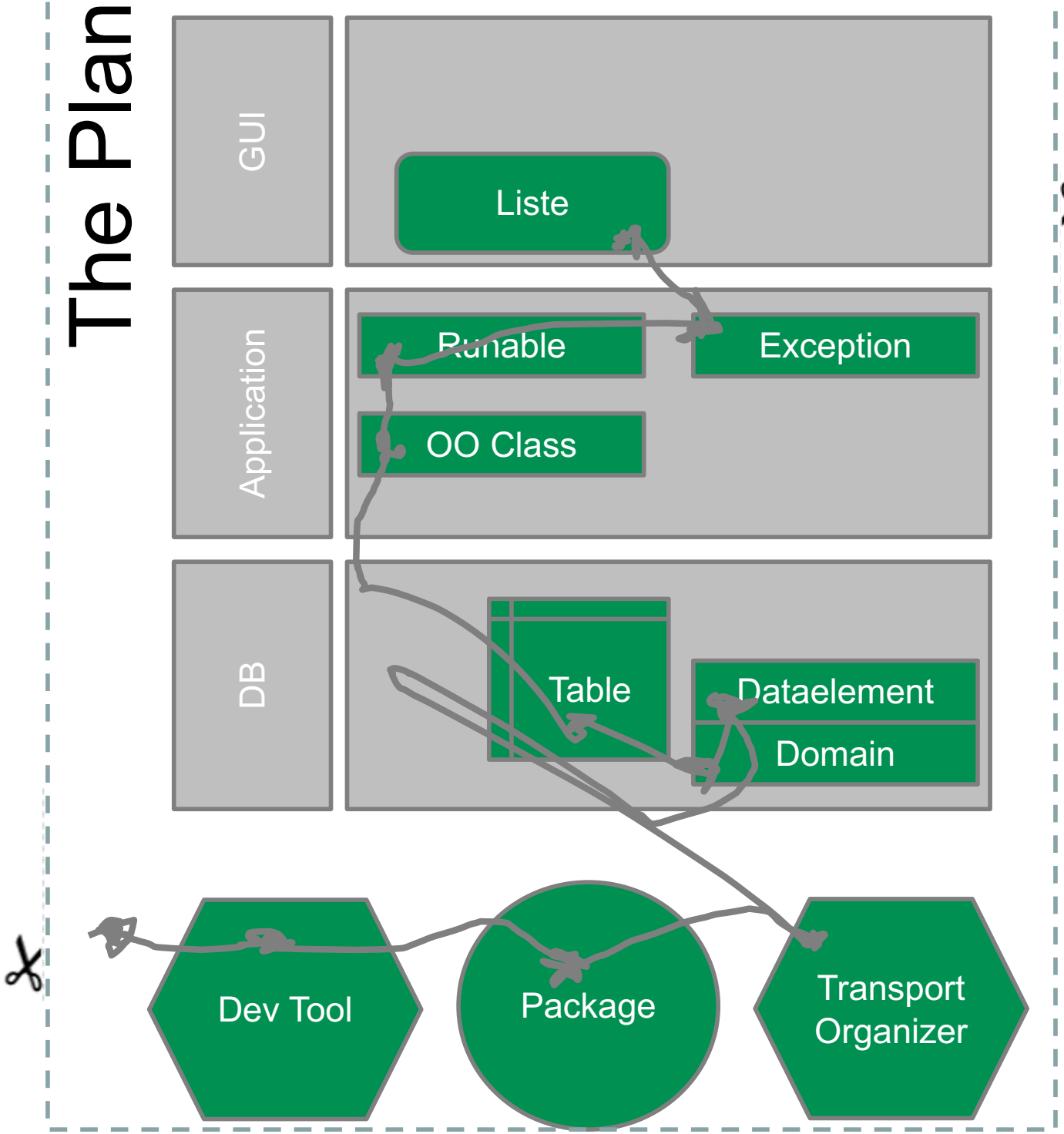
Analysis Tools





Plan

The Plan

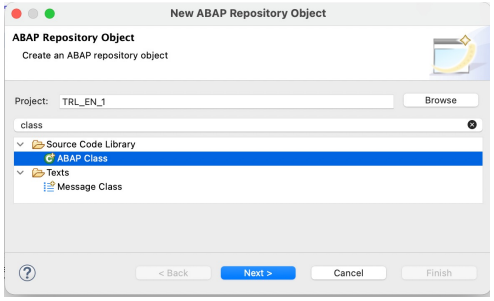


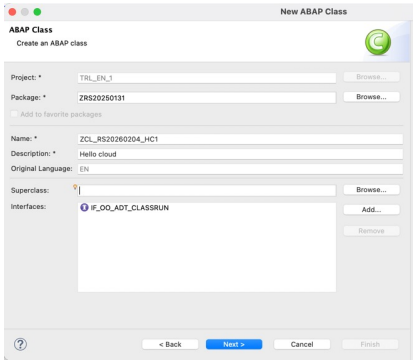
NoR
EMPOWER THE EVOLUTION OF NOW

Hello Cloud

Create a console application

- Create an ABAP class with interface IF_OO_ADT_CLASSRUN
  - ZCL_<INITIAL>20260204_HC





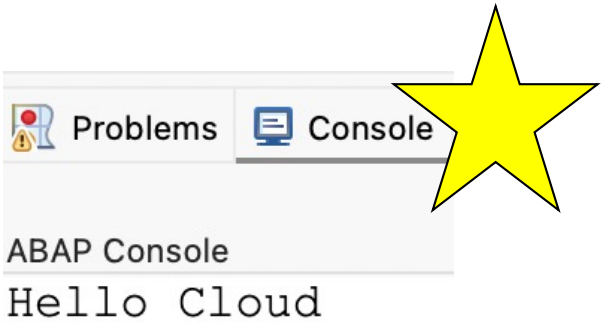
Created class
Short introduction

```
CLASS zcl_rs20260204_hc1 DEFINITION PUBLIC FINAL CREATE PUBLIC .

PUBLIC SECTION.
INTERFACES if_oo_adt_classrun .

PROTECTED SECTION.

PRIVATE SECTION.
ENDCLASS.


CLASS zcl_rs20260204_hc1 IMPLEMENTATION.

METHOD if_oo_adt_classrun~main.
* Here goes the code
ENDMETHOD.

ENDCLASS.
```

Great code!

```
CLASS zcl_rs20260204_hc1 DEFINITION PUBLIC FINAL CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES if_oo_adt_classrun .
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.

CLASS zcl_rs20260204_hc1 IMPLEMENTATION.

METHOD if_oo_adt_classrun~main.
* Here goes the code
out->write( 'Hello Cloud' ).
ENDMETHOD.

ENDCLASS.
```

1. Activate and test your class.
    1. Activate the class with the keyboard shortcut `Ctrl + F3`.
    2. Run the class with the F9 key.
2. Check the output in the *Console* view of Eclipse.
    1. Check the *Console* view that should have opened as a new tab below the editor view.
    2. If the *Console* view is not visible, open it by choosing *Window→Show view→Other*. Double-click *Console* in the hit list.



Dictionary

Dictionary

Structured Analysis and Structured Design (SA/SD) is a traditional software development methodology that was popular in the 1980s and 1990s. It involves a series of techniques for designing and developing software systems in a structured and systematic way. Here are some key concepts of SA/SD:

- **Functional Decomposition**: SA/SD uses functional decomposition to break down a complex system into smaller, more manageable subsystems. This technique involves identifying the main functions of the system and breaking them down into smaller functions that can be implemented independently.
- **Data Flow Diagrams** (DFDs): SA/SD uses DFDs to model the flow of data through the system. DFDs are graphical representations of the system that show how data moves between the system's various components.
- **Data Dictionary**: A data dictionary is a central repository that contains descriptions of all the data elements used in the system. It provides a clear and consistent definition of data elements, making it easier to understand how the system works.
- **Structured Design**: SA/SD uses structured design techniques to develop the system's architecture and components It involves identifying the major components of the system, designing the interfaces between them, and specifying the data structures and algorithms that will be used to implement the system.
- **Modular Programming**: SA/SD uses modular programming techniques to break down the system's code into smaller, more manageable modules. This makes it easier to develop, test, and maintain the system.

Data Objects

Data Objects

Variables

- Value may change during runtime
- Identified by a name
- Well-defined starting value

Constants

- Value hard-coded in source code
- Identified by a name

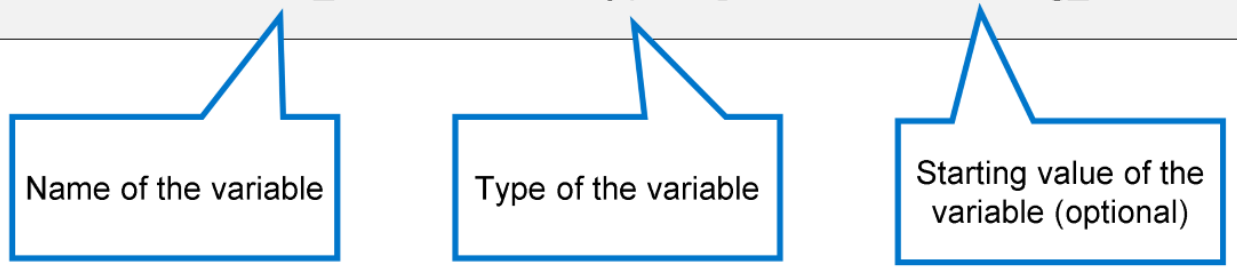
Literals

- Value hard-coded in source code
- Anonymous (no name)

Declaration of variables

Statement DATA

```
DATA <variable_name> TYPE <type> [ VALUE <starting_value> ].
```



Examples

```
DATA my_var1 TYPE i.
DATA my_var2 TYPE string.
DATA my_var3 TYPE string VALUE `Hello World`.
```

ABAP Dictionary (DDIC)
Data Types

General Data Types

Scope of Types

- Built in types in ABAP
- Custom Types
- ABAP Dictionary Types

Kind of Types

- Elementary Types
- Structured Types
  - Structure
  - View
  - Transparent Table
- Table Types
- Referential Types

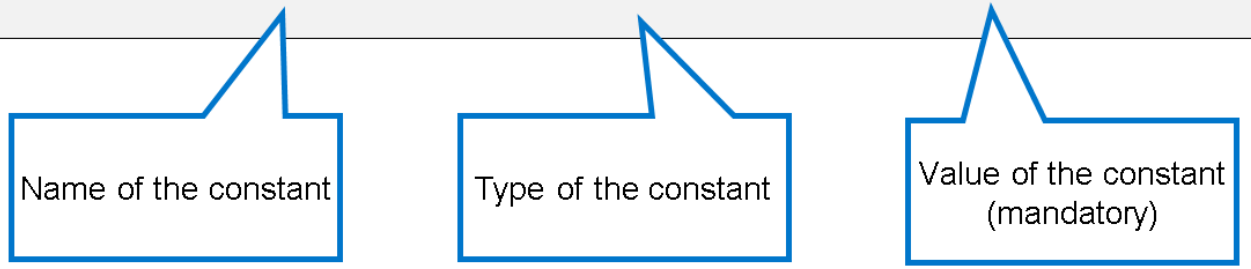
Try it out!

```
* Data Objects with Built-in Types
*********************************************************************

   " comment/uncomment the following declarations and check the output
   DATA variable TYPE string.
*    DATA variable TYPE i.
*    DATA variable TYPE d.
*    DATA variable TYPE c LENGTH 10.
*    DATA variable TYPE n LENGTH 10.
*    DATA variable TYPE p LENGTH 8 DECIMALS 2.

* Output
*********************************************************************

   out->write(  'Result with Initial Value)' ).
   out->write(   variable ).
   out->write(  '---------' ).

   variable = '19891109'.

   out->write(  'Result with Value 19891109' ).
   out->write(   variable ).
   out->write(  '---------' ).
```

Constants

Statement CONSTANTS

```
CONSTANTS <constant_name> TYPE <type>  VALUE <value>.
```



Examples

```
CONSTANTS my_const1    TYPE i VALUE 12345.

CONSTANTS my_const2    TYPE string VALUE `Hello World`.

CONSTANTS my_const3    TYPE C LENGTH 3 VALUE IS INITIAL.
```

Literals

Number Literals 123, -123

- integer numbers
- With or without sign
- Data type: I or P (with 0 decimals)

Text Literals 'LH', '123.45'

- Marked with simple quote (')
- TYPE C, length without trailing blanks

String Literals `Hello`, `What's up?`

- Marked with backquote (`)
- TYPE STRING

Try it out!

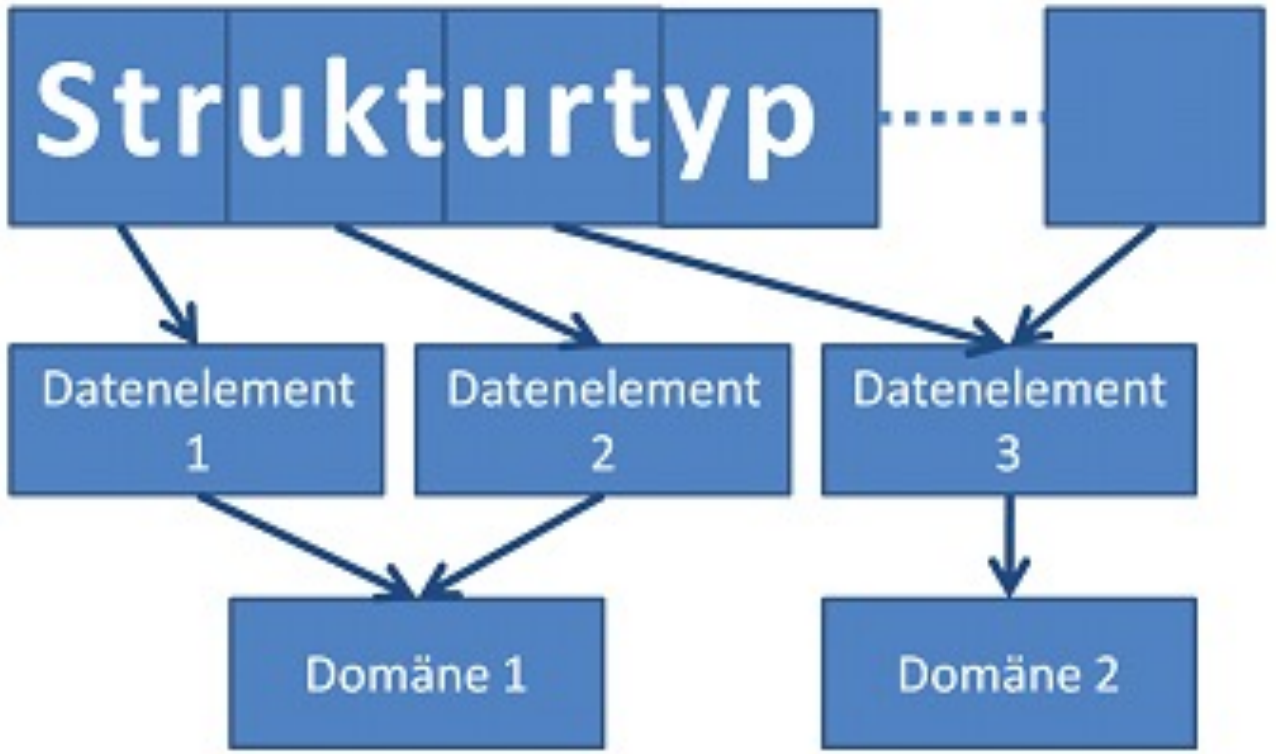
```
* Example 1: Local Types
*&---------------------------------------------------------------------*

* Comment/Uncomment the following lines to change the type of my_var
  TYPES my_type TYPE p LENGTH 3 DECIMALS 2.
*   TYPES my_type TYPE i .
*   TYPES my_type TYPE string.
*   TYPES my_type TYPE n length 10.

* Variable based on local type
  DATA my_variable TYPE my_type.

  out->write( `my_variable (TYPE MY_TYPE)` ).
  out->write(  my_variable ).

* Example 2: Global Types
*&---------------------------------------------------------------------*

* Variable based on global type .
  " Place cursor on variable and press F2 or F3
  DATA airport TYPE /dmo/airport_id VALUE 'FRA'.

  out->write( `airport (TYPE /DMO/AIRPORT_ID )` ).
  out->write(  airport ).

* Example 3: Constants
*&---------------------------------------------------------------------*

  CONSTANTS c_text   TYPE string VALUE `Hello World`.
  CONSTANTS c_number TYPE i      VALUE 12345.

  "Uncomment this line to see syntax error ( VALUE is mandatory)
*  constants c_text2   type string.

  out->write( `c_text (TYPE STRING)` ).
  out->write(  c_text ).
  out->write( '---------' ).

  out->write( `c_number (TYPE I )` ).
  out->write(  c_number ).
  out->write( `---------` ).

* Example 4: Literals
*&---------------------------------------------------------------------*

  out->write( '12345           ' ).   "Text Literal   (Type C)
  out->write( `12345           ` ).   "String Literal (Type STRING)
  out->write( 12345              ).   "Number Literal (Type I)

  "uncomment this line to see syntax error (no number literal with digits)
*   out->write( 12345.67              ).
```

DDIC
Structured Type

Package
/DMO/FLIGHT
Table
/DMO/AGENCY
Structure
/dmo/s_ext_incl_agency

NoR
EMPOWER THE EVOLUTION OF NOW

Two-tier domain approach in the ABAP Dictionary

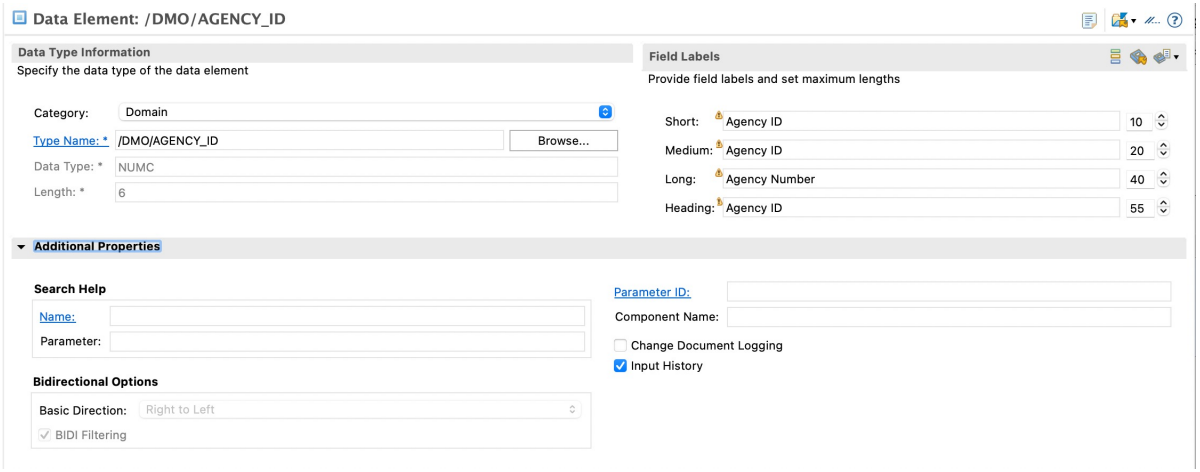


Sematic information

Technical information

DDIC
Data Element

- Motivation: definition of global scalars Types in the ABAP Dictionary (business or also semantic Level)
- Create :

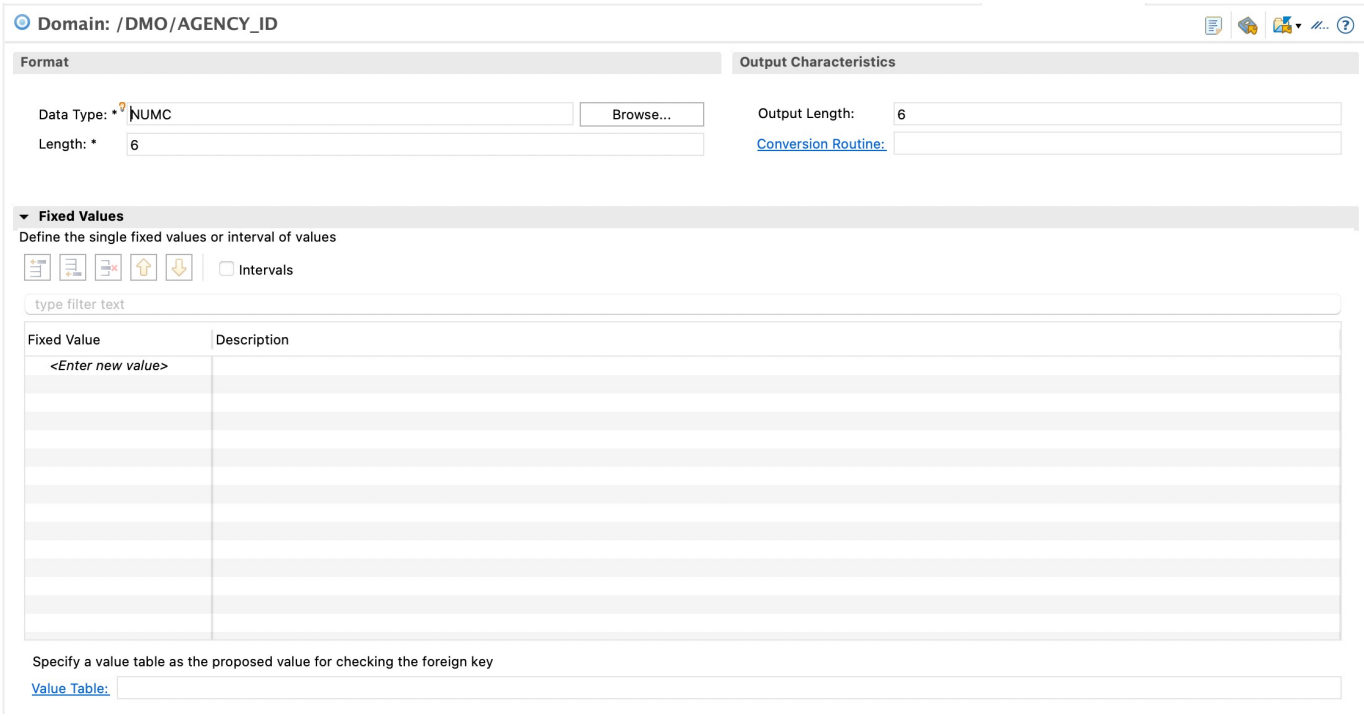


- Usage:

  Typing of variables , interfaces , ...

DDIC
Domain

- Motivation: Serves to define global technical Types with fixed values
- Create :
- Use: Technical Type for Data elements



DDIC
Transparent Table

- Motivation: Used to define a DB table
- Create:
- Indices
  - Primary
  - Secondary
- Technical settings

```
@EndUserText.label : 'Flight Reference Scenario: Managing Agencies'
@AbapCatalog.enhancement.category : #EXTENSIBLE_ANY
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table /dmo/agency {

  key client             : abap.clnt not null;
  key agency_id          : /dmo/agency_id not null;
  name                   : /dmo/agency_name;
  street                 : /dmo/street;
  postal_code            : /dmo/postal_code;
  city                   : /dmo/city;
  country_code           : land1;
  phone_number           : /dmo/phone_number;
  email_address          : /dmo/email_address;
  web_address            : /dmo/web_address;
  attachment             : /dmo/attachment;
  mime_type              : /dmo/mime_type;
  filename               : /dmo/filename;
  local_created_by       : abp_creation_user;
  local_created_at       : abp_creation_tstmpl;
  local_last_changed_by  : abp_locinst_lastchange_user;
  local_last_changed_at  : abp_locinst_lastchange_tstmpl;
  last_changed_at        : abp_lastchange_tstmpl;
  include /dmo/s_ext_incl_agency;

}
```

DDIC
Transparent Table

- Column definition :
  - Columns
  - Part of the primary key ?
  - Initialization
  - Typing

```
@EndUserText.label : 'Flight Reference Scenario: Managing Agencies'
@AbapCatalog.enhancement.category : #EXTENSIBLE_ANY
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table /dmo/agency {

  key client             : abap.clnt not null;
  key agency_id          : /dmo/agency_id not null;
  name                   : /dmo/agency_name;
  street                 : /dmo/street;
  postal_code            : /dmo/postal_code;
  city                   : /dmo/city;
  country_code           : land1;
  phone_number           : /dmo/phone_number;
  email_address          : /dmo/email_address;
  web_address            : /dmo/web_address;
  attachment             : /dmo/attachment;
  mime_type              : /dmo/mime_type;
  filename               : /dmo/filename;
  local_created_by       : abp_creation_user;
  local_created_at       : abp_creation_tstmpl;
  local_last_changed_by  : abp_locinst_lastchange_user;
  local_last_changed_at  : abp_locinst_lastchange_tstmpl;
  last_changed_at        : abp_lastchange_tstmpl;
  include /dmo/s_ext_incl_agency;

}
```

DDIC
Transparent Table

By maintaining a foreign key check **Non-referential integrity** is defined at the database level.

The definition of this check only has an effect if a Dynpro or Web Dynpro field is defined that creates a reference to the foreign key field.

DDIC
Transparent Table

Technical settings & expandability

```
@EndUserText.label : 'Flight Reference Scenario: Managing Agencies'
@AbapCatalog.enhancement.category : #EXTENSIBLE_ANY
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table /dmo/agency {

  key client             : abap.clnt not null;
  key agency_id          : /dmo/agency_id not null;
  name                   : /dmo/agency_name;
  street                 : /dmo/street;
  postal_code            : /dmo/postal_code;
  city                   : /dmo/city;
  country_code           : land1;
  phone_number           : /dmo/phone_number;
  email_address          : /dmo/email_address;
  web_address            : /dmo/web_address;
  attachment             : /dmo/attachment;
  mime_type              : /dmo/mime_type;
  filename               : /dmo/filename;
  local_created_by       : abp_creation_user;
  local_created_at       : abp_creation_tstmpl;
  local_last_changed_by  : abp_locinst_lastchange_user;
  local_last_changed_at  : abp_locinst_lastchange_tstmpl;
  last_changed_at        : abp_lastchange_tstmpl;
  include /dmo/s_ext_incl_agency;

}
```

ABAP

ABAP Properties

"The language"

- is proprietary,
- is typed,
- enables multilingual applications,
- enables SQL access,
- has been extended in an object-oriented manner,
- is platform independent,
- is upwardly compatible.

ABAP Syntax

EMPOWER THE EVOLUTION OF NOW

Allgemeiner Aufbau einer ABAP-Anweisung

Programmbeispiel

```
PARAMETERS pa_num TYPE i.

DATA gv_result TYPE i.

MOVE pa_num TO gv_result.

ADD 1 TO gv_result.

WRITE 'Your input:'.
WRITE pa_num.

NEW-LINE.

WRITE 'Result:    '.
WRITE gv_result.
```

ABAP Instructions

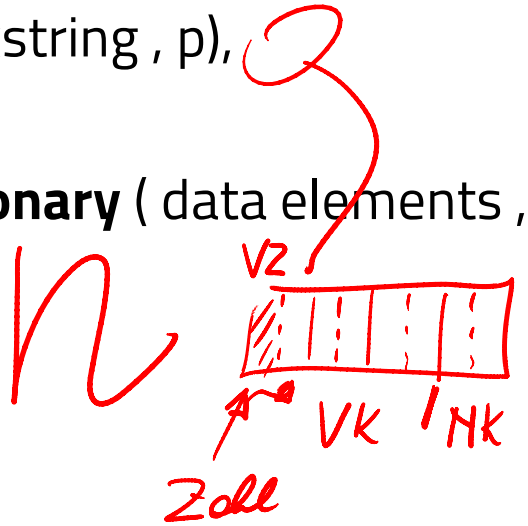
Some important Instructions :

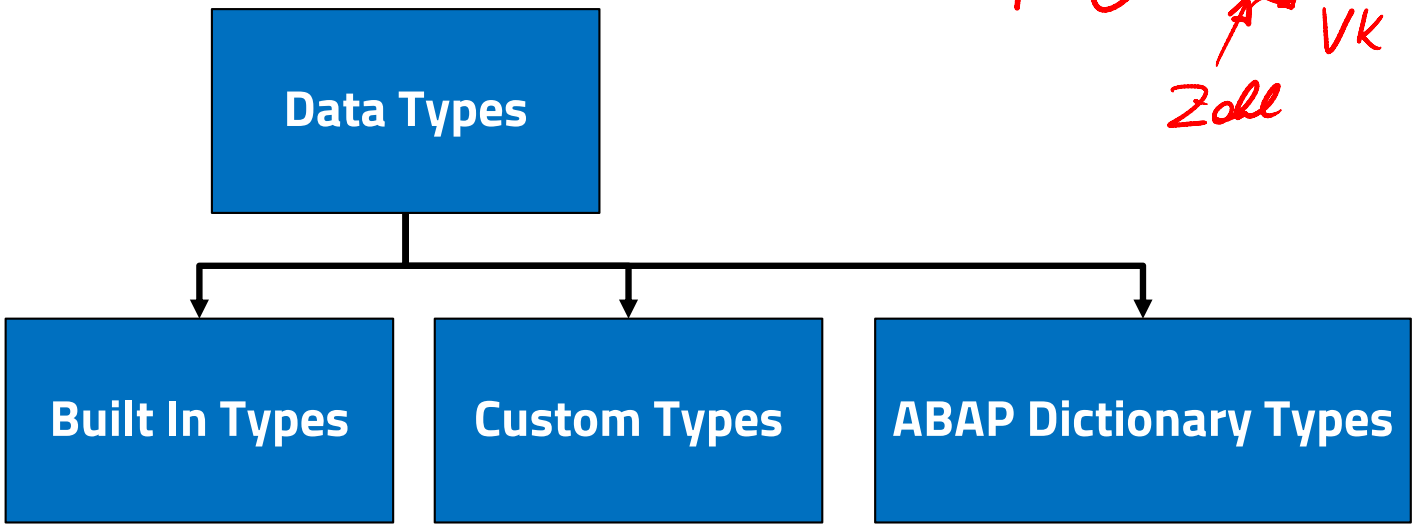
- DATA
- TYPES
- CONSTANTS

ABAP
Data Types

When typing variables we may use:

- **Built In** Types (d, t, i, c, n, f, string, xstring , p),
- **Custom** Types (TYPES) or
- **Global** Types from the **ABAP Dictionary** (data elements , structure types , table types)



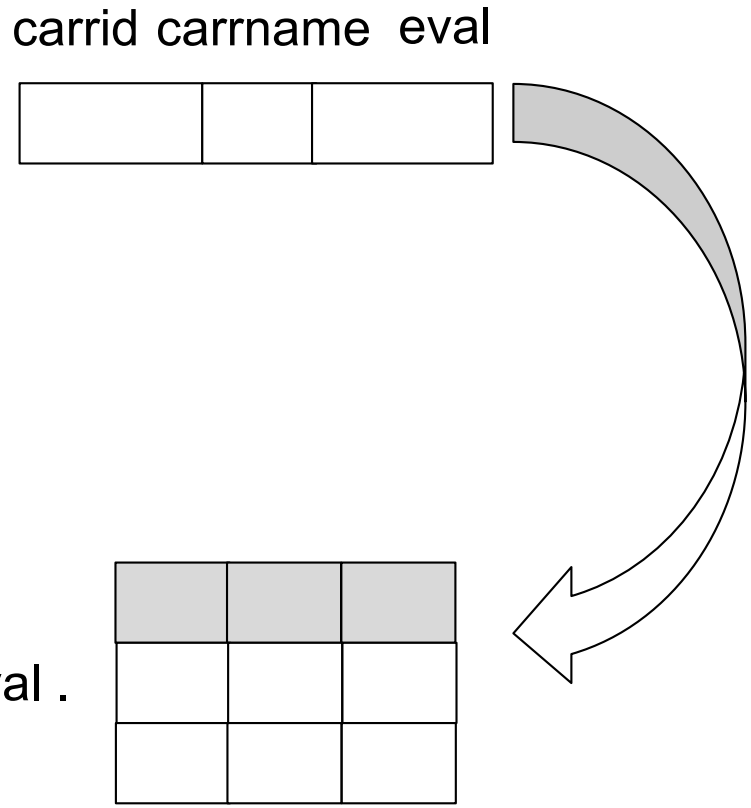


ABAP
Type Complexity

- Elementary Types

- Structured Types

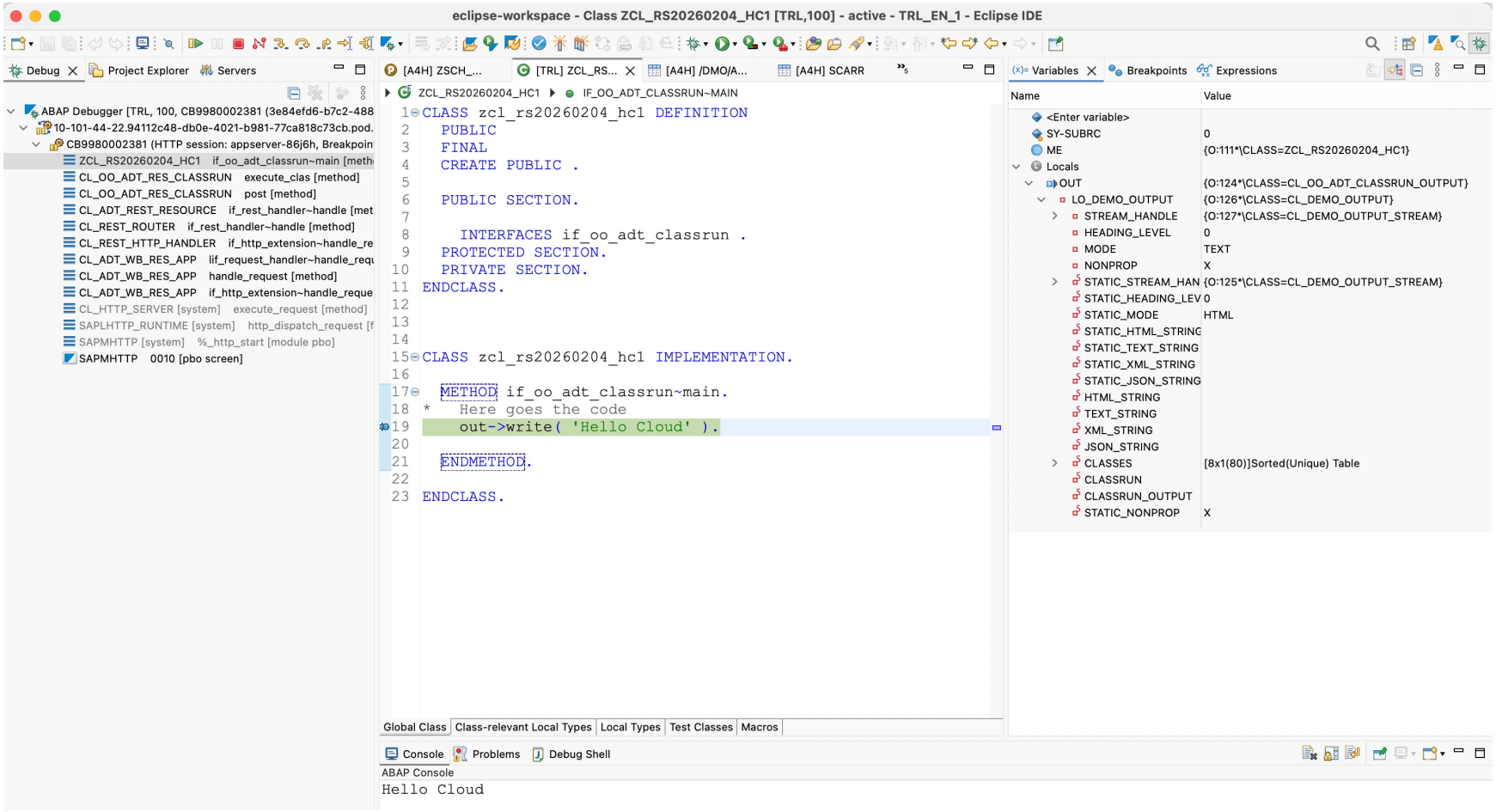
```
* Line type for data buffering to FG and Eval
TYPES: BEGIN OF gst_fg_eval ,
         carrid TYPE scarr-carrid ,
         carrname TYPE scarr-carrname ,
         eval TYPE zscarreval-eval ,
END OF gst_fg_eval .
```



- Table-like Types

```
* Table type (standard table) with row type
         gtt_fg_eval TYPE TABLE OF gst_fg_eval .
```

ABAP Debugging

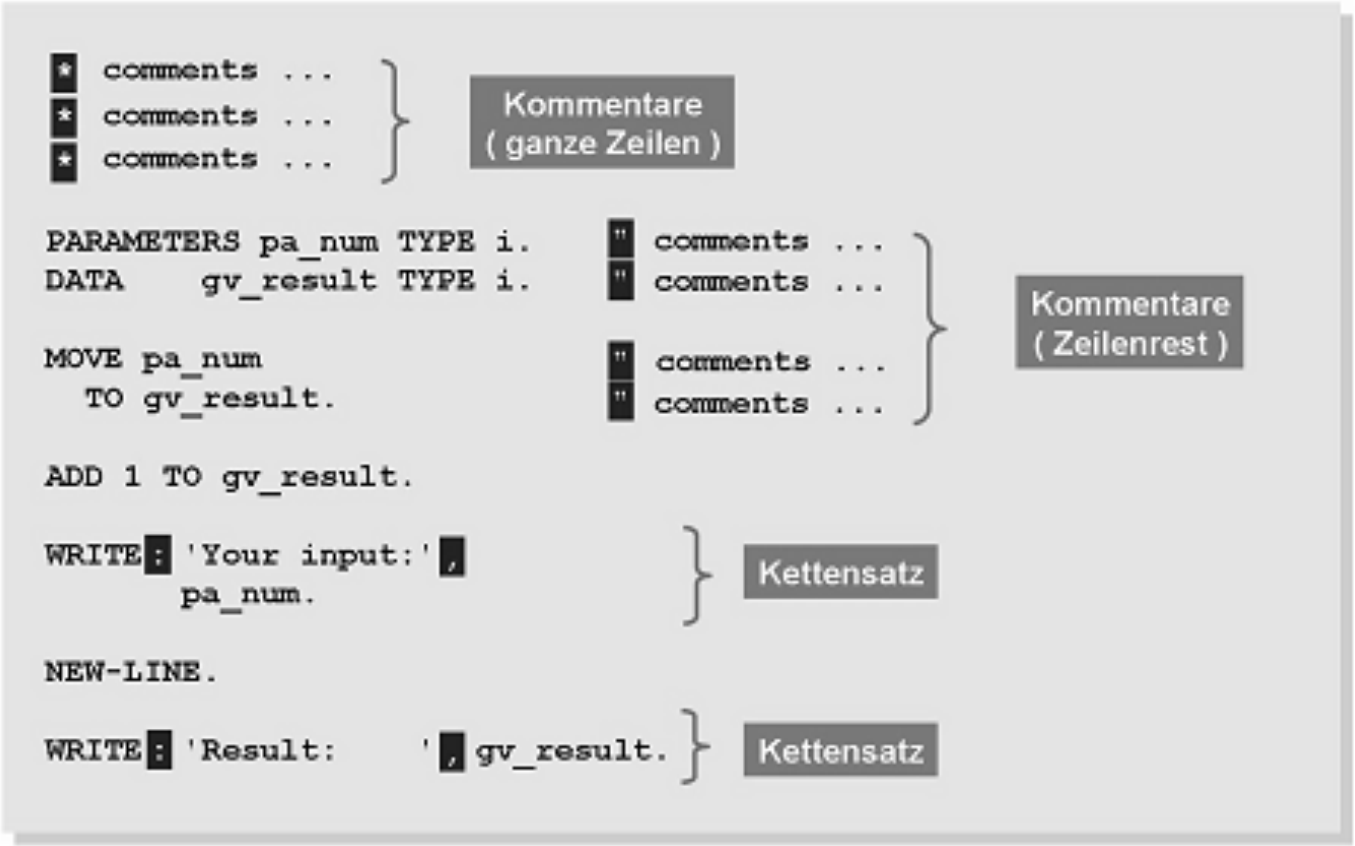


ABAP
Structured Types

- Structure (Dictionary or program local)
- Transparent Table (Dictionary)
- View (Dictionary)
- CDS (Eclipse)

ABAP Literals

ABAP
Examples: Syntax



ABAP
Return Codes (sy-subrc)

- Important System fields are in the ABAP Dict structure SYST defined .
- The most important is sy-subrc , which contains the statement about the exit value the previous operation returns .

```
IF sy-subrc = 0. " Everything is ok
...
ELSE. " There something went wrong
...
ENDIF.
```

```
* Reading the FG from the DB table SCARR
SELECT SINGLE carrname FROM scarr INTO gd_carrname
WHERE carrid = pa_car .
IF sy-subrc <> 0.
   message e002(zws19) with pa_car .
ENDIF.
```

ABAP Modularization

- Level local – local reuse
  - Classic: Subprograms
  - Classic: SAP GUI modules
  - Classic: Event blocks
- Level global – global reuse
  - Classic: Function blocks (function groups)
  - **ABAP classes (ABAP interfaces)**
  - **New: CDS**
  - **New: AMDP = Stored Procedure with OO**

ABAP
Idea Encapsulation

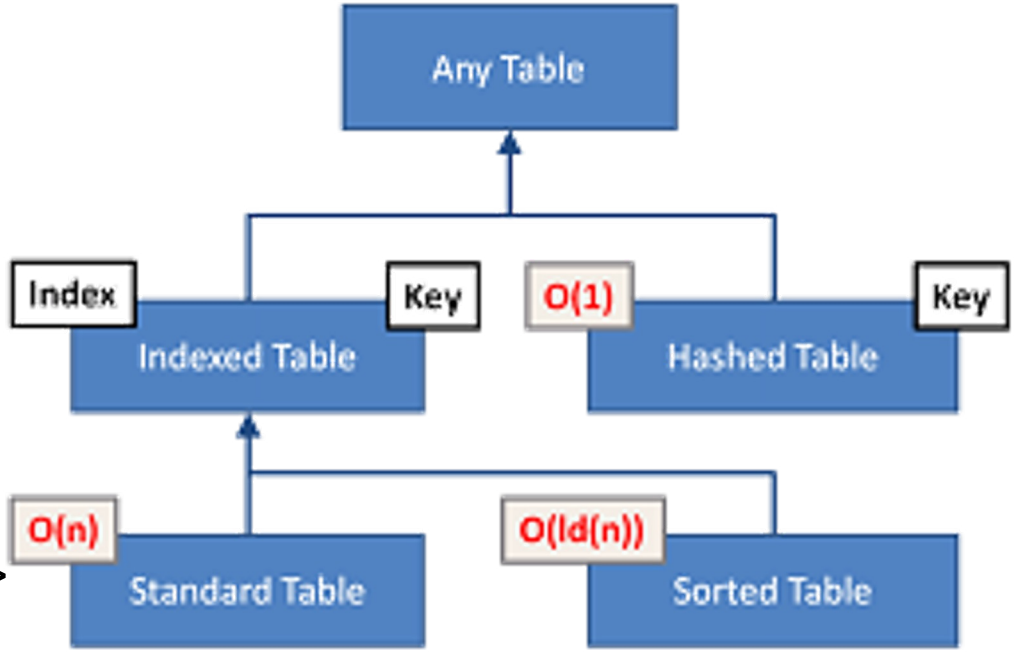
```
*&---------------------------------------------------------------------*
*& READ_EVAL
*&---------------------------------------------------------------------*
* Reading the review
*----------------------------------------------------------------------*
* -->ID_CARRID ID of the FG
* <--ED_EVAL rating
*----------------------------------------------------------------------*
METHOD read_eval            IMPORTING id_carrid TYPE scarr-carrid
                                      EXPORTING ed_eval TYPE zscarreval-eval .

SELECT SINGLE eval FROM zscarreval INTO ed_eval
WHERE carrid = id_carrid .

ENDMETHOD. " READ_EVAL
```

ABAP
DDIC: Structure

- Motivation: definition of global structured type (comparable : Record or one-dim array)
- Create :
- Usage: Typing of variables , interfaces , table types , ...
- Access to components : hyphen semantics
- Copying content : MOVE-CORRESPONDING

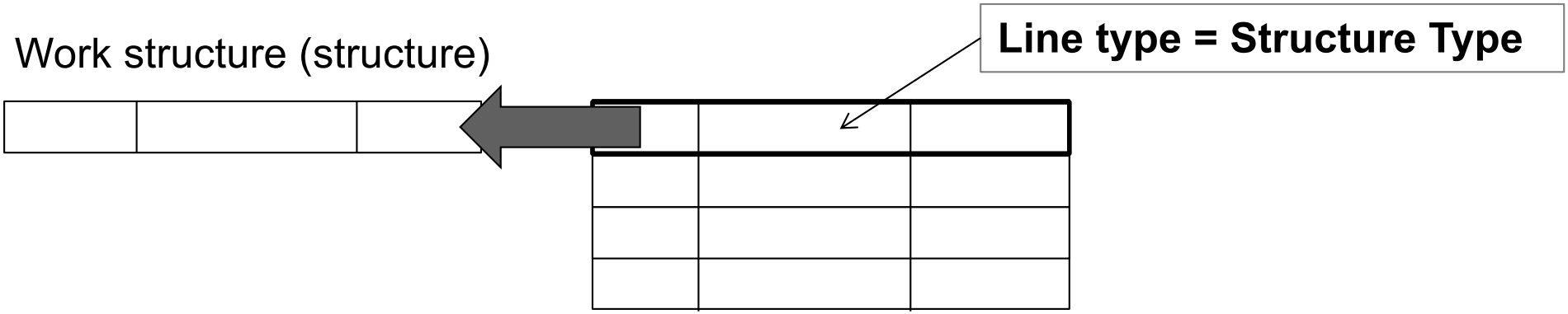
ABAP
DDIC: Table Type

- Motivation: Defines the global type of an internal table
- Create : TA SE11
- Definition of one internal Table
- Important Instructions :
  - Sort < table > BY < column > [ascending|decending]
  - APPEND <row> TO <table> .
  - DELETE <row> FROM <table> .



ABAP
Internal Table

Is in main memory!



Type of table
- Standard
- Sorted
- Hashed

Key
- Components
- Sequence
- Uniqueness

Output table

```
*&---------------------------------------------------------------------*
*& WRITE_FG_EVAL
*&---------------------------------------------------------------------*
* Output the table to the list
*----------------------------------------------------------------------*
* -->IT_FG_EVAL Table of FV and eval
*----------------------------------------------------------------------*
METHOD write_fg_eval IMPORTING it_fg_eval TYPE gtt_fg_eval .

DATA: ls_fg_eval LIKE LINE OF it_fg_eval .

LOOP AT it_fg_eval INTO ls_fg_eval .
WRITE: / ls_fg_eval-carrid ,
         ls_fg_eval-carrname ,
         ls_fg_eval-eval .
ENDLOOP.

ENDMETHOD. " WRITE_FG_EVAL
```

Read into internal table

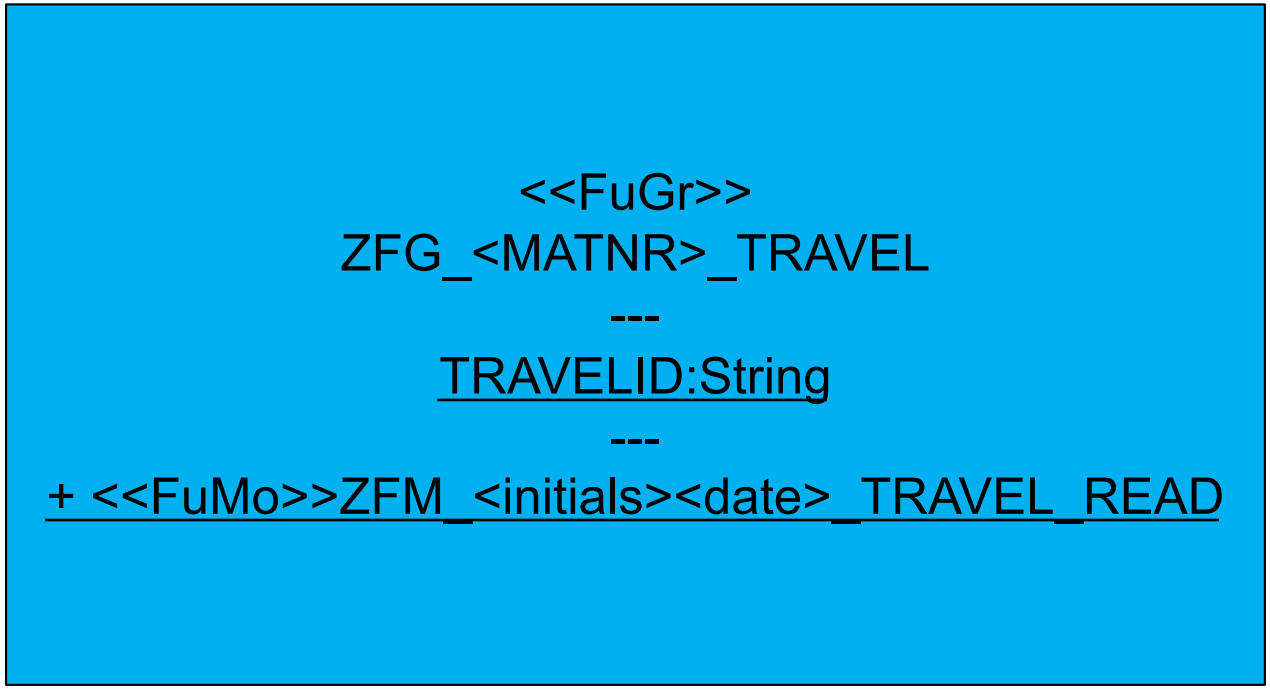
```
*&---------------------------------------------------- ----------------------*
*& READ_FG_EVAL
*&---------------------------------------------------- ----------------------*
* Determine the FG for an eval
*---------------------------------------------------- ----------------------*
* --> ID_EVAL    Eval
* <--ET_FG_EVAL FG to eval
*---------------------------------------------------- ----------------------*
METHOD read_fg_eval IMPORTING id_eval TYPE zscarreval-eval
EXPORTING et_fg_eval TYPE gtt_fg_eval .

DATA: ld_carrid TYPE scarr-carrid ,
      ls_fg_eval LIKE LINE OF et_fg_eval ,
      ld_carrname TYPE scarr-carrname .

* Bulk operation with SELECT
SELECT carrid FROM zscarreval INTO ld_carrid
WHERE eval = id_eval .
* Fill auxiliary structure
   ls_fg_eval-eval = id_eval .
   ls_fg_eval-carrid = ld_carrid .
* Reading the fg from the db tabel scarr
me->read_carrname
exporting
              ld_carrid
importing
              ld_carrname .

   ls_fg_eval-carrname = ld_carrname .
* Create entry in the internal table
APPEND ls_fg_eval TO et_fg_eval .
ENDSELECT.

ENDMETHOD. " READ_FG_EVAL
```

ABAP console

Modularization

Motivation FuGr/FuBa

Function Group / Function Module



ABAP

Function Group / Function Module

- Definition
  - Interface
    - IMPORTING
    - EXPORTING
    - CHANGING
    - EXCEPTIONS
- Use Auto completion
  - CALL FUNCTION `<name>`

```
FUNCTION Z
IMPORTING
              VALUE(IM_P1) TYPE type1 OPTIONAL
              VALUE(IM_P2) TYPE type2 DEFAULT def_value
            EXPORTING
              EX_P1        TYPE REF TO STRING
            CHANGING
              CH_1         TYPE ANY
            TABLES
              TAB_P1       LIKE structure_name
              TAB_P2       TYPE tab_type
            RAISING
              CX_SY_ZERODIVIDE
              RESUMABLE(CX_SY_ASSIGN_CAST_ERROR).
ENDFUNCTION.
```

Function Module
Does it do anything useful?

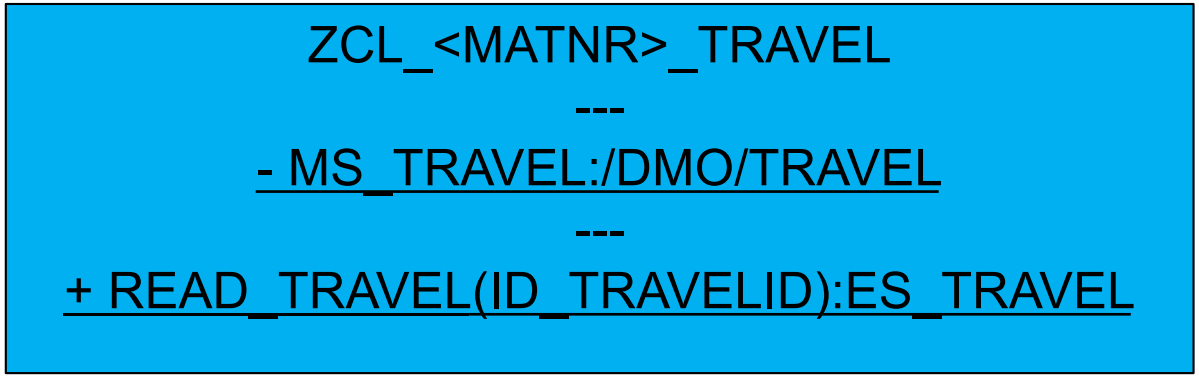
```
FUNCTION zfm_rs20250202_travel_read
  IMPORTING
    VALUE(id_key) TYPE i
  EXPORTING
    ed_key TYPE i.

  ed_key = id_key.

ENDFUNCTION.
```

Function Module

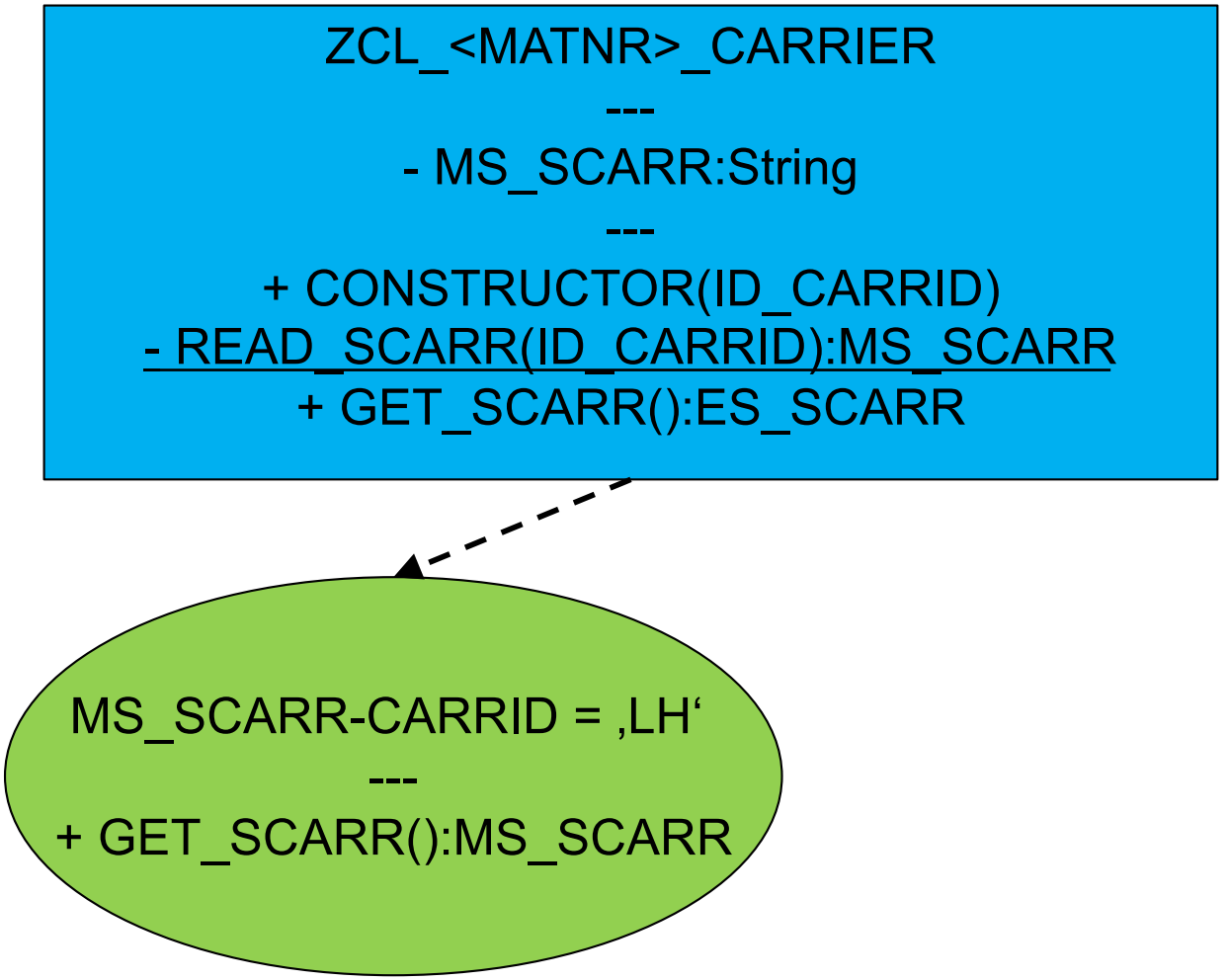
```
FUNCTION z8820038_ws19_fg_r_eval.
*"----------------------------------------------- ----------------------
*"*" Local Interface:
*" IMPORTING
*" REFERENCE(ID_CARRID) TYPE SCARR-CARRID
*" EXPORTING
*" REFERENCE(ED_EVAL) TYPE ZSCARREVAL-EVAL
*"----------------------------------------------- ----------------------


SELECT SINGLE eval FROM zscarreval INTO ed_eval
WHERE carrid = id_carrid .


END FUNCTION.
```

ABAP Object Oriented (OO)

Motivation OO Classes



Motivation OO Classes



ABAP OO
Motivation

- For what? What for?
- Procedural vs. Object Orientation
- Properties of the OO
- Level of coverage in ABAP OO
  - Inheritance
  - Polymorphism
  - Encapsulation
- Separate definition and implementation

ABAP OO

NoR
EMPOWER THE EVOLUTION OF NOW

Modeling with the Unified Modeling Language (UML)

- Class diagram
  - Class
  - Attribute
  - Method
  - Association
  - Cardinality
- Sequence diagram

ABAP OO

Class and Object

- Dynamic vs. Static Type
- Visibilities
- Attribute
- Method
- Event
- Construction plan / blueprint

ABAP OO

Types of ABAP classes

- **Normal Class**
- **Exception class**
- ABAP Units Test class
- Persistent object

Properties of ABAP classes

- Abstract
- Final

ABAP OO

Attribute

- Static/Class
- Instance/Object
- Visibilities/Public, Protected, Private

ABAP OO

Definition **Method**

- Static/Class
- Instance/Object
- Visibilities/
  Public,Protected,Private

ABAP OO

Testing Classes

- ABAP units

ABAP OO

States of objects

- Create (Constructor)

```
IRT.
* Objekt instanziieren
      CREATE OBJECT go_fg
        EXPORTING
          id_carrid = pa_car.
* Fehlerbehandlung
```

- Terminate (Grabage Collector - GC)
  - Using CLEAR the object reference initialized become and thereby the GC collects the object and gives the resources free .
  - In principle, if a object is not reachable anymore , then the GC frees those affected resources

ABAP OO

Calling Methods:

- Static
- Instance
- Functional

```
      cl_salv_table=>factory(
*    EXPORTING
*      list_display   = IF_SALV_C_BOOL_SAP=>FALSE
*      r_container    =
*      container_name =
       IMPORTING
         r_salv_table   = go_alv
       CHANGING
         t_table        = gt_fg_eval
            ).
```

```
go_alv->display( ).
```

```
gd_carrname_single = go_fg->get_carrname( ).
```

Use of attributes

ABAP
Exceptions

ABAP OO
Exceptions - History

1. Classic
IF sy-subrc = 0.
...
ENDIF.

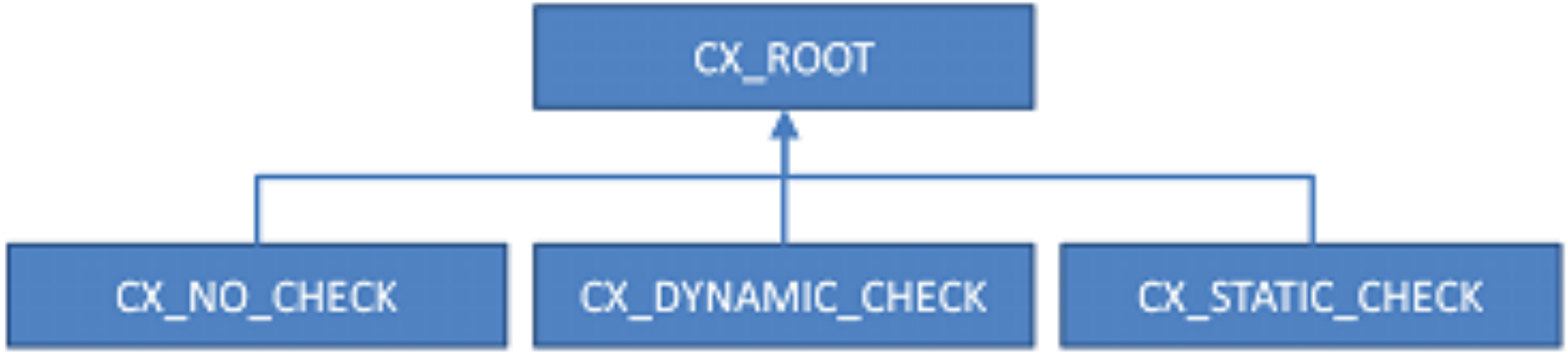
2. Intermediate solution on the way to Object Orientation
CATCH SYSTEM EXCEPTION.
...
ENDCATCH. + sy-subrc Evaluation

3. Object Oriented exception concept
TRY.
...
CATCH.
...
ENDTRY.

ABAP OO
OO Exceptions

Inheritance hierarchy for exceptions

CX_ROOT



ABAP OO
OO Exceptions

Create exception

Exception text

ABAP OO
OO Exceptions

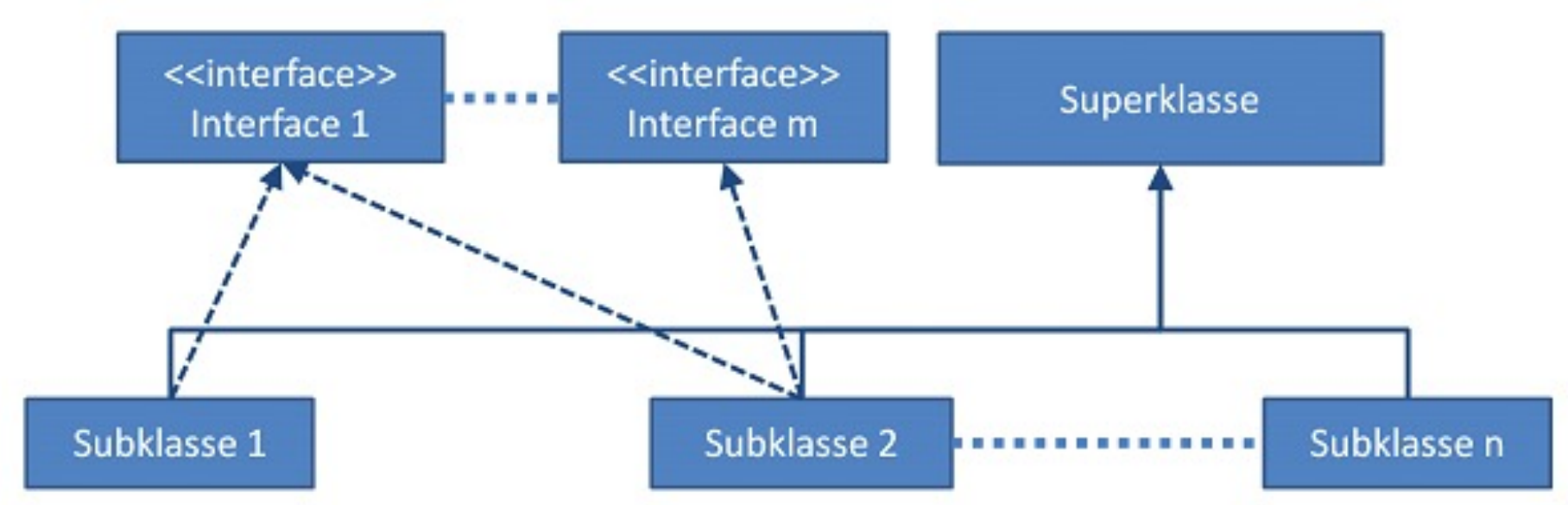
Trigger exception

RAISE EXCEPTION

ABAP OO

Inheritance

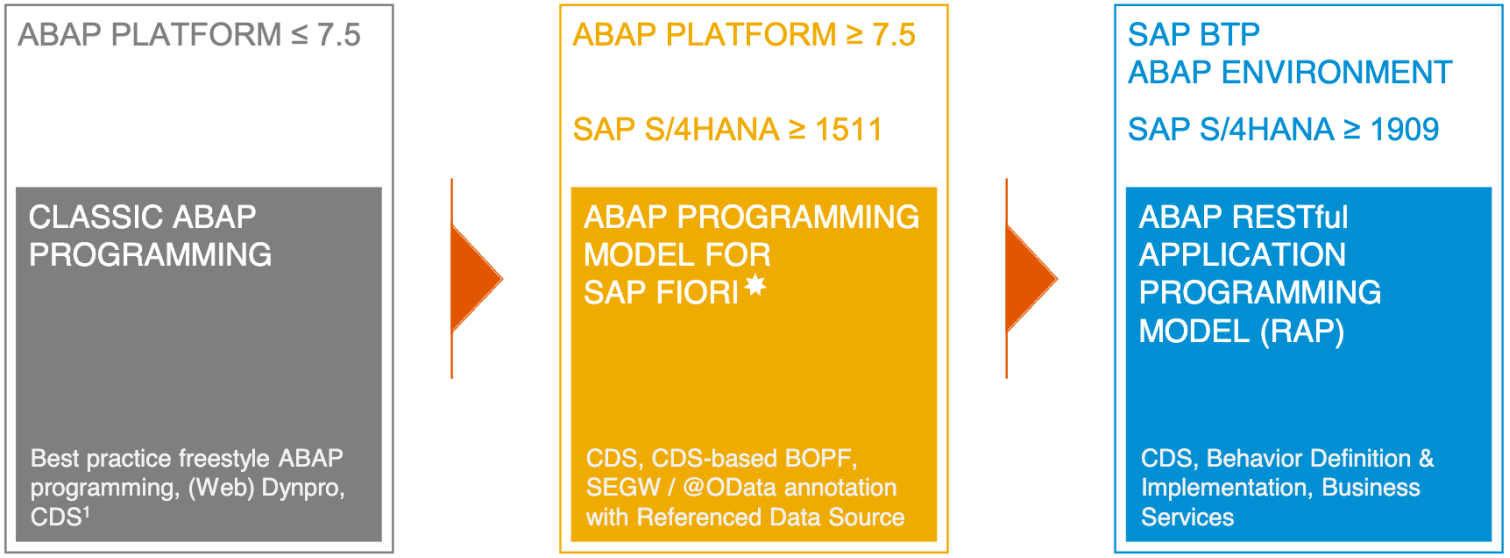
- Redefinition
- Casting (Up/Down)



ARCHITECTURE

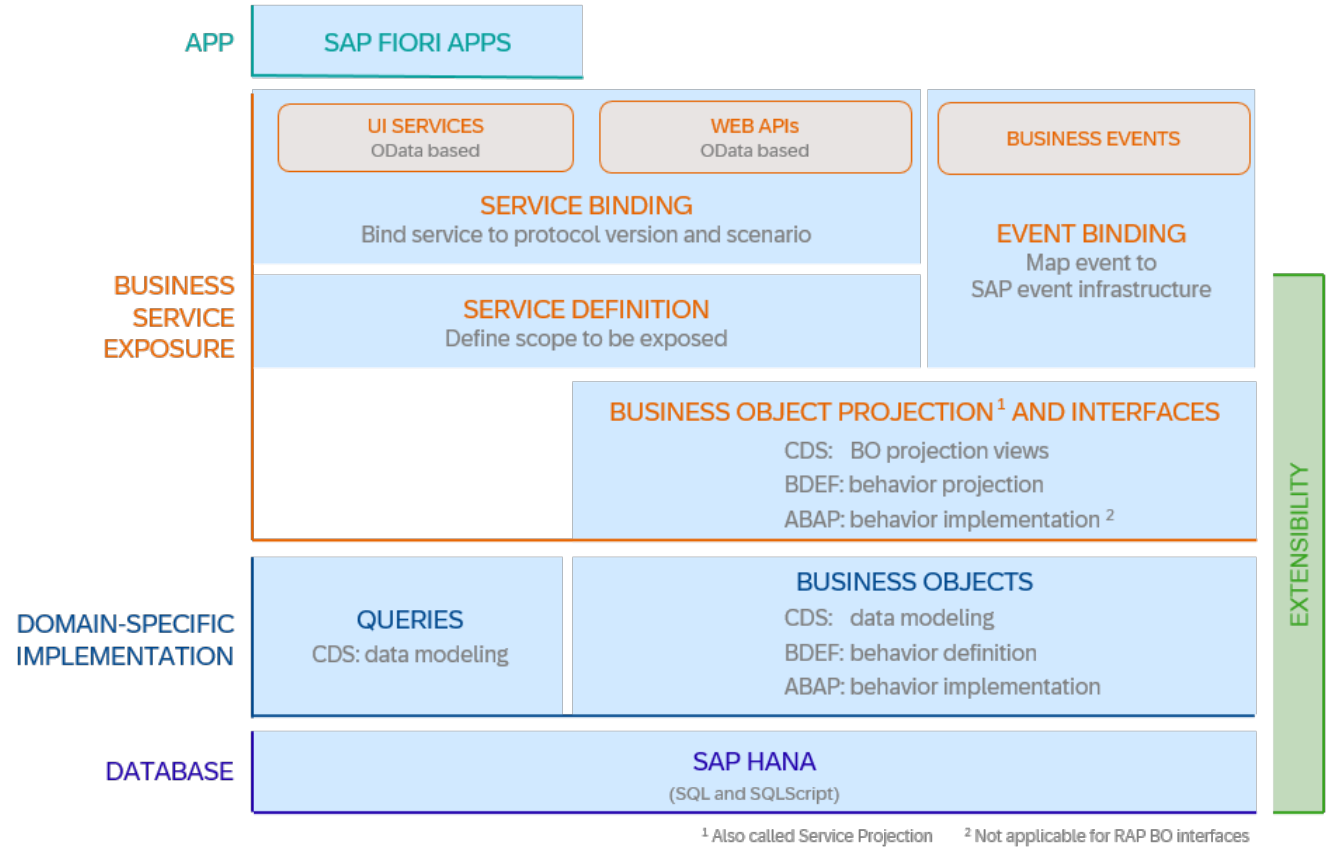
Evolution

Evolution of the ABAP programming model

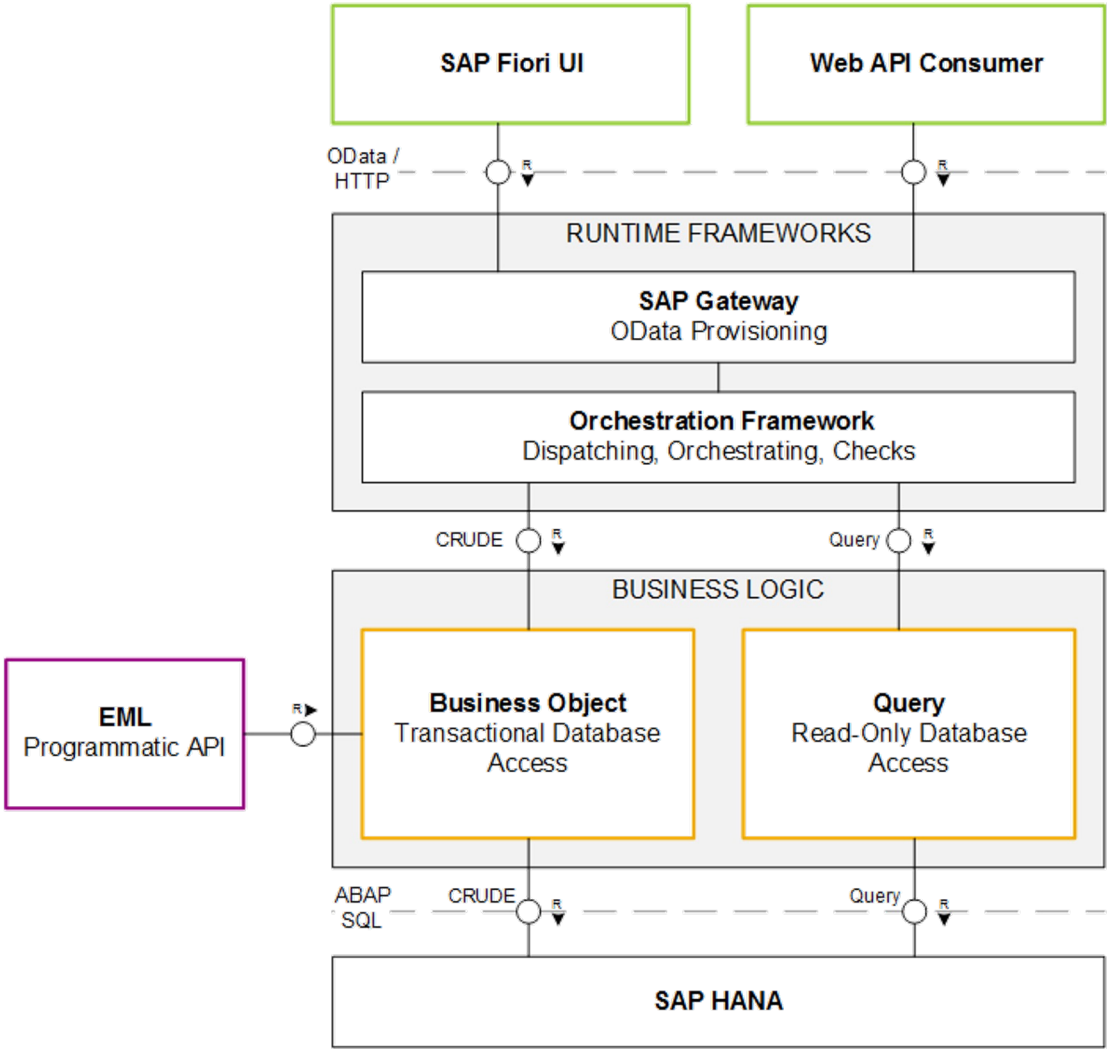


[1] starting with release 7.4 SPS05

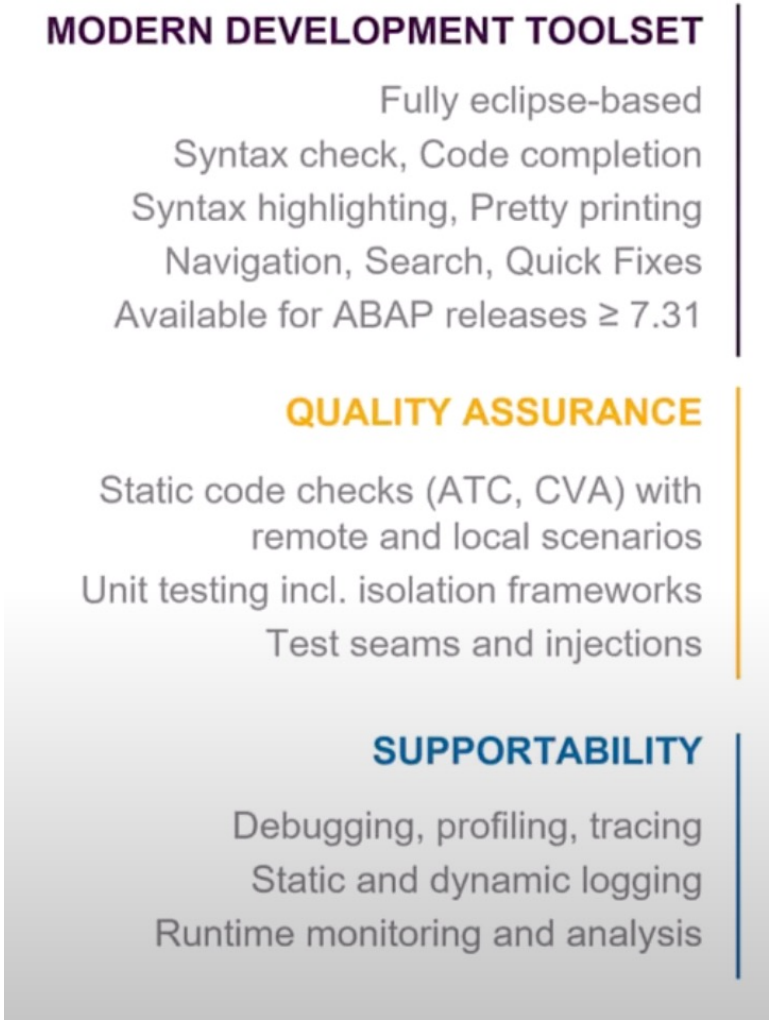
RAP Big Picture



RAP Runtime


SAP Fiori UI
Web API Consumer
OData / HTTP
RUNTIME FRAMEWORKS
SAP Gateway
OData Provisioning
Orchestration Framework
Dispatching, Orchestrating, Checks
CRUDE
Query
BUSINESS LOGIC
EML
Programmatic API
Business Object
Transactional Database Access
Query
Read-Only Database Access
ABAP SQL
CRUDE
Query
SAP HANA

Eclipse


MODERN DEVELOPMENT TOOLSET
Fully eclipse-based
Syntax check, Code completion
Syntax highlighting, Pretty printing
Navigation, Search, Quick Fixes
Available for ABAP releases ≥ 7.31
QUALITY ASSURANCE
Static code checks (ATC, CVA) with remote and local scenarios
Unit testing incl. isolation frameworks
Test seams and injections
SUPPORTABILITY
Debugging, profiling, tracing
Static and dynamic logging
Runtime monitoring and analysis

OData

Technology Base

OData V2 (O2)

- Classic standard (around 2010), **built into SAP Gateway**.
- Typically implemented in **SEGW (Service Builder)** or via **CDS → OData Publish (@OData.publish: true)**.
- **Protocol:** OData V2 (Atom/XML, JSON).
- **Framework:** SAP Gateway Runtime.

OData V4 (O4)

- Newer standard (from 2014), broadly supported in SAP only **from ABAP Platform 7.55+**.
- Implemented only via **RAP (RESTful ABAP Programming Model)** → SEGW is no longer used.
- **Protocol:** OData V4 (JSON only, more compact, modern).
- **Framework:** RAP BO (Business Object) + OData V4 Framework.

OData

Data Model & Metadata

NoR
EMPOWER THE EVOLUTION OF NOW

- **O2:** EDMX metadata in V2 format. Supports EntitySets, Navigation, Associations.
- **O4:** EDMX in V4 format. Supports *Singletons, Functions, Actions, Complex Types*, richer navigation.

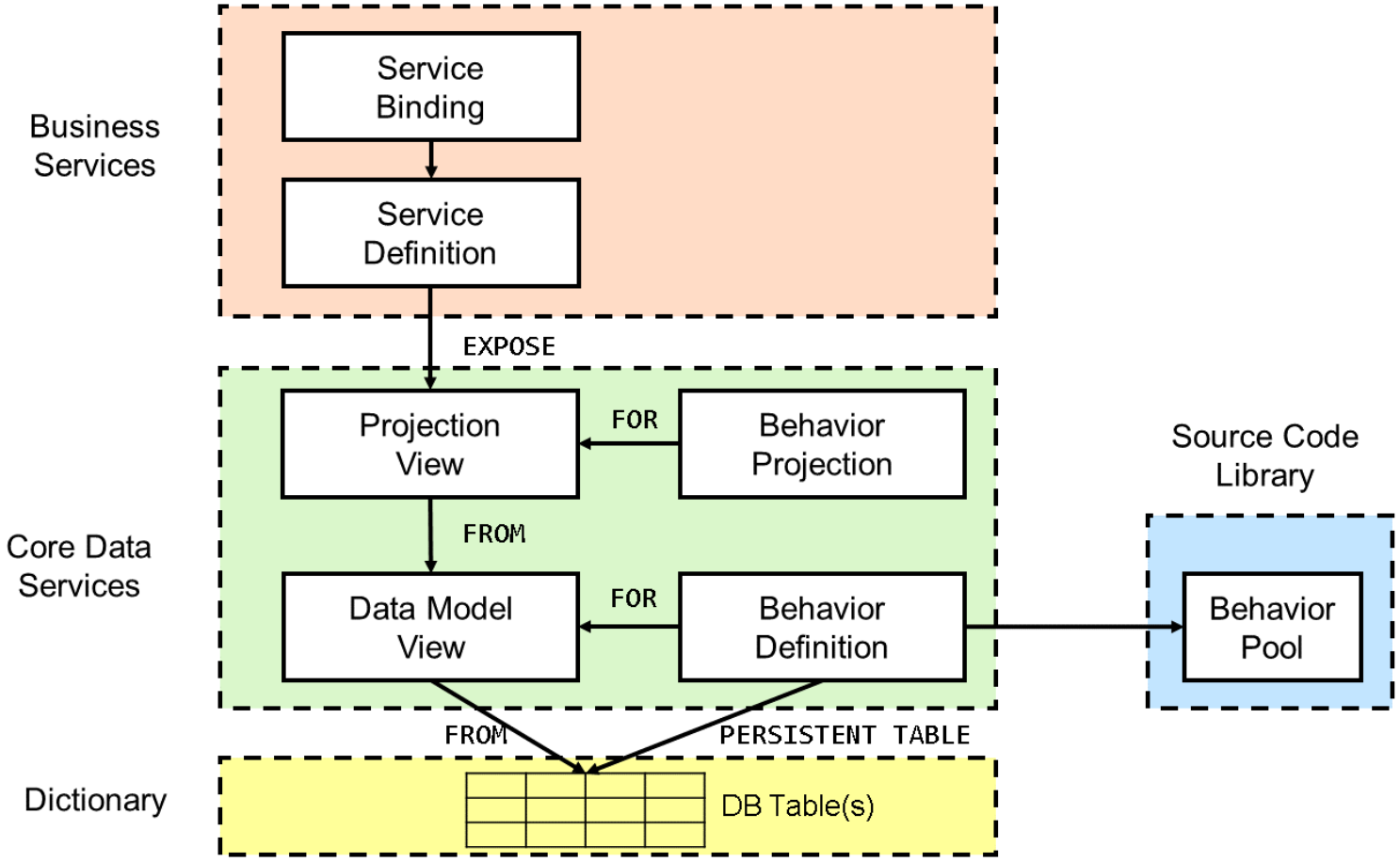
OData

Feature Comparison

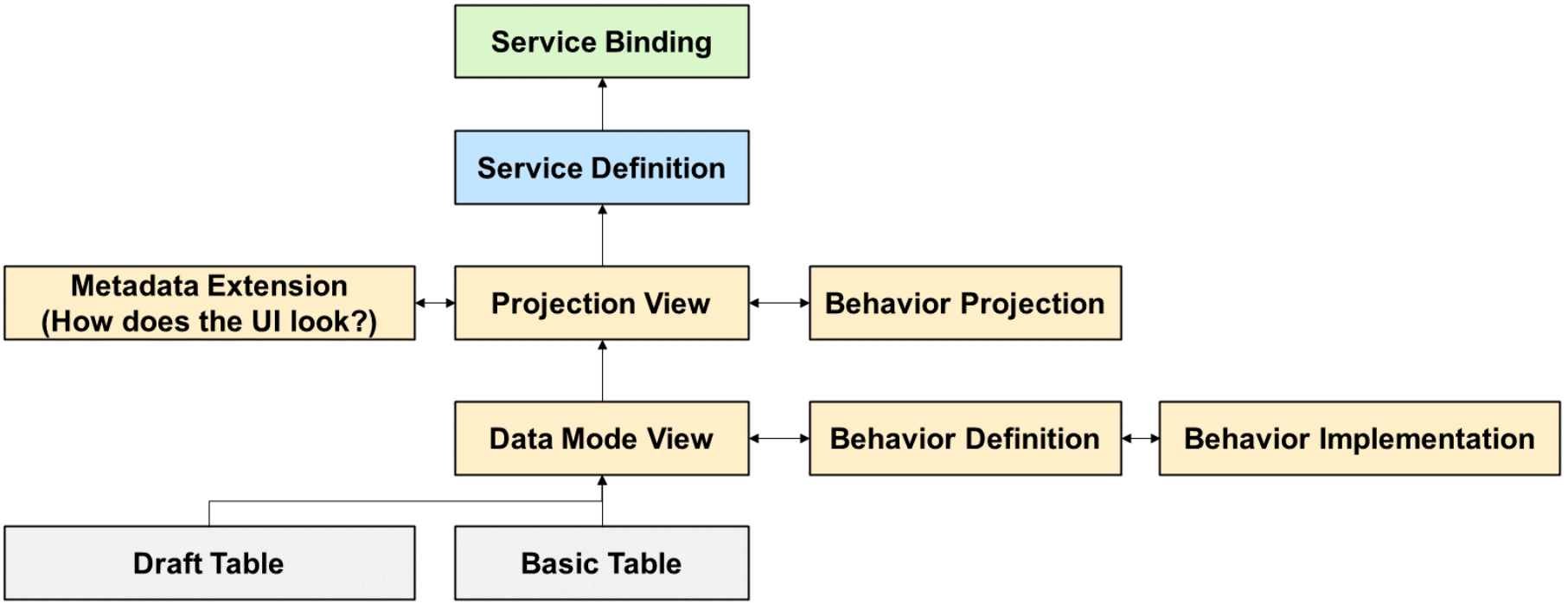
Area	OData V2 (O2)	OData V4 (O4)
Development	SEGW, MPC/DPC methods, CDS Publish	RAP BO Definition (Behavior, Service Definition, Service Binding)
Formats	XML + JSON	JSON only
Batch Support	Yes ($batch requests)	Yes, cleaner and more efficient
Deep Insert/Update	Complex, partly custom	Fully supported in RAP
Actions/Functions	Limited	Fully standardized
Annotations	Basic	Rich (UI, semantic, CAP-like)
Draft Handling	Only via custom coding	Native in RAP
Value Helps (F4)	Manual implementation	Provided via CDS annotations
Query Options	$filter, $expand, $select, $orderby	Extended: $count, $search, $compute, richer expressions
Delta Handling	Token-based but complex to implement	Native Delta Queries in RAP
Optimization	Limited paging and performance control	Server-driven paging, optimized query planning

APPLICATION

Developing a RAP application consists of the following main steps



Generate additional objects



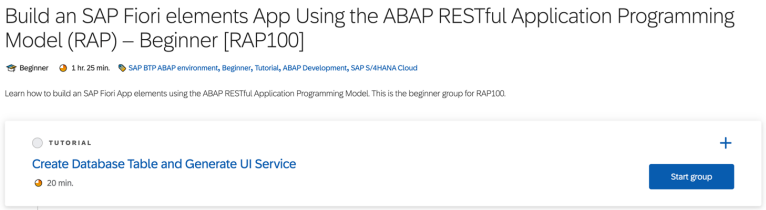
Create a RAP Application

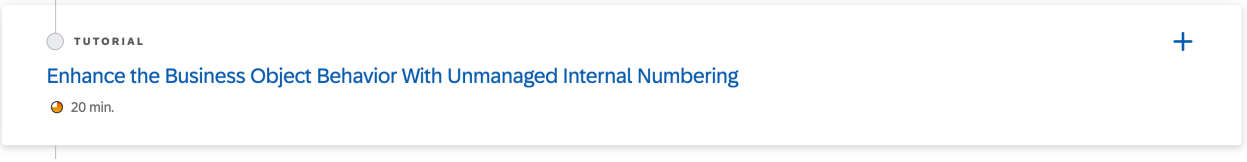
https://developers.sap.com/group.abap-build-fiori-element-rap.html

Package: `$<Initials><Date>_ANALYSIS`

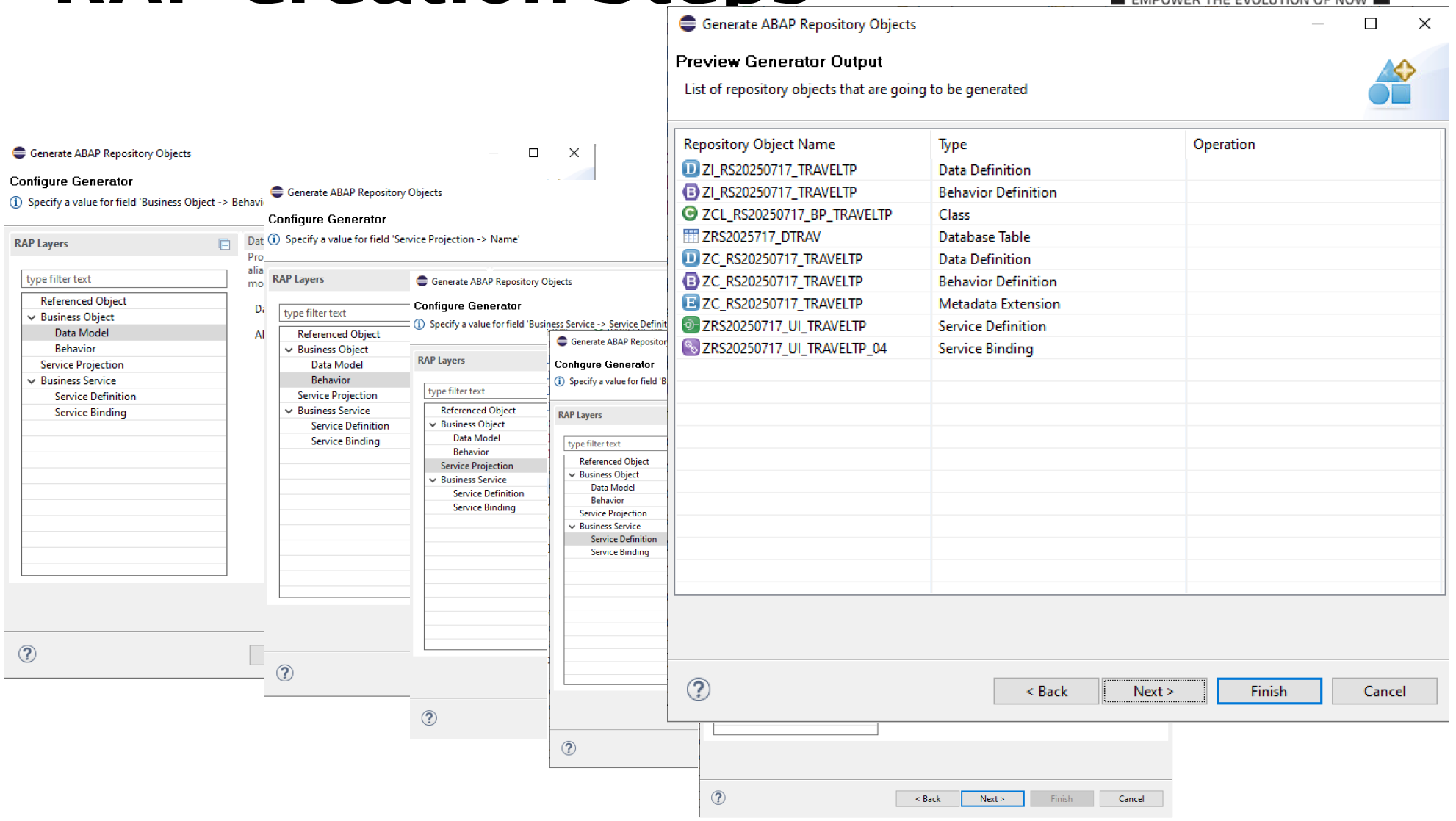
Table: `Z<Initials><Date>_TRAV (max 16 char!)`

RAP Application:

1. 

2. 

RAP Creation Steps



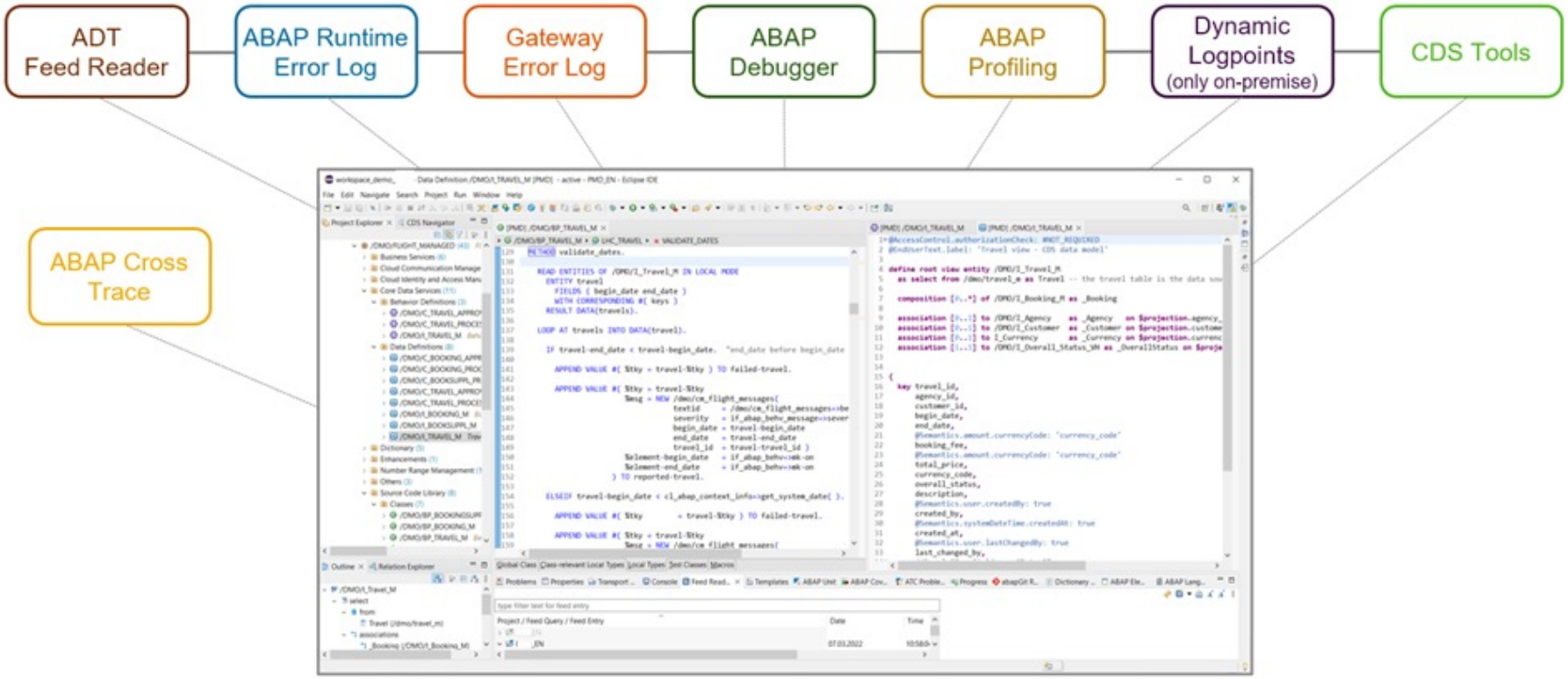
Code

EARLYNUMBERING_CREATE

```
METHOD earlynumbering_create.
DATA:
entity TYPE STRUCTURE FOR CREATE ZRS00_R_TravelTP,
travel_id_max TYPE /dmo/travel_id,
" change to abap_false if you get the ABAP Runtime error 'BEHAVIOR_ILLEGAL_STATEMENT'
use_number_range TYPE abap_bool VALUE abap_false.

"raise EXCEPTION TYPE cx_abap_behv_runtime_error.

raise EXCEPTION type /iwbep/cx_v4_conversion.

"Ensure Travel ID is not set yet (idempotent)- must be checked when BO is draft-enabled
LOOP AT entities INTO entity WHERE TravelID IS NOT INITIAL.
APPEND CORRESPONDING #( entity ) TO mapped-travel.
ENDLOOP.

DATA(entities_wo_travelid) = entities.

"Remove the entries with an existing Travel ID

DELETE entities_wo_travelid WHERE TravelID IS NOT INITIAL.

"determine the first free travel ID without number range

"Get max travel ID from active table

SELECT SINGLE FROM zrs00_atrav FIELDS MAX( travel_id ) AS travelID INTO @travel_id_max.

"Get max travel ID from draft table

SELECT SINGLE FROM zrs00_datrav FIELDS MAX( travelid ) INTO @DATA(max_travelid_draft).

IF max_travelid_draft > travel_id_max.

travel_id_max = max_travelid_draft.

ENDIF.

"Set Travel ID for new instances w/o ID

LOOP AT entities_wo_travelid INTO entity.

travel_id_max += 1.

entity-TravelID = travel_id_max.

APPEND VALUE #( %cid = entity-%cid

%key = entity-%key

%is_draft = entity-%is_draft

) TO mapped-travel.

ENDLOOP.

ENDMETHOD.
```

TOOLS

Troubleshooting Tools



Naming Conventions

l Local data object (DO)*
*g * Global DO*
i Importing parameters*
*e * Exporting parameters*
*r * Returning parameters*
c Changing parameters*
+d_* elementary DO
+s_* structured DO
+t_* table-like DO
+o_* Object/ instance reference variable
+ r _* data reference variable
< fs _*> Field symbol
pa _ Parameters*
so_* Select-Options

Literature List

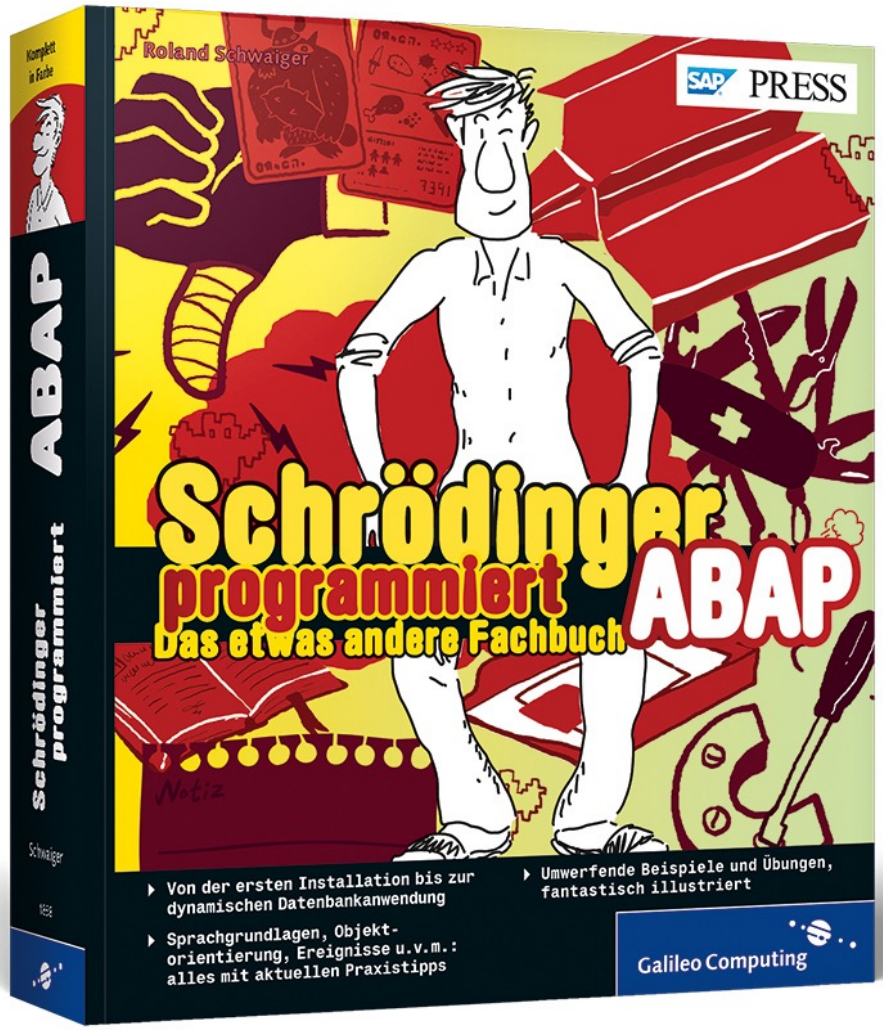
www.wikipedia.org

developers.sap.com

help.sap.com

nor.gmbh, rolandschwaiger.at

Recommendation



First Assignment

Register for BTP Trial

- Follow instructions on https://www.sap.com/products/technology-platform/trial.html
- Send screen shot of your sub account to my e-mail address
- Due to 19.12.2025
- Precondition for participating the course

Second Assignment

Instantiate ABAP Environment on SAP BTP, Install Eclipse with ADT and Create an ABAP Cloud Project

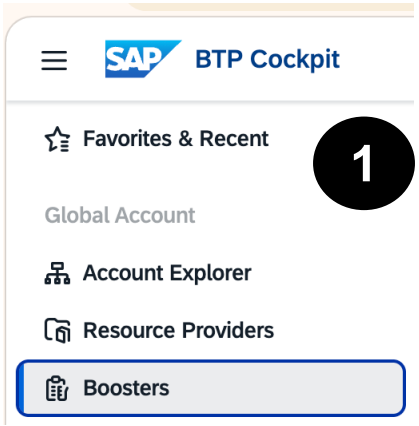
- Follow instructions on
  - Create ABAP Environment and save key:
    - See page 10
  - Install Eclipse: https://developers.sap.com/tutorials/abap-install-adt..html
  - Create an ABAP cloud project in Eclipse:
    - https://developers.sap.com/tutorials/abap-environment-create-abap-cloud-project..html
    - See page 11
- Send a screen shot of your cloud project to my e-mail address, looks like 

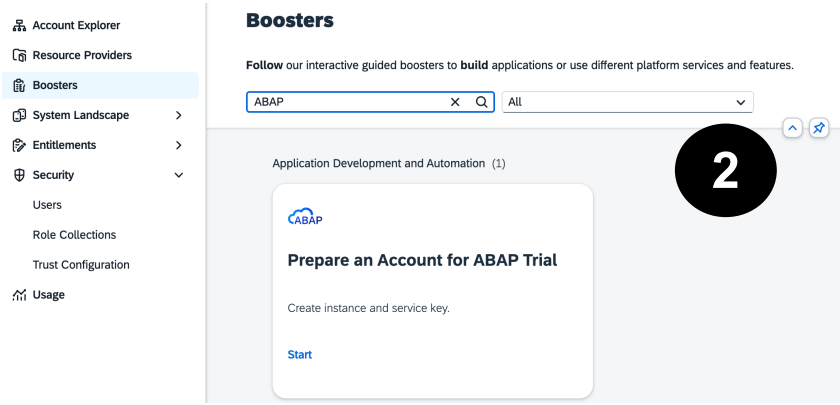
- Due to 31.01.2026
- Precondition for participating the course

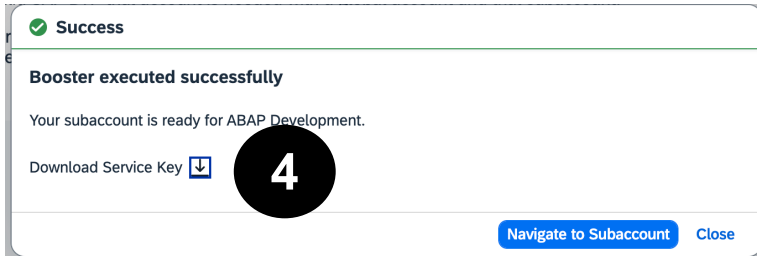
Second Assignment

Instantiate ABAP Environment on SAP BTP

1. Select Booster
2. Search “Prepare an Account for ABAP Trial”
3. Select Tile, Press Start and wait for the Booster to finish
4. Download Service Key
5. Navigate to subaccount







Second Assignment

Create Cloud Project – Select URL

1. Select ABAP Cloud Project
2. Click the Extract link
3. Click the **Import** button on the “Extract Service Instance URL” screen. Select the key file you have downloaded before.
4. Click the **Copy to Clipboard button** and **Close**
5. Paste the URL in the “ABAP Service Instance URL” input field
6. Choose Next and Choose the button “Open Logon Page in Browser” (if this is not working then choose the second button and copy the url to the clipboard and paste it in your browser)