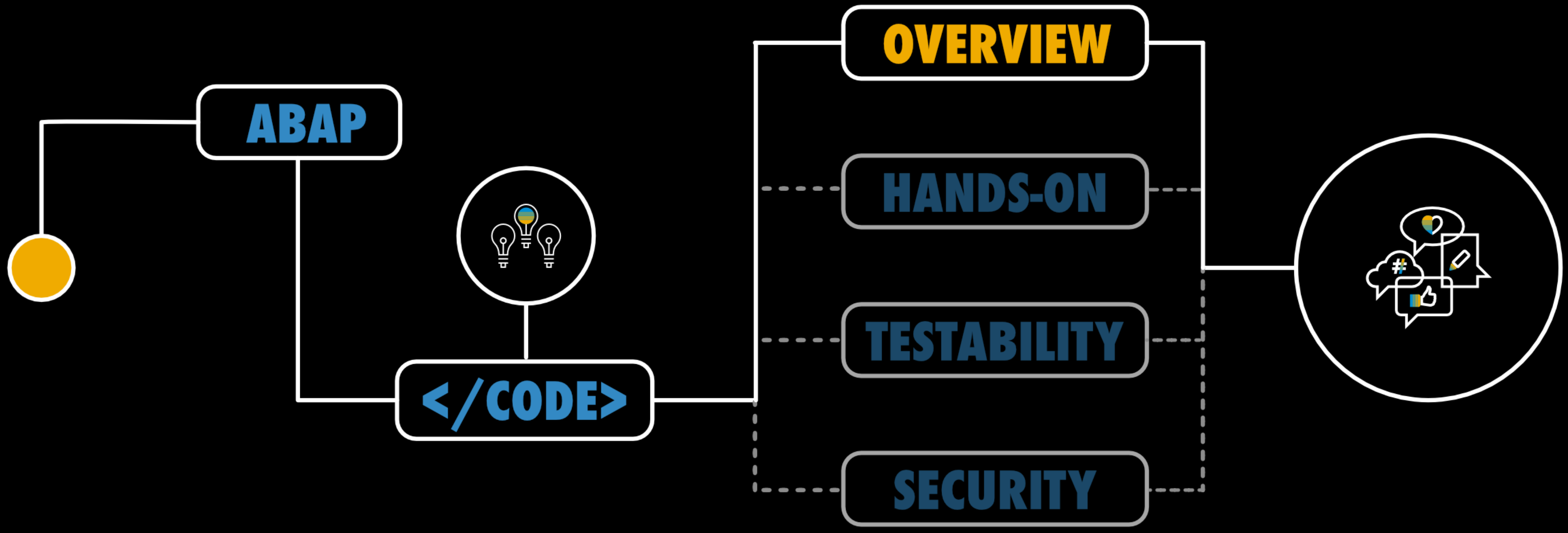


S4H140 – The ABAP Programming Model for SAP S/4HANA

Presentation from SAP TechEd 2017

EXTERNAL

SAP TECHED

Speakers

Las Vegas

September 25 – 29, 2017

Marcel Hermanns

Thomas Alexander Ritter

Bangalore

October 25 – 27, 2017

Gopalakrishnan Ramachandran

Barcelona

November 14 – 16, 2017

Marcel Hermanns

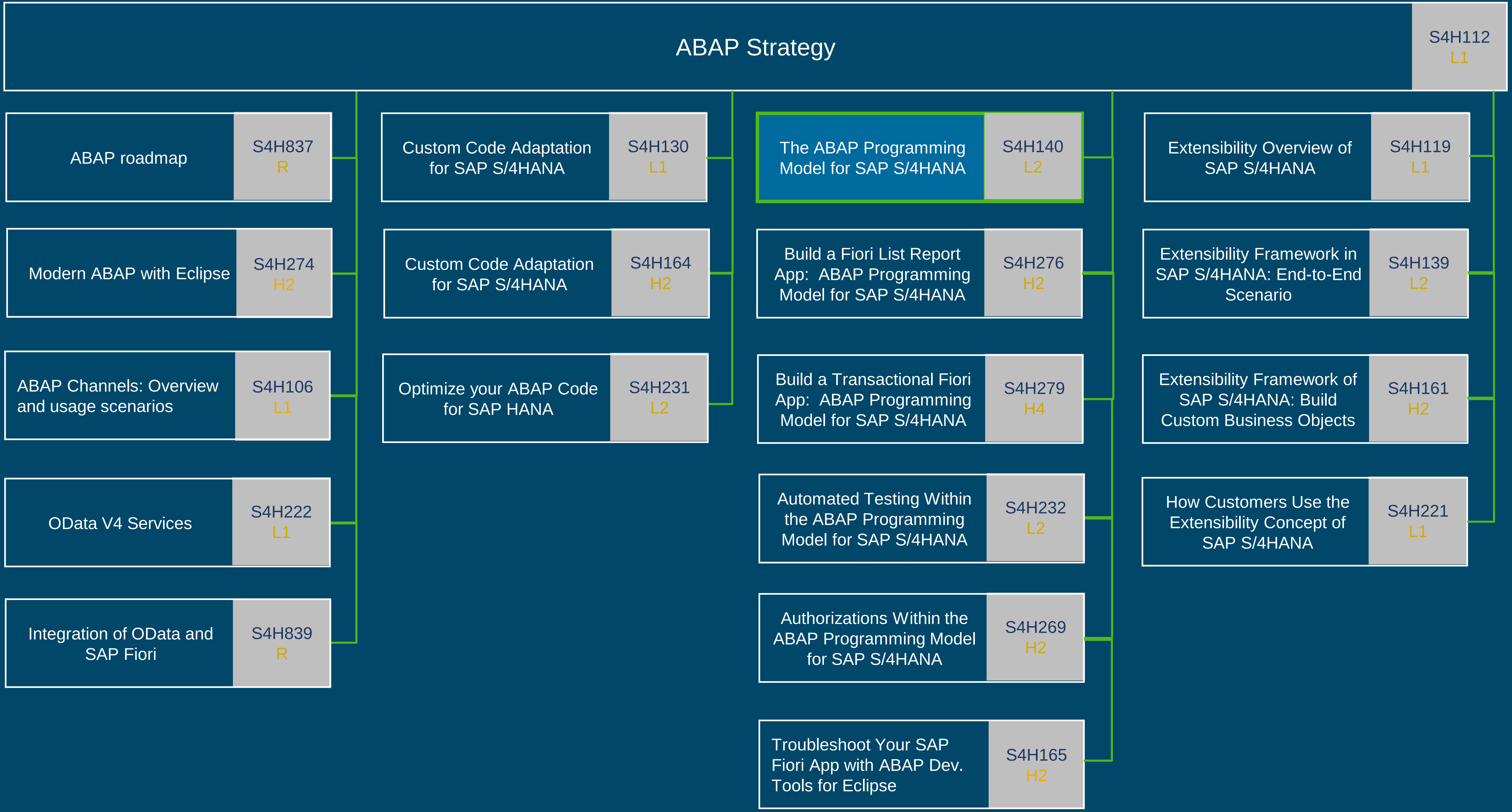
Thomas Alexander Ritter

Disclaimer

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

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

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



ABAP Strategy S4H112 L1

- ABAP roadmap S4H837 R
- Modern ABAP with Eclipse S4H274 H2
- ABAP Channels: Overview and usage scenarios S4H106 L1
- OData V4 Services S4H222 L1
- Integration of OData and SAP Fiori S4H839 R

- Custom Code Adaptation for SAP S/4HANA S4H130 L1
- Custom Code Adaptation for SAP S/4HANA S4H164 H2
- Optimize your ABAP Code for SAP HANA S4H231 L2

- The ABAP Programming Model for SAP S/4HANA S4H140 L2
- Build a Fiori List Report App: ABAP Programming Model for SAP S/4HANA S4H276 H2
- Build a Transactional Fiori App: ABAP Programming Model for SAP S/4HANA S4H279 H4
- Automated Testing Within the ABAP Programming Model for SAP S/4HANA S4H232 L2
- Authorizations Within the ABAP Programming Model for SAP S/4HANA S4H269 H2
- Troubleshoot Your SAP Fiori App with ABAP Dev. Tools for Eclipse S4H165 H2

- Extensibility Overview of SAP S/4HANA S4H119 L1
- Extensibility Framework in SAP S/4HANA: End-to-End Scenario S4H139 L2
- Extensibility Framework of SAP S/4HANA: Build Custom Business Objects S4H161 H2
- How Customers Use the Extensibility Concept of SAP S/4HANA S4H221 L1

Agenda

Product Qualities
and their Impact on the Programming Model

Read-only Apps
Basic Architecture: From Persistence to SAP Fiori Apps

Transactional Apps
RESTful Architecture: State and Draft

Cross Topics
OData V4, Testability and Access Authorizations

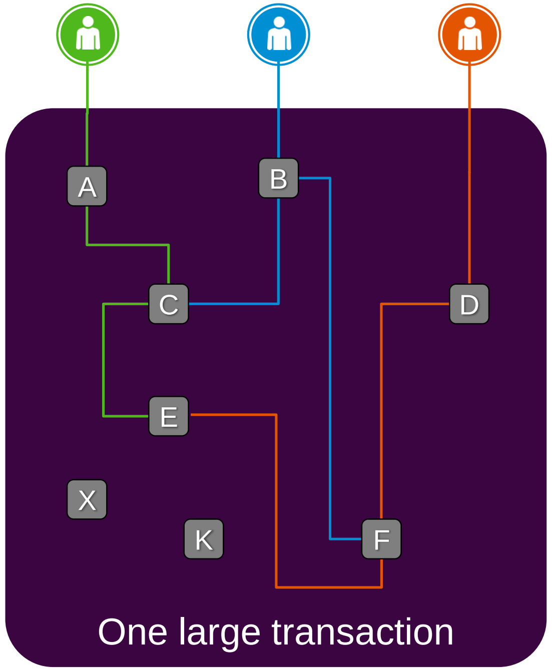
Outlook and Roadmap
The RESTful ABAP Programming Model

PRODUCT QUALITIES

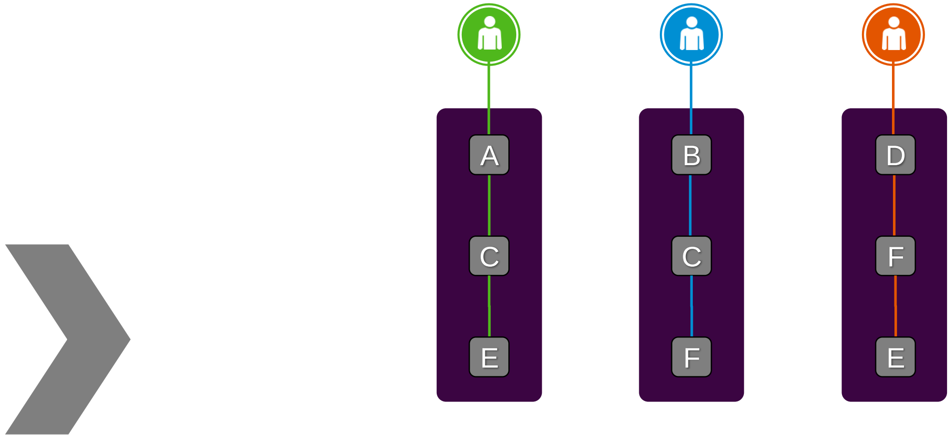
AND THEIR IMPACT ON THE PROGRAMMING MODEL

User experience qualities
Easy to use and role-based

Requirement — I want an easy to use application providing only relevant functionality for the tasks of my role



One large transaction



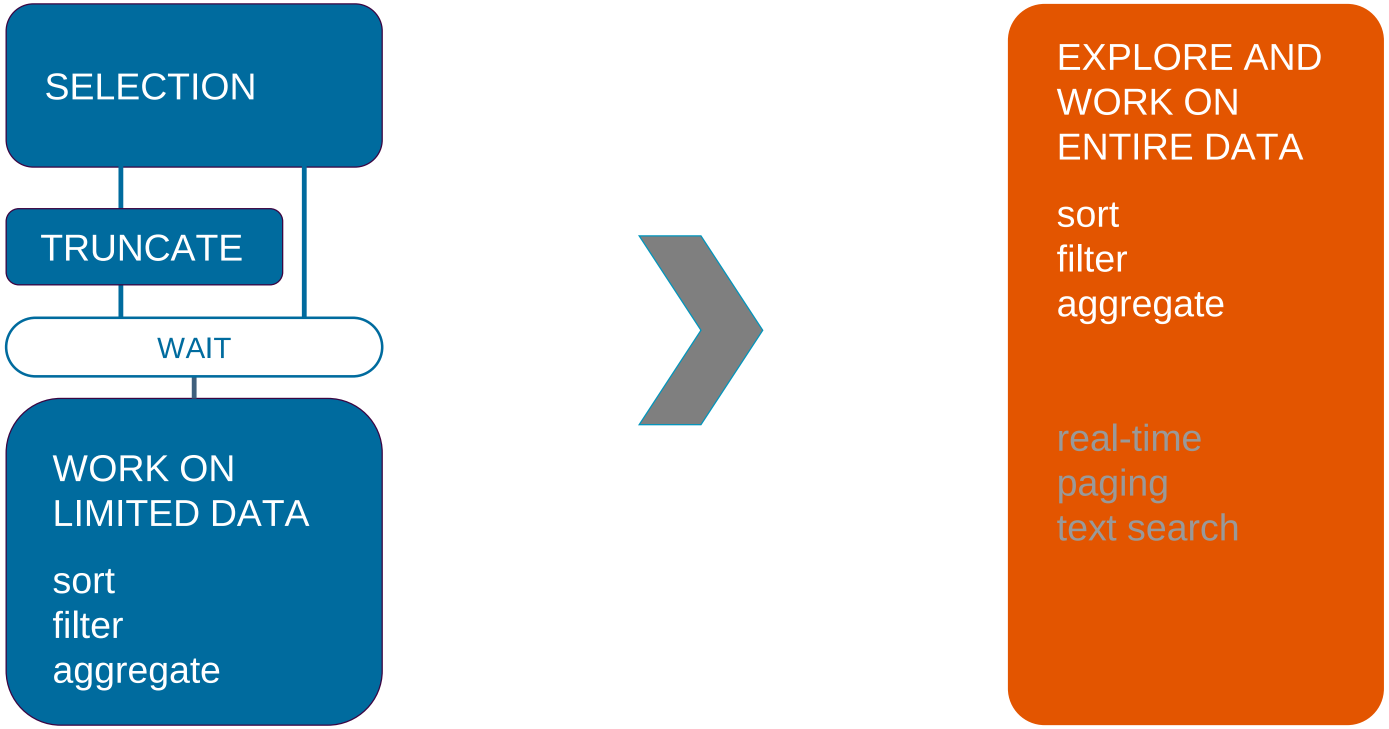
Multiple role-based and simple apps but with redundant (but not duplicated) functionality and data access

Impact on programming model — Efficient programming model to support multiple applications on same data and functionality
Manage redundancies in applications without inconsistencies

User experience qualities

Effective and efficient work on entirety of data

Requirement — I want the freedom to explore and work with entire data – not only on restricted or truncated data


SELECTION

TRUNCATE

WAIT

WORK ON LIMITED DATA

sort
filter
aggregate

EXPLORE AND WORK ON ENTIRE DATA

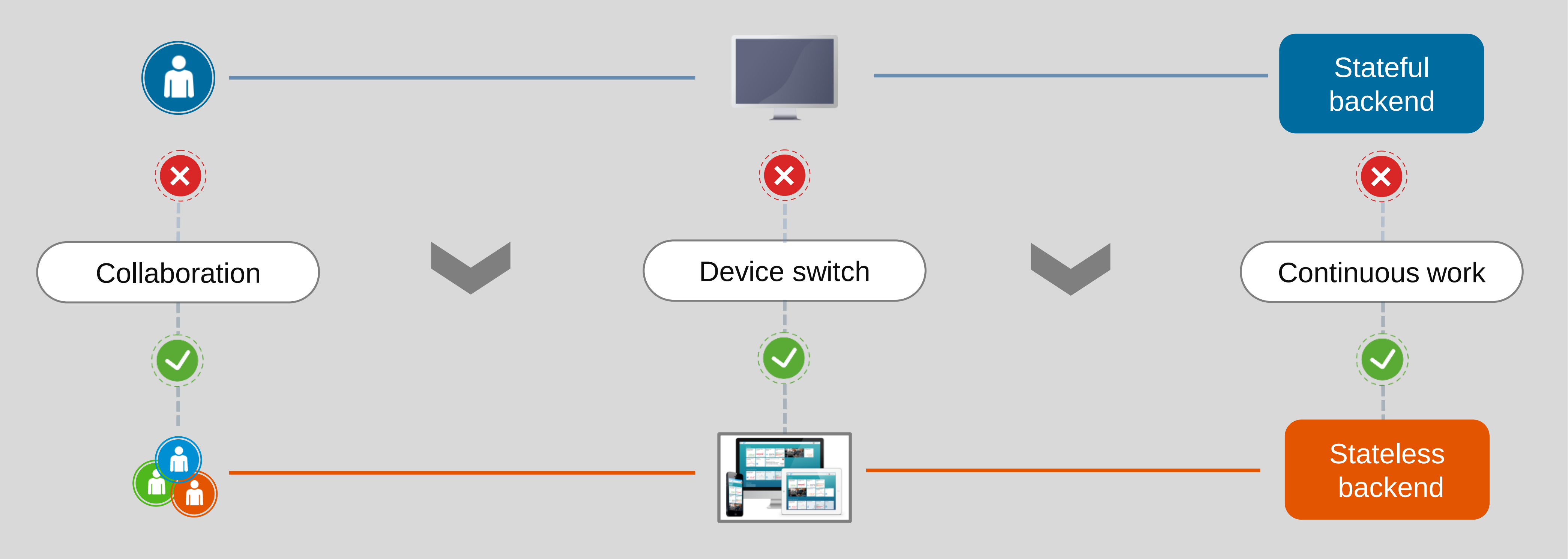
sort
filter
aggregate

real-time
paging
text search

Impact on programming model — Easily enable navigation, filtering, text search and grouping of data for all applications
Provide intrinsic support for exploiting SAP HANA features and performance

User experience qualities
Continuous work, device switch and collaboration

Requirement I want to work from everywhere, continue on interrupted work and change my devices when needed



Impact on programming model Remove tight coupling of client and server to avoid server stickiness

Cloud qualities

Scalability & elasticity, availability

On-premise	Cloud		
	Role	**Expectations**	**Requirement**
Rare updates with planned downtime	Consumer	**Availability and continuous delivery** of fixes and features	**Business Continuity** Zero downtime maintenance and online imports **No Server stickiness of apps**
Long lasting planning of physical resources	Provider	**TCO** - Resource consumption based on system load	**Changing assignment** of virtual resources by adding/removing app servers dynamically **No Server stickiness of apps**

Impact on programming model

Transparent integration of cloud services as part of the programming model
Eliminate server stickiness

Cloud qualities
Extensibility*

UI adaptation

Custom fields

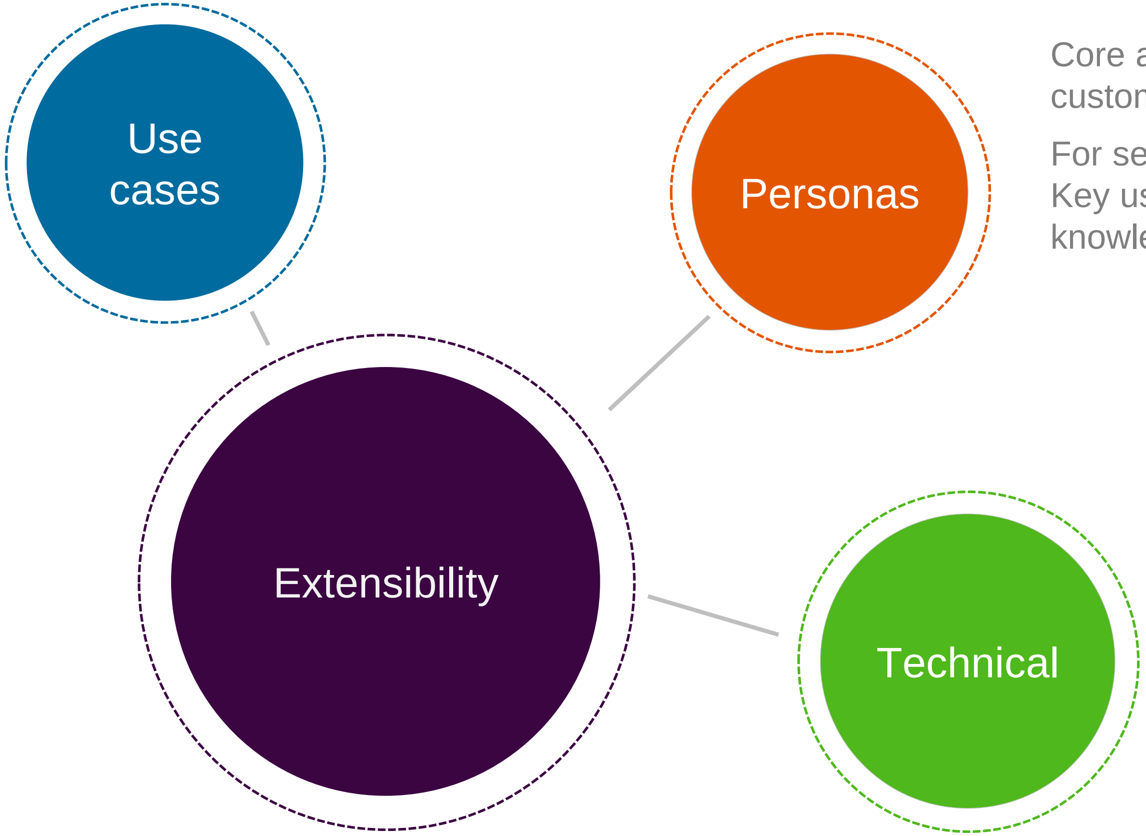
Business logic extensions

Output management

Analytics extensibility

Custom business objects and nodes

Service configuration and SAP Fiori catalog / role extensibility



Core app developer at SAP, partner or customer

For selected scenarios:
Key users with special business knowledge / consultants

Modification-free

Clash-free

Use whitelisted APIs and extension points

Compliant with zero-downtime principles

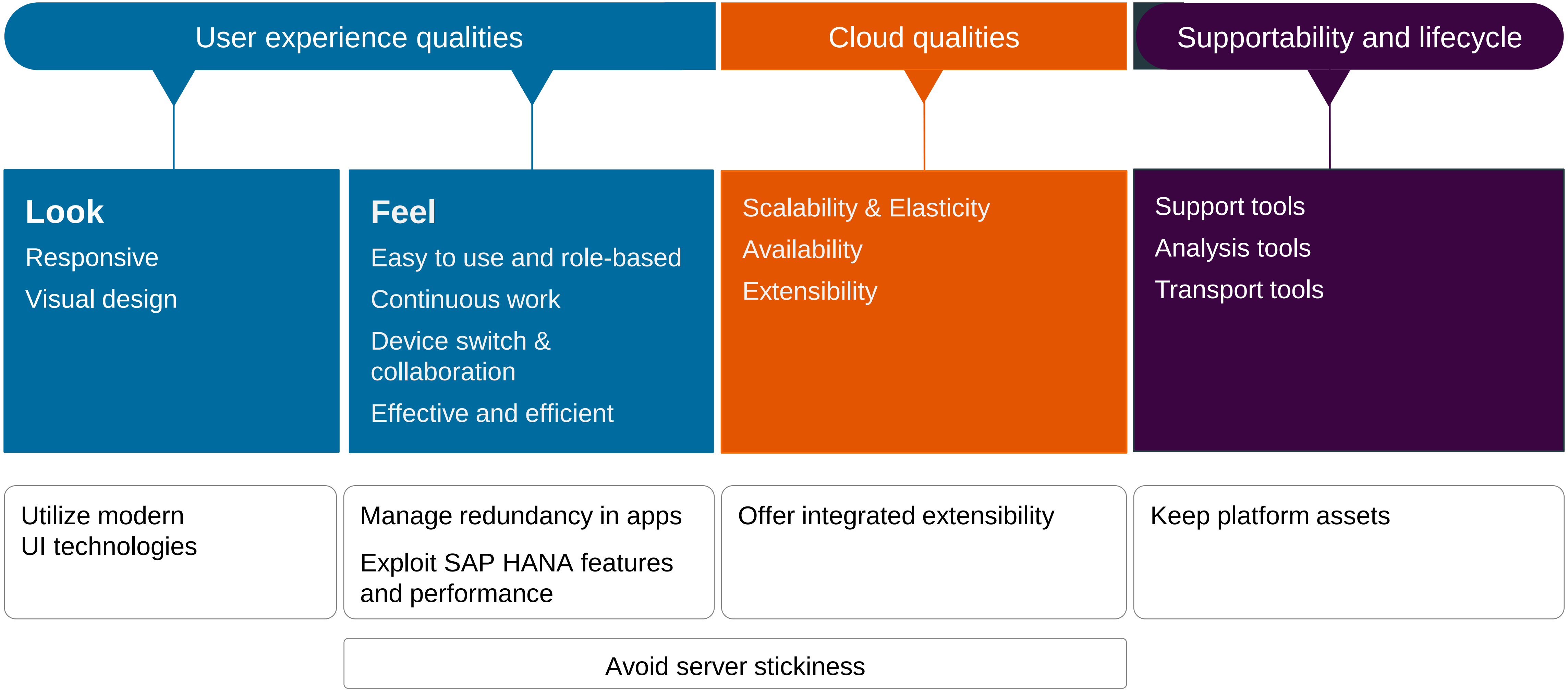
Impact on programming model

Offer integrated decoupled extensibility on developer and key user level
Toolset to be cloud ready

*mandatory for cloud environment, recommended for on-premise environment

Product qualities

Impact on programming model



User experience qualities

Cloud qualities

Supportability and lifecycle

Look

Responsive

Visual design

Feel

Easy to use and role-based

Continuous work

Device switch & collaboration

Effective and efficient

Scalability & Elasticity

Availability

Extensibility

Support tools

Analysis tools

Transport tools

Utilize modern UI technologies

Manage redundancy in apps

Exploit SAP HANA features and performance

Offer integrated extensibility

Keep platform assets

Avoid server stickiness

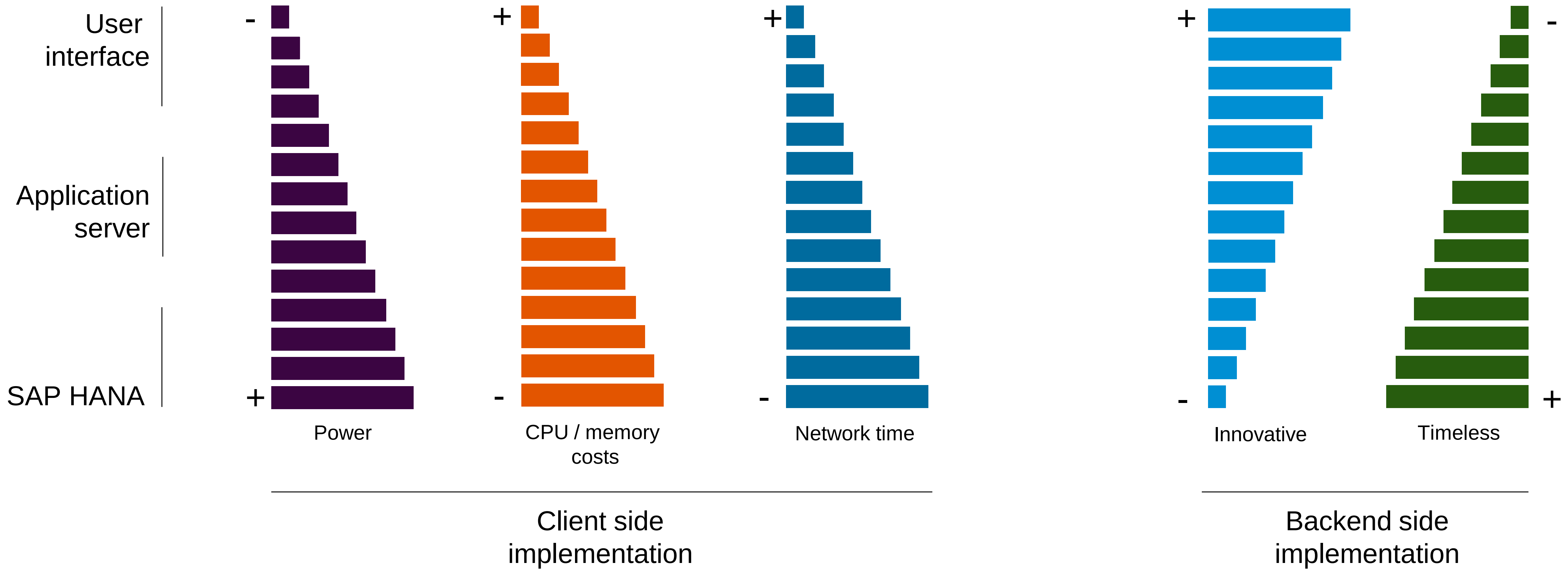
A new programming model from a different angle

Engineering aspects

Do the right things at the right layer

Business logic, state, caching, no. of roundtrips

Decouple UI (technology independent) semantics from client (repository)



READ-ONLY APPS

BASIC ARCHITECTURE: FROM PERSISTENCE TO SAP FIORI APPS

Develop SAP HANA optimized SAP Fiori apps

The new programming model

Freestyle or templates based UI development

SAPUI5 / SAP Fiori Elements

OData protocol infrastructure

SAP Gateway

Effective and efficient application development

Modern ABAP language and development tools

Transaction, error and event handling services

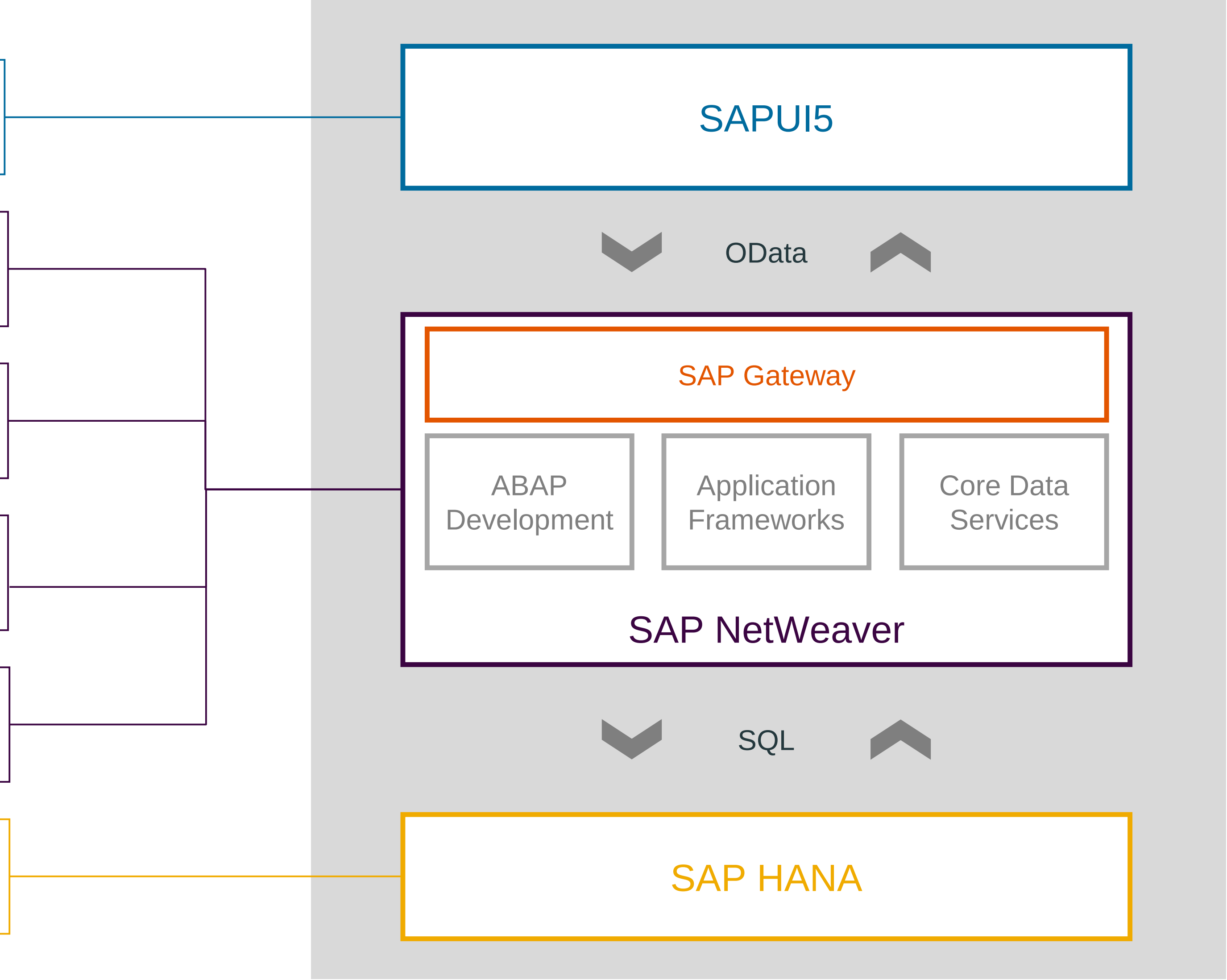
Business object framework (BOPF)

One common data modelling technology for all scenarios

Core Data Services (CDS**)**

Exploit SAP HANA features and performance

AMDP and CDS Table Functions



Core Data Services (CDS)

Combine the best of two worlds for ABAP development

VIEW BUILDING

SQL views with restricted feature set for SAP HANA capabilities

LIFECYCLE MANAGEMENT

ABAP lifecycle management e.g. transports

ABAP type-system

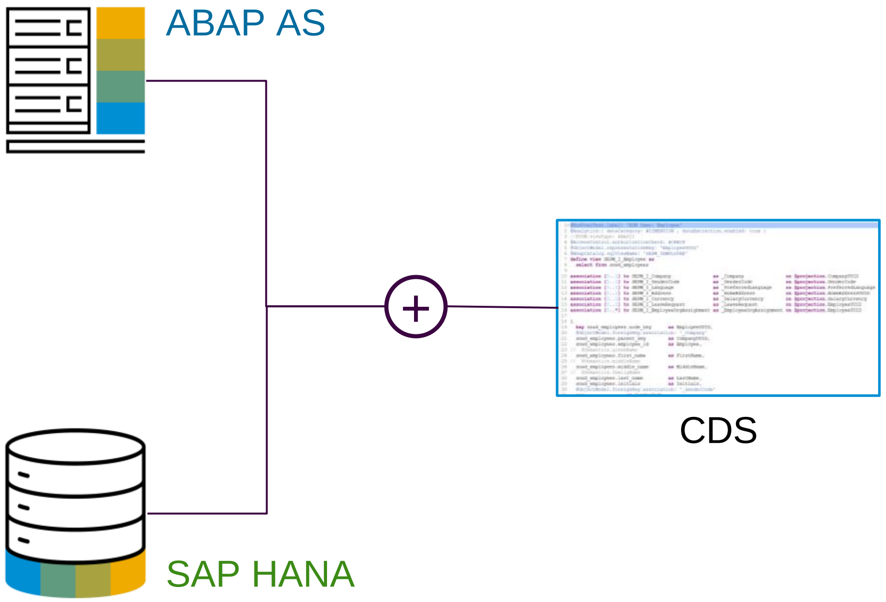
VIEW BUILDING

Specialized views (analytical views, attribute views, calculated views, …)

LIFECYCLE MANAGEMENT

Non ABAP lifecycle management e.g. transports

Non ABAP type-system



VIEW BUILDING

Advanced SQL view building

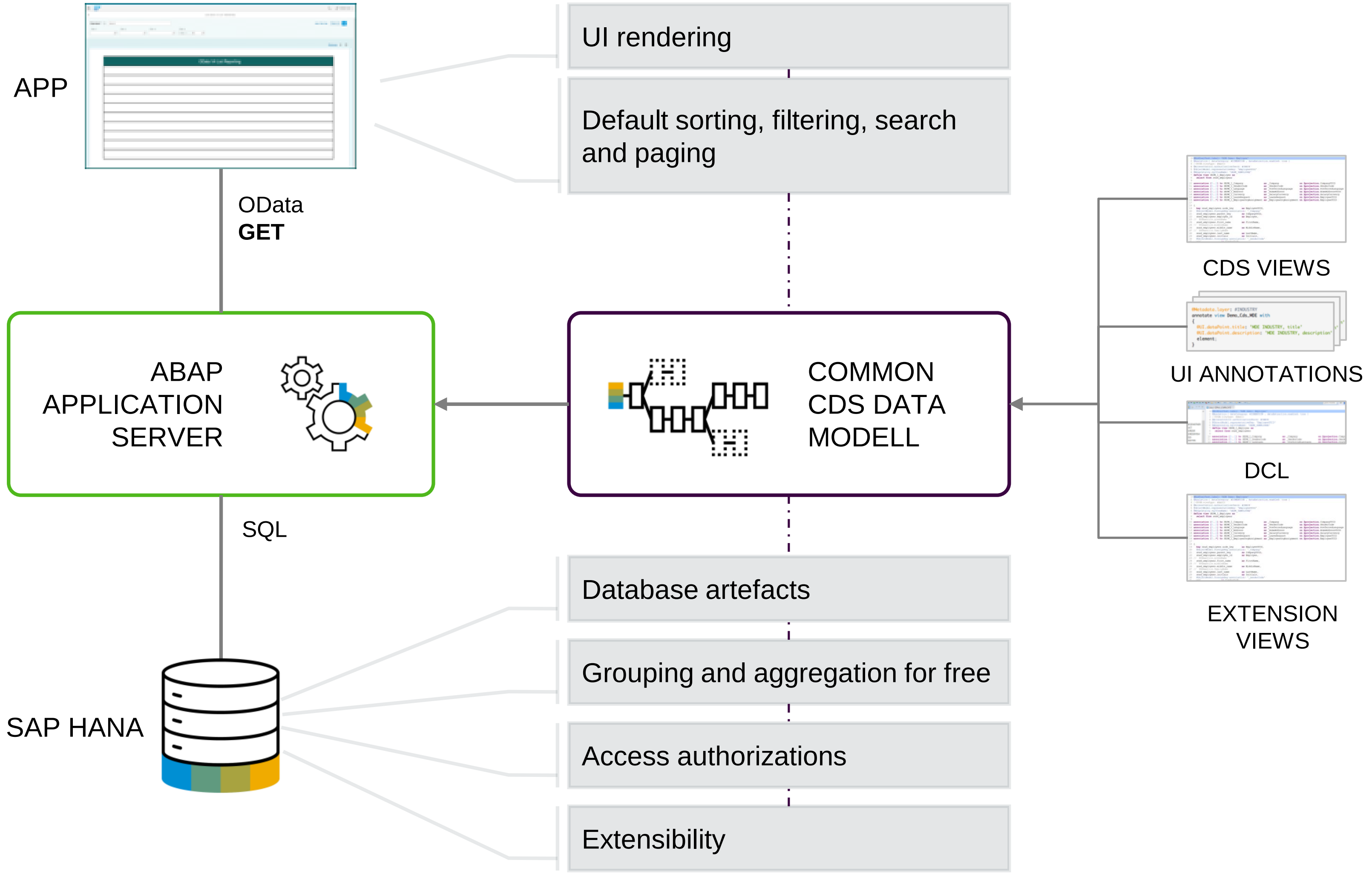
Integration of break-outs to utilize SAP HANA features with code pushdown capabilities (CDS Table Functions, built-in SQL functions)

LIFECYCLE MANAGEMENT

ABAP lifecycle management e.g. transports

ABAP type-system

CDS in an end-to-end programming model



APP

OData
GET

ABAP
APPLICATION
SERVER

SQL

SAP HANA

UI rendering

Default sorting, filtering, search and paging

COMMON
CDS DATA
MODELL

Database artefacts

Grouping and aggregation for free

Access authorizations

Extensibility

CDS VIEWS

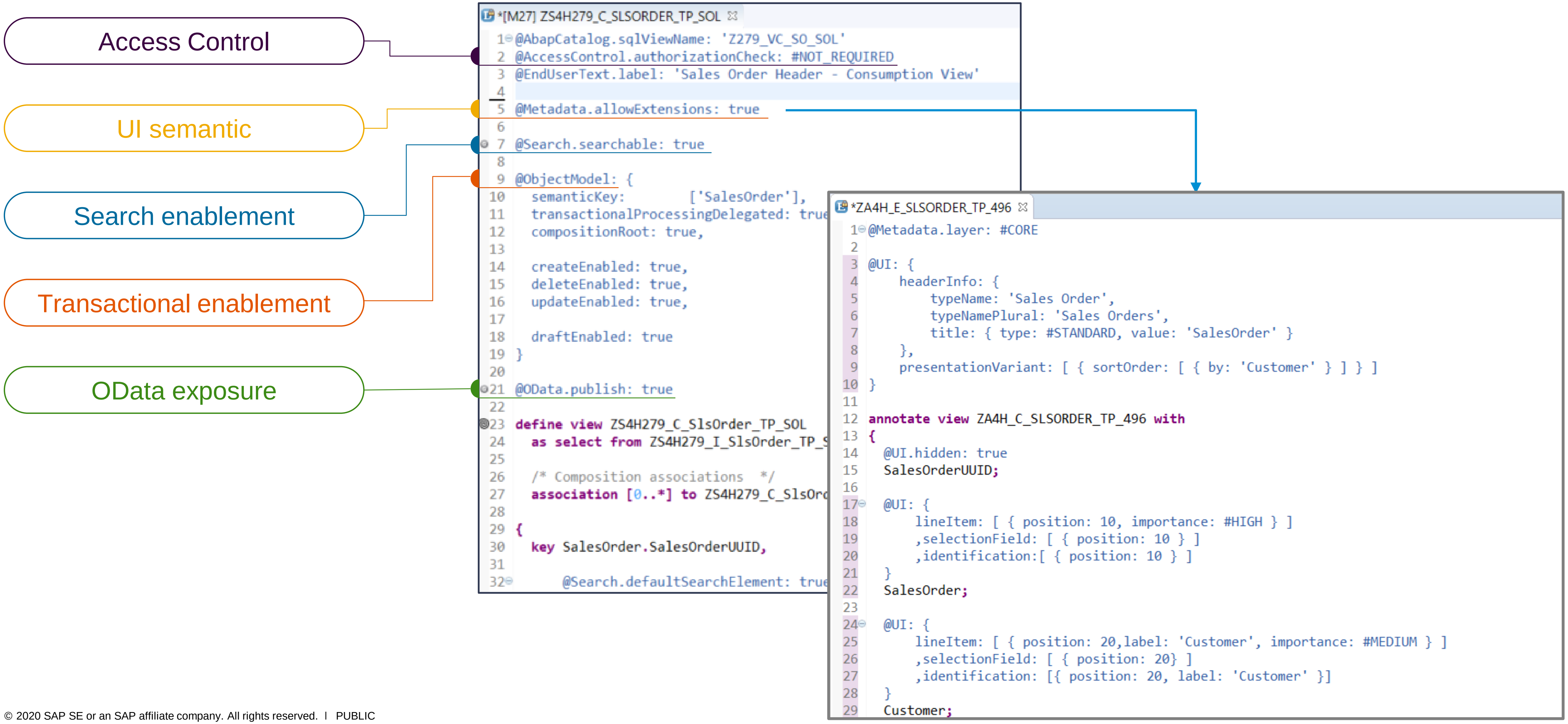
UI ANNOTATIONS

DCL

EXTENSION
VIEWS

Develop SAP HANA optimized SAP Fiori apps

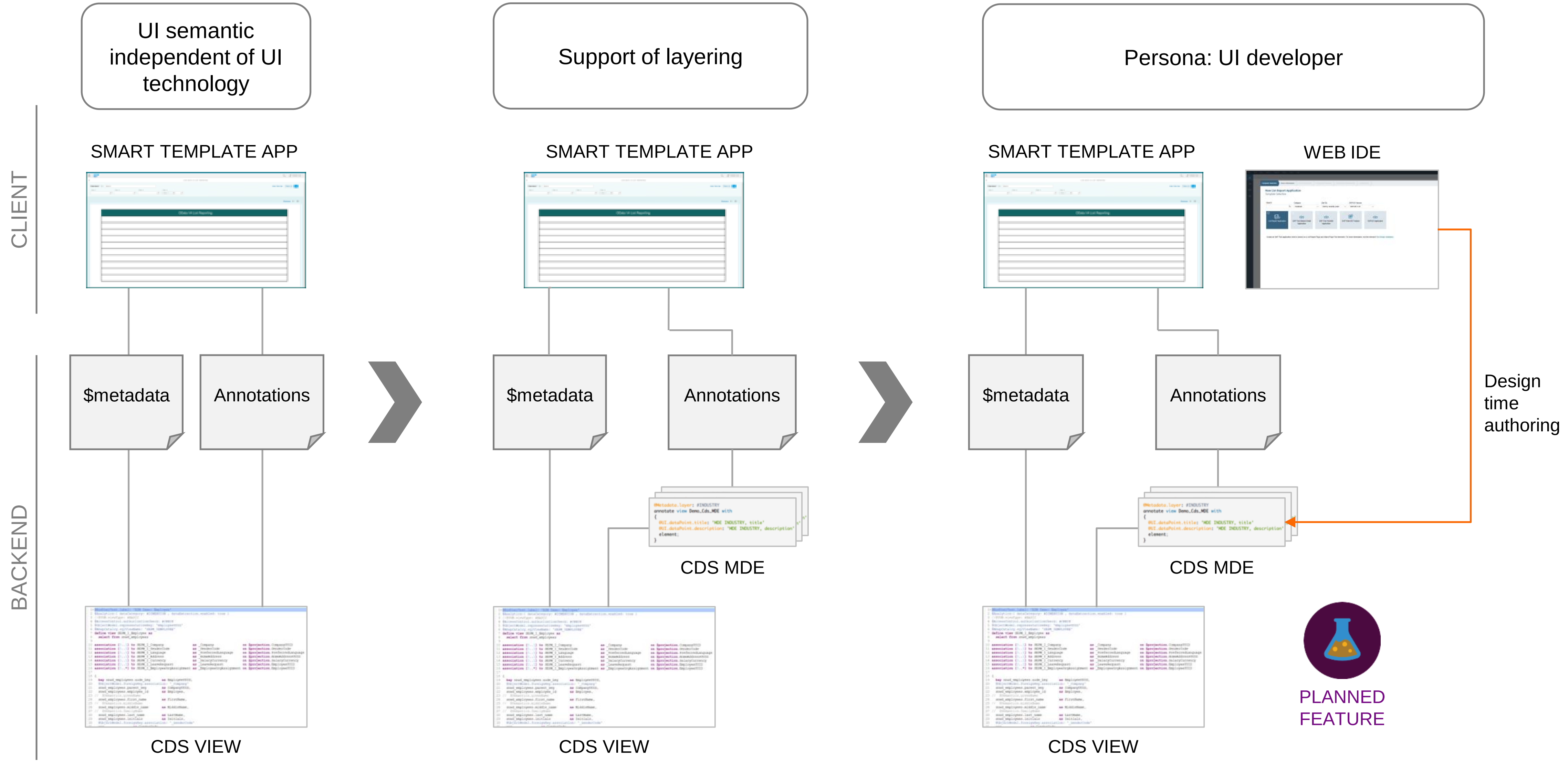
Example CDS view with annotations and Metadata Extension (MDE)



```
@AbapCatalog.sqlViewName: 'Z279_VC_SO_SOL'
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Sales Order Header - Consumption View'

@Metadata.allowExtensions: true

@Search.searchable: true

@ObjectModel: {
  semanticKey:        ['SalesOrder'],
  transactionalProcessingDelegated: true
  compositionRoot: true,

  createEnabled: true,
  deleteEnabled: true,
  updateEnabled: true,

  draftEnabled: true
}

@OData.publish: true

define view ZS4H279_C_SlsOrder_TP_SOL
  as select from ZS4H279_I_SlsOrder_TP_S

  /* Composition associations  */
  association [0..*] to ZS4H279_C_SlsOrd

{
  key SalesOrder.SalesOrderUUID,

      @Search.defaultSearchElement: true
```

```
@Metadata.layer: #CORE

@UI: {
    headerInfo: {
        typeName: 'Sales Order',
        typeNamePlural: 'Sales Orders',
        title: { type: #STANDARD, value: 'SalesOrder' }
    },
    presentationVariant: [ { sortOrder: [ { by: 'Customer' } ] } ]
}

annotate view ZA4H_C_SLSORDER_TP_496 with
{
  @UI.hidden: true
  SalesOrderUUID;

  @UI: {
      lineItem: [ { position: 10, importance: #HIGH } ]
      ,selectionField: [ { position: 10 } ]
      ,identification:[ { position: 10 } ]
  }
  SalesOrder;

  @UI: {
      lineItem: [ { position: 20,label: 'Customer', importance: #MEDIUM } ]
      ,selectionField: [ { position: 20} ]
      ,identification: [{ position: 20, label: 'Customer' }]
  }
  Customer;
```

UI annotations



```
@Metadata.layer: #INDUSTRY
annotate view Demo_Cds_MDE with
{
  @UI.dataPoint.title: 'MDE INDUSTRY, title'
  @UI.dataPoint.description: 'MDE INDUSTRY, description'
  element;
}
```

Demo

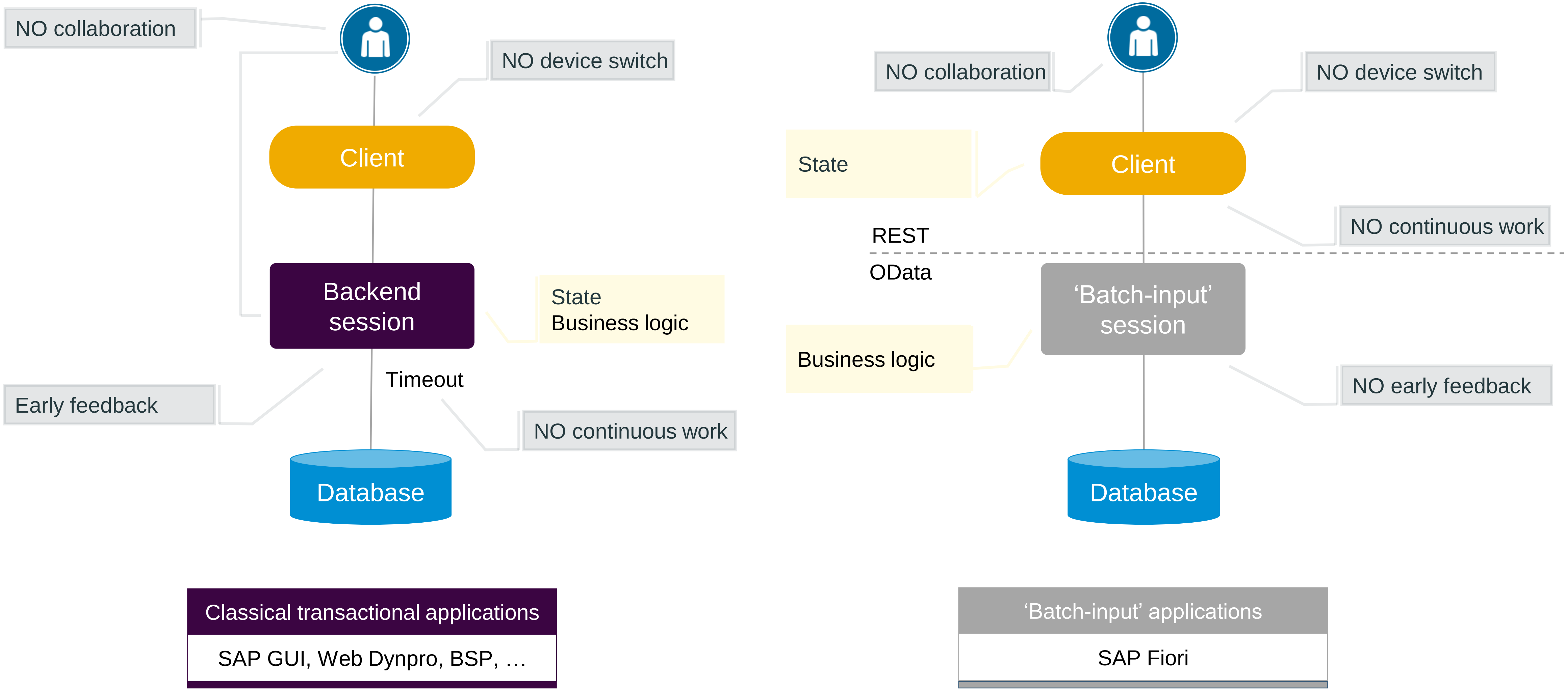
READ-ONLY APPS
FROM PERSISTENCE TO SAP FIORI APPS

TRANSACTIONAL APPS

RESTFUL ARCHITECTURE: STATE AND DRAFT

Impact of a RESTful architecture

Impact on session, state and business logic in classical applications



NO collaboration

NO device switch

Client

Backend session

State
Business logic

Early feedback

Timeout

NO continuous work

Database

Classical transactional applications

SAP GUI, Web Dynpro, BSP, …

NO collaboration

NO device switch

State

Client

NO continuous work

REST

OData

'Batch-input' session

Business logic

NO early feedback

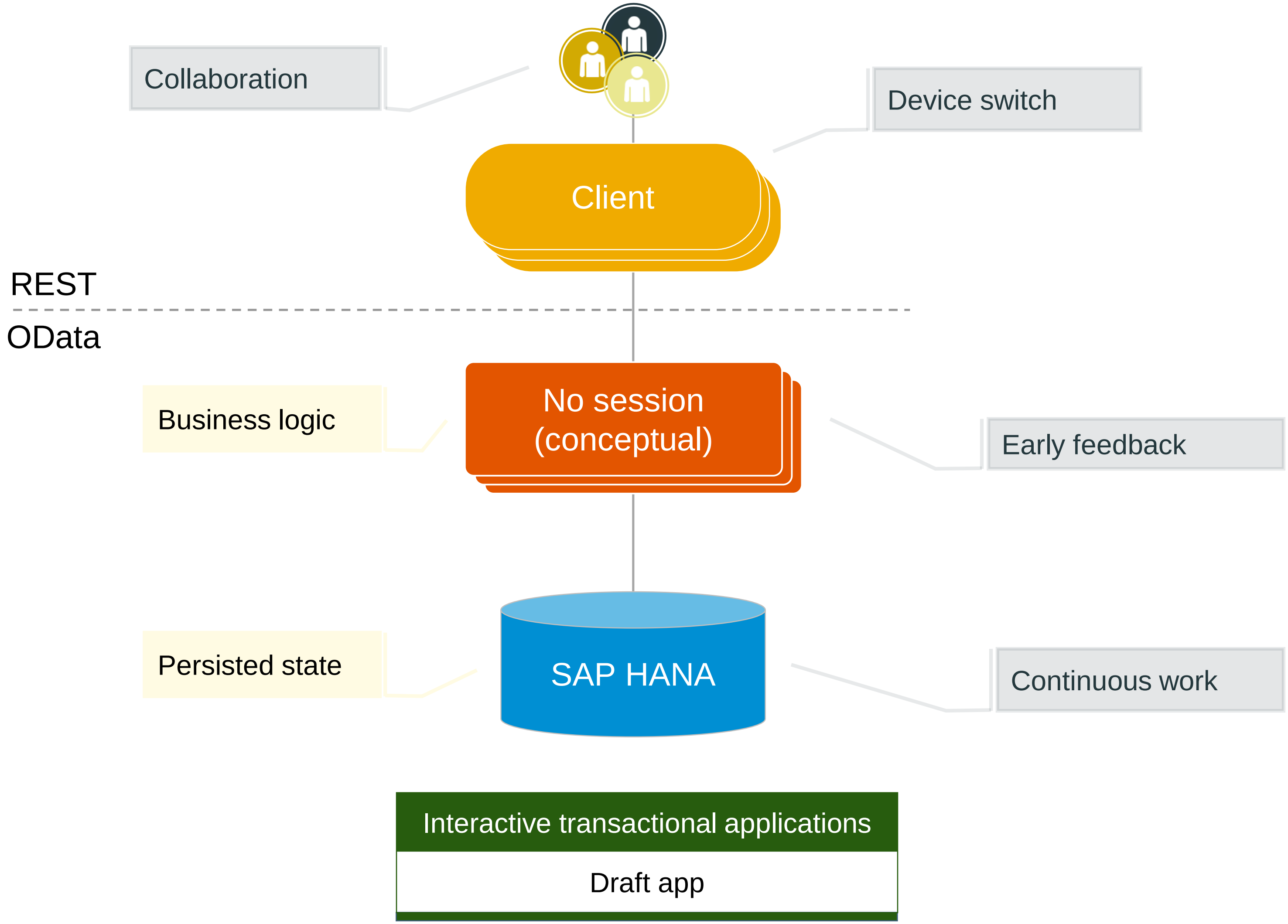
Database

'Batch-input' applications

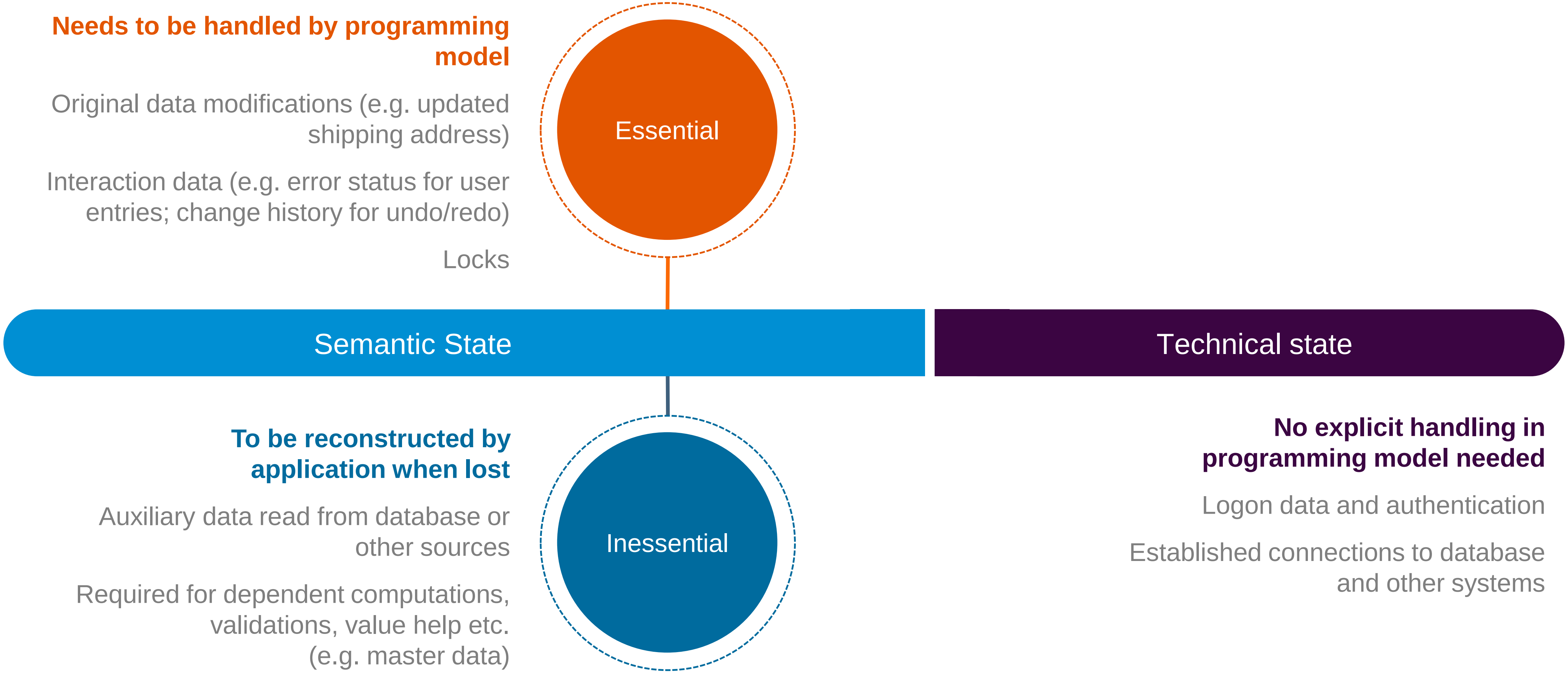
SAP Fiori

Impact of a RESTful architecture

Impact on session, state and business logic in new applications



Impact of a RESTful architecture – From stateful to REST: State categories



Needs to be handled by programming model

Original data modifications (e.g. updated shipping address)

Interaction data (e.g. error status for user entries; change history for undo/redo)

Locks

Essential

Semantic State

Technical state

Inessential

To be reconstructed by application when lost

Auxiliary data read from database or other sources

Required for dependent computations, validations, value help etc.
(e.g. master data)

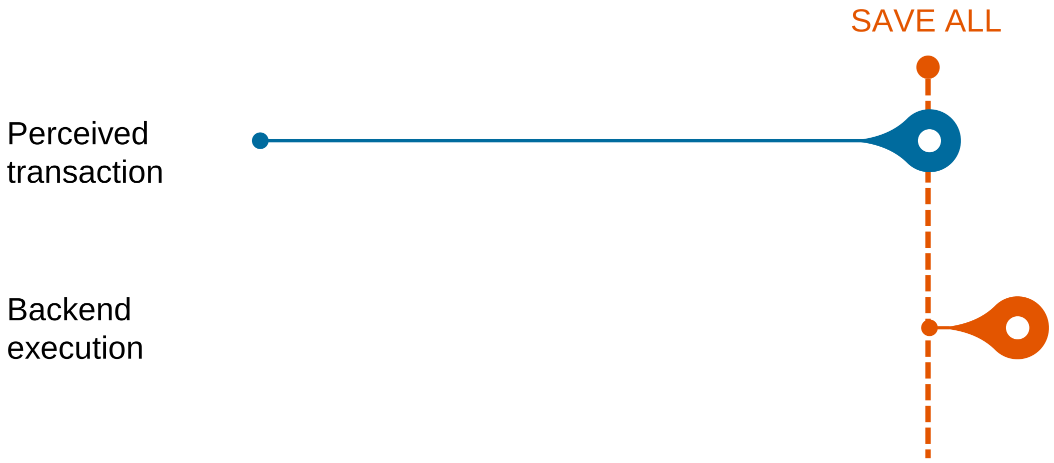
No explicit handling in programming model needed

Logon data and authentication

Established connections to database and other systems

Impact of a RESTful architecture

Impact of choosing ‘batch-input’ transactional as application type

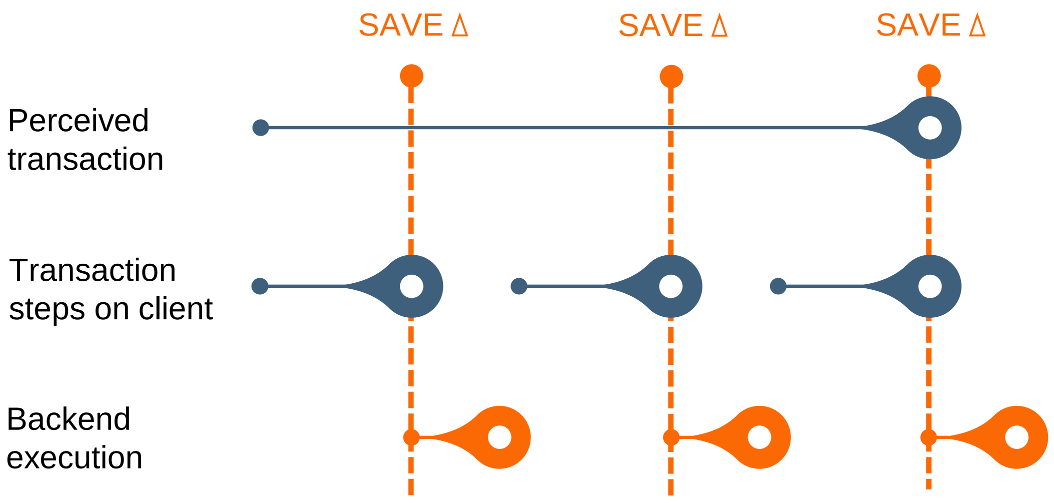


IMPACT
on backend

Calculation of derived data and validations not during interaction, but only after SAVE

➡ No early feedback
Result (e.g. data and messages) after successful or declined SAVE

Impact of a RESTful architecture
Draft apps



IMPACT
for end-user

Early feedback: Calculations, validations and field control during interaction (e.g. cursor move)

Draft qualities: Data loss prevention, continuous work, device switch,…

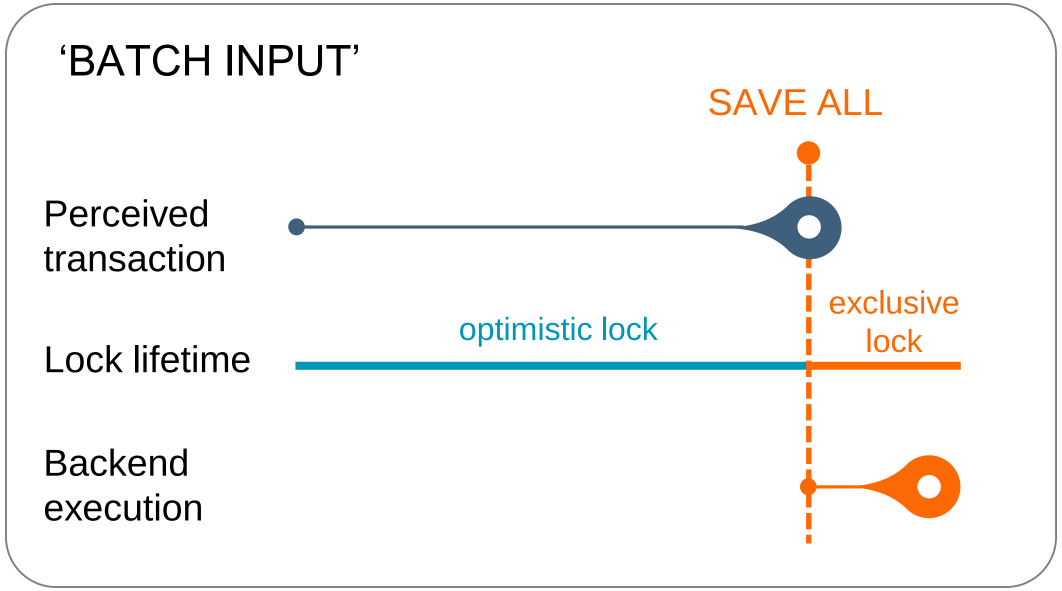
IMPACT for programming model

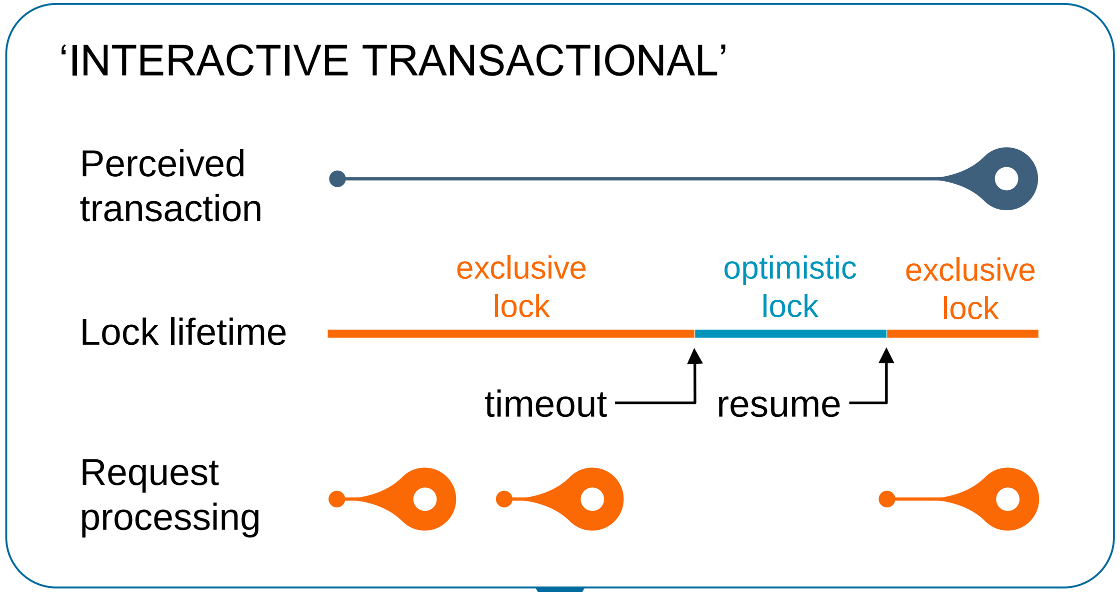
Changes to be stored even incomplete and inconsistent in secondary persistence ➨ **Draft**

When data is activated, draft is transferred into process relevant data ➨ **Active Data**

Impact of a RESTful architecture

State handling / locks





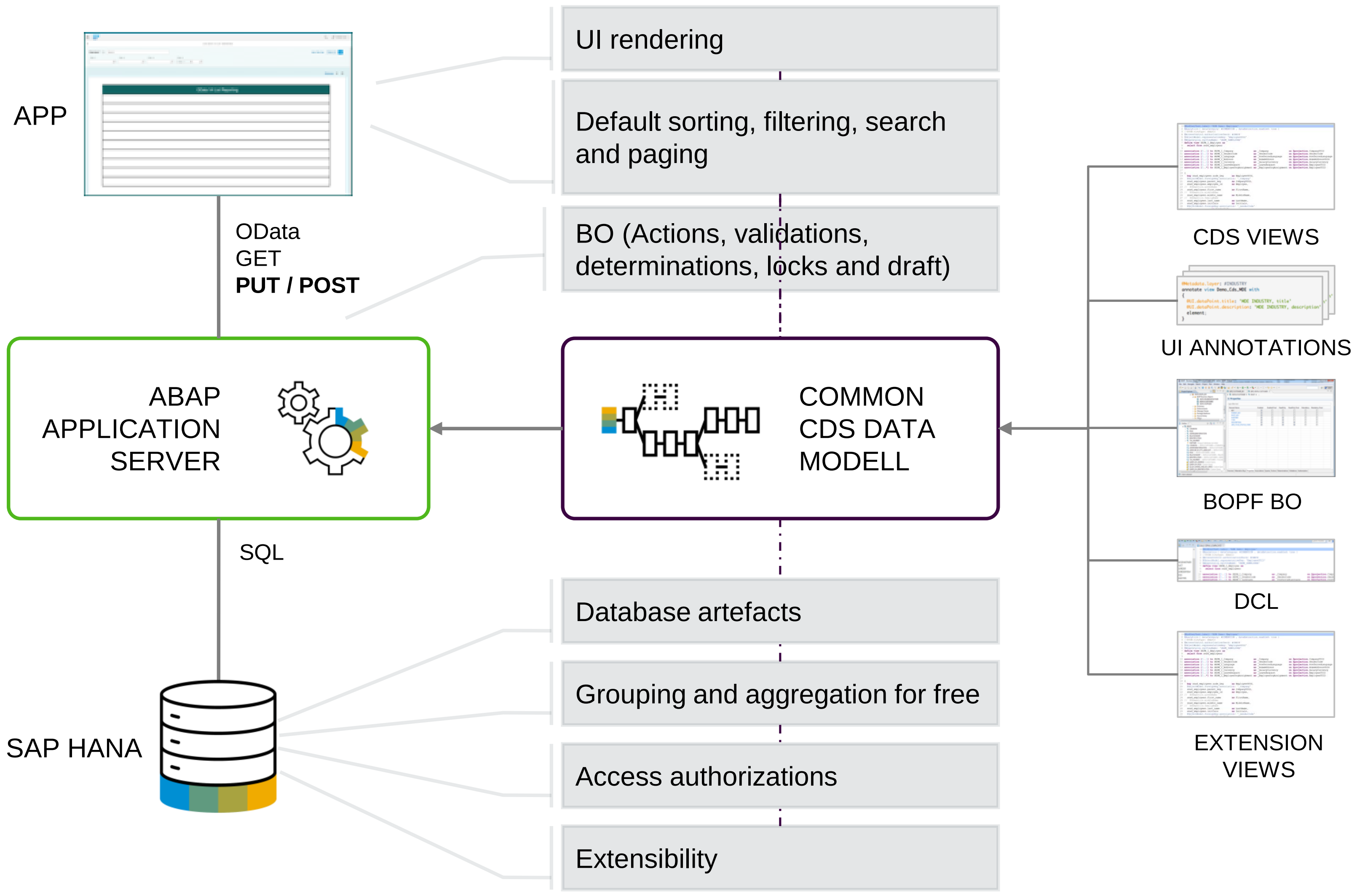
End-User perceived transaction distributed over several days, so locks can‘t be hold all the time

Locks are not bound to a session and have their own time-out. After time-out of exclusive locks an optimistic lock is used. ‘Resume’ re-acquires the exclusive locks

Request processing is executed in a single ABAP session

Lifetime of session might be longer for performance reasons (‘softstate’)

CDS in an end-2-end programming model



APP

OData
GET
PUT / POST

ABAP
APPLICATION
SERVER

SQL

SAP HANA

UI rendering

Default sorting, filtering, search and paging

BO (Actions, validations, determinations, locks and draft)

COMMON
CDS DATA
MODELL

Database artefacts

Grouping and aggregation for free

Access authorizations

Extensibility

CDS VIEWS

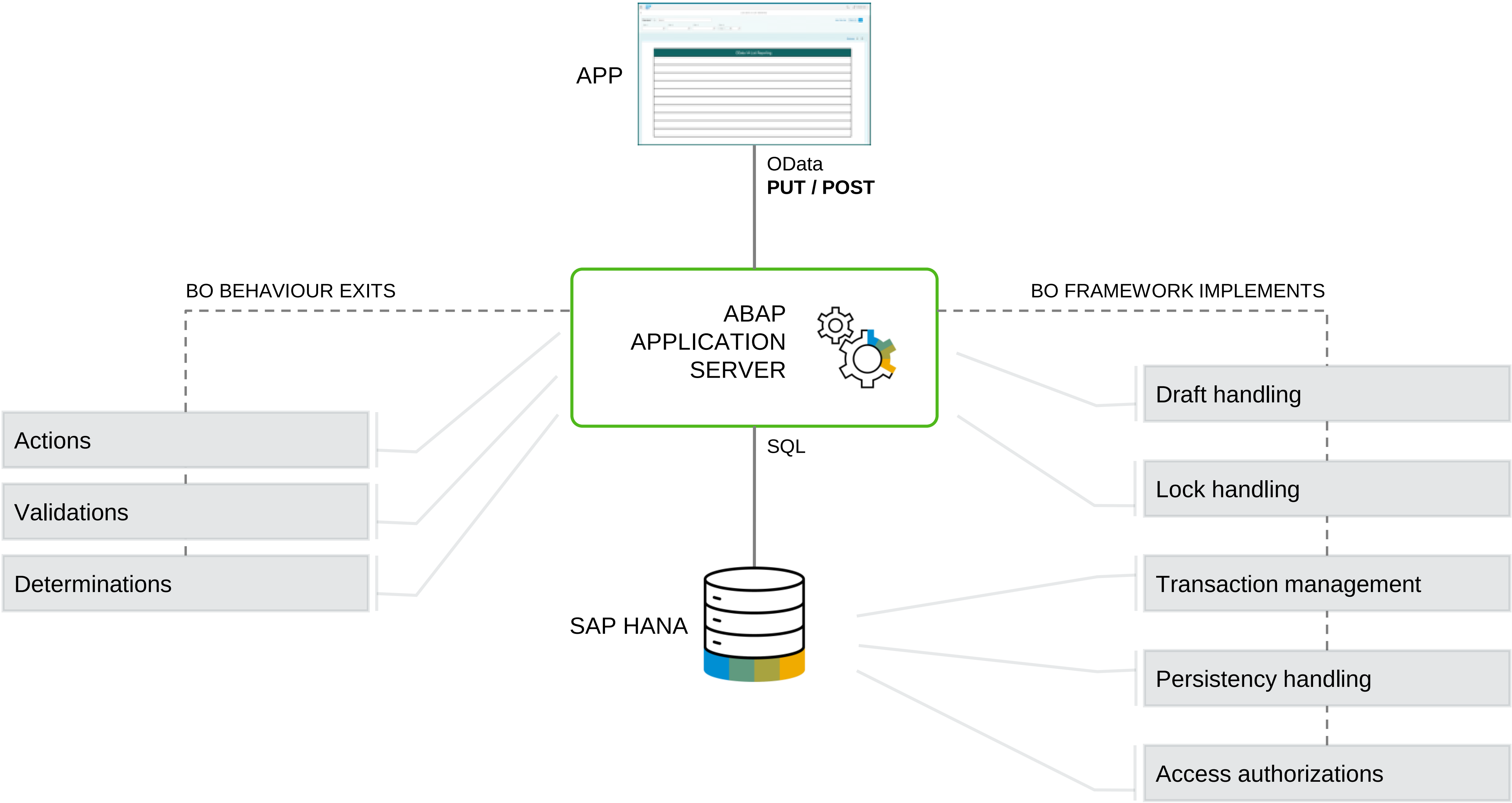
UI ANNOTATIONS

BOPF BO

DCL

EXTENSION
VIEWS

End-2-end support for transactional apps based on Business Objects



APP

OData
PUT / POST

ABAP
APPLICATION
SERVER

SQL

SAP HANA

BO BEHAVIOUR EXITS

- Actions
- Validations
- Determinations

BO FRAMEWORK IMPLEMENTS

- Draft handling
- Lock handling
- Transaction management
- Persistency handling
- Access authorizations

Where to invest now to be prepared for the SAP S/4HANA programming model?

YOU SHOULD

Follow the programming model and best practices and use…

- *Core Data Services (CDS) for database artefacts
  (≥ 7.40, Documentation)
- *CDS Metadata Extensions for UI Annotations
  (≥ 7.51 SP2, Documentation)
- *DCL for read/query instance-based authority checks
  (≥ 7.50, Documentation)
- BOPF stand-alone
  (≥ 7.40, Documentation, newer releases: CDS/BOPF integration)
- *BOPF and CDS integration including draft
  (≥ 7.51 SP2, Documentation)
- Gateway integration of CDS or BOPF
  (= 7.40, Documentation, newer releases: OData Exposure)
- *OData Exposure of CDS / BOPF for SAP Fiori and future development
  (≥ 7.50 SP5, Documentation)
- Floorplan-Manager integration of CDS and BOPF
  (≥ 7.40, Documentation)

DO NOT

Implement things that are already solved:

- Manual implementation of **read-only OData calls** to DB
- Business logic mixed with **technical aspects**
  (e.g. locks, authority-check, LUW handling, persistency)
- Business logic mixed with **protocol specific** APIs
  (e.g. PBO/PAI, Gateway classes: DPC_EXT)

BENEFITS

Reuse / prepare your skillset and coding for the future

- Reuse CDS and DCL in SAP S/4HANA
- Lower TCD for the future: Minimal investment on technical protocol level

***: best practice Fiori apps, E2E documentation**

Demo

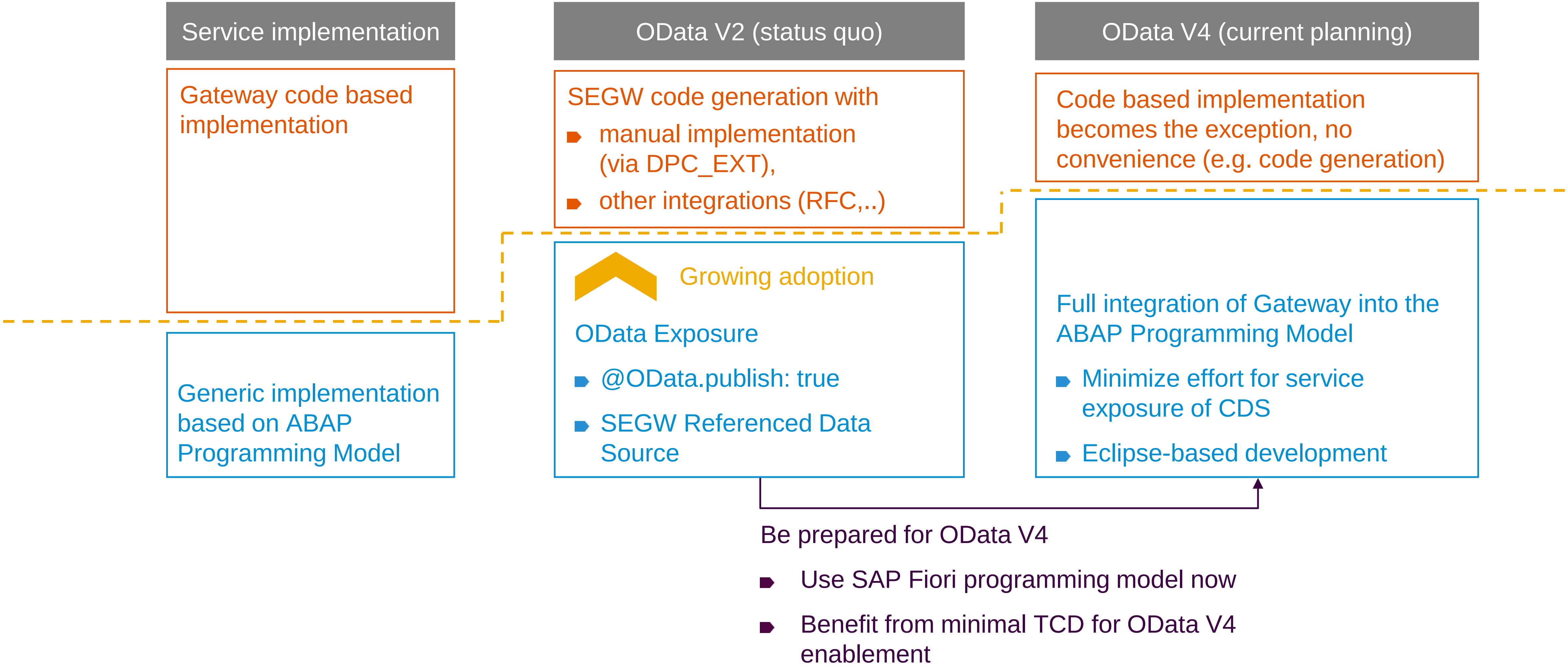
Draft enabling of an transactional app

CROSS TOPICS

ODATA V4, TESTABILITY AND ACCESS AUTHORIZATIONS

OData V4

Implementation options for OData services



Service implementation

Gateway code based implementation

Generic implementation based on ABAP Programming Model

OData V2 (status quo)

SEGW code generation with

- manual implementation (via DPC_EXT),
- other integrations (RFC,..)

Growing adoption

OData Exposure

- @OData.publish: true
- SEGW Referenced Data Source

OData V4 (current planning)

Code based implementation becomes the exception, no convenience (e.g. code generation)

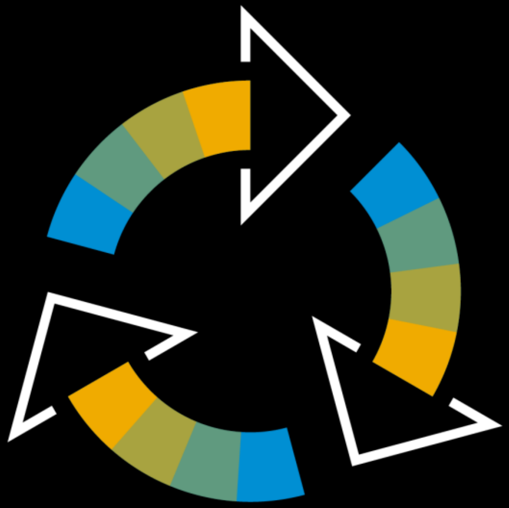
Full integration of Gateway into the ABAP Programming Model

- Minimize effort for service exposure of CDS
- Eclipse-based development

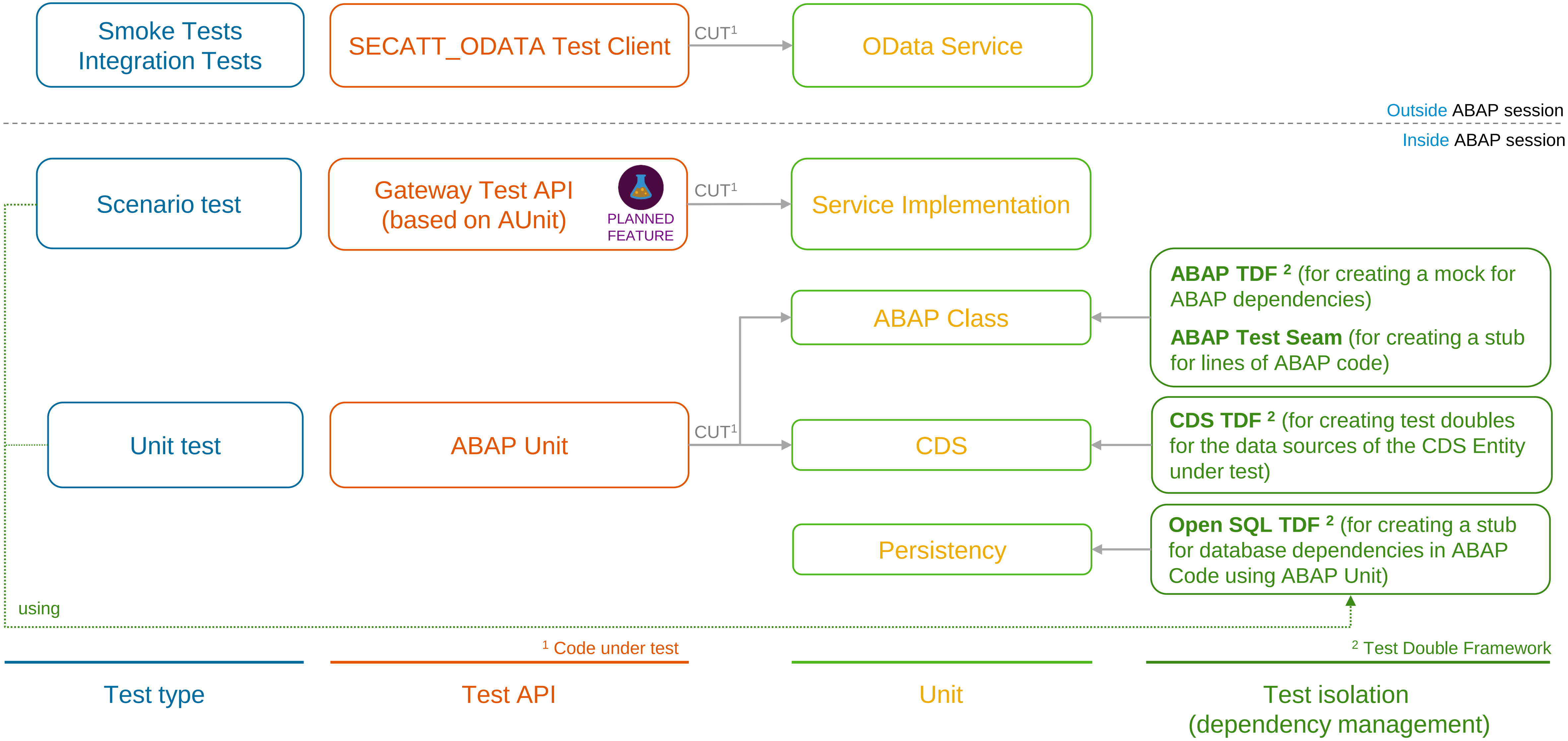
Be prepared for OData V4

- Use SAP Fiori programming model now
- Benefit from minimal TCD for OData V4 enablement

Testability



Testability - Overview



Smoke Tests
Integration Tests

SECATT_ODATA Test Client

CUT[1]

OData Service

Outside ABAP session

Inside ABAP session

Scenario test

Gateway Test API
(based on AUnit)

PLANNED
FEATURE

CUT[1]

Service Implementation

ABAP TDF [2] (for creating a mock for ABAP dependencies)

ABAP Test Seam (for creating a stub for lines of ABAP code)

ABAP Class

Unit test

ABAP Unit

CUT[1]

CDS

CDS TDF [2] (for creating test doubles for the data sources of the CDS Entity under test)

Persistency

Open SQL TDF [2] (for creating a stub for database dependencies in ABAP Code using ABAP Unit)

using

[1] Code under test

[2] Test Double Framework

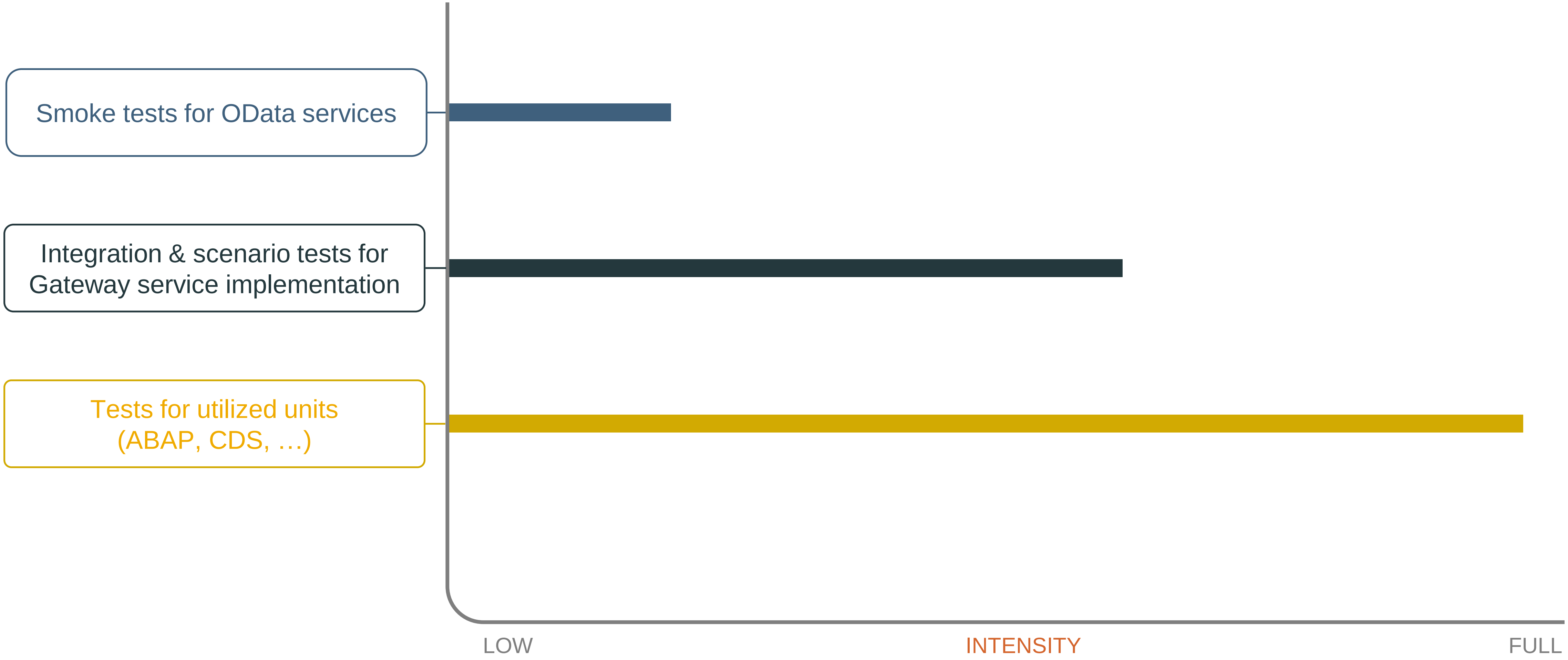
Test type

Test API

Unit

Test isolation
(dependency management)

Intensity of test means – Best practises



Access Authorizations

Access authorizations

USE CASE 1

As an **app developer** define instance-based and operation-based authorizations for my data model and its functionality

USE CASE 2

As an **app developer** using a CDS view I don't want to be forced to know which authorization objects I need to check. This should work out-of-the-box

USE CASE 3

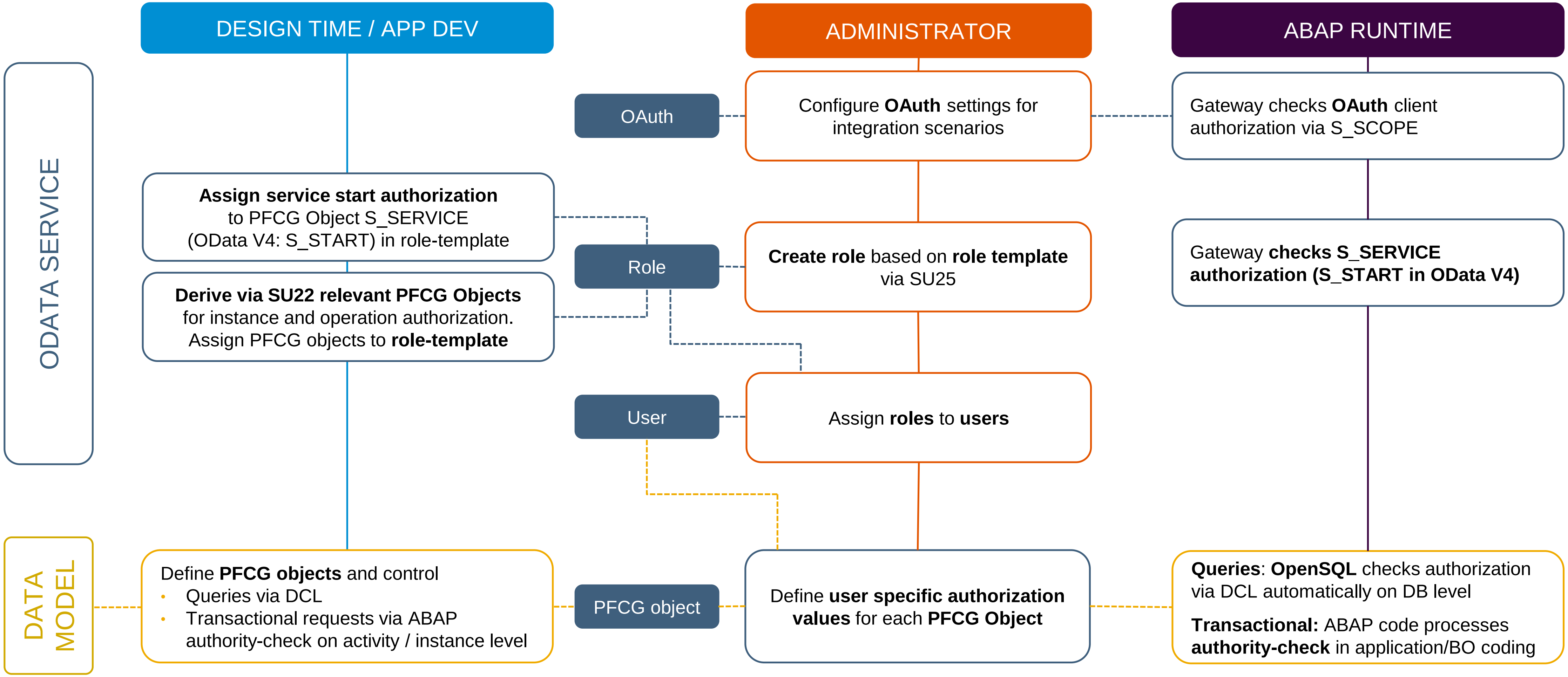
As an **app developer** I want to derive and declare which authorization objects are relevant for a specific OData service

USE CASE 4

As an **administrator** I want to

- Define roles based on the relevant authorization objects
- Define the authorization values for each user

Access authorization overview



ODATA SERVICE

DATA MODEL

DESIGN TIME / APP DEV

Assign service start authorization to PFCG Object S_SERVICE (OData V4: S_START) in role-template

Derive via SU22 relevant PFCG Objects for instance and operation authorization. Assign PFCG objects to **role-template**

Define **PFCG objects** and control
- Queries via DCL
- Transactional requests via ABAP authority-check on activity / instance level

OAuth

Role

User

PFCG object

ADMINISTRATOR

Configure **OAuth** settings for integration scenarios

Create role based on **role template** via SU25

Assign **roles** to **users**

Define **user specific authorization values** for each **PFCG Object**

ABAP RUNTIME

Gateway checks **OAuth** client authorization via S_SCOPE

Gateway **checks S_SERVICE authorization (S_START in OData V4)**

Queries: **OpenSQL** checks authorization via DCL automatically on DB level

Transactional: ABAP code processes **authority-check** in application/BO coding

Demo

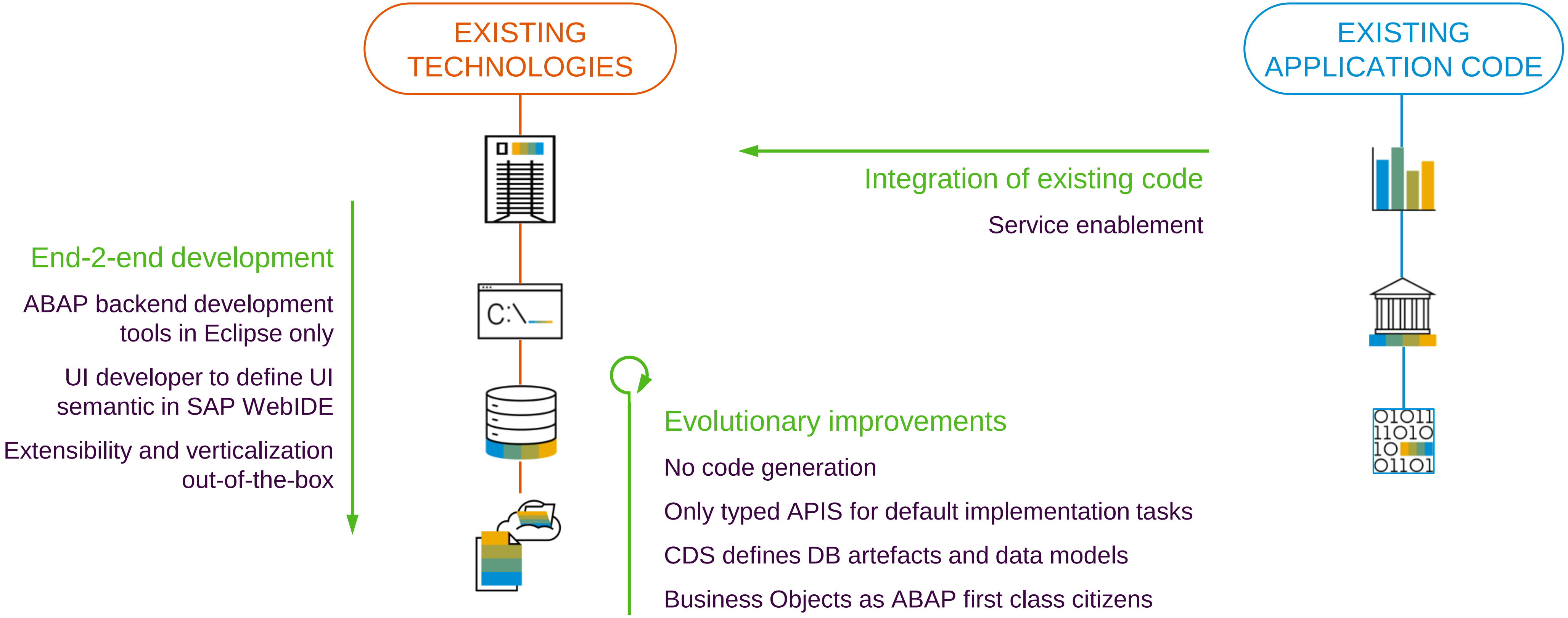
Access authorizations

OUTLOOK AND ROADMAP

THE RESTFUL ABAP PROGRAMMING MODEL

SAP’s approach for optimizing the total cost of development

PLANNED INNOVATION



EXISTING TECHNOLOGIES

EXISTING APPLICATION CODE

Integration of existing code

Service enablement

End-2-end development

ABAP backend development tools in Eclipse only

UI developer to define UI semantic in SAP WebIDE

Extensibility and verticalization out-of-the-box

Evolutionary improvements

No code generation

Only typed APIS for default implementation tasks

CDS defines DB artefacts and data models

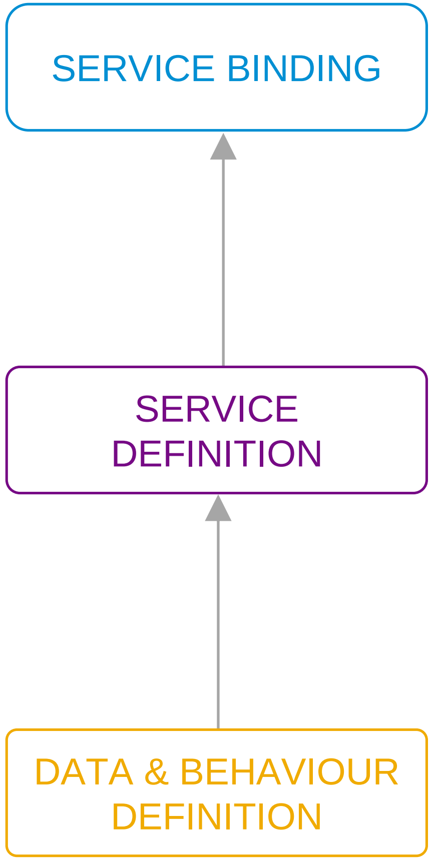
Business Objects as ABAP first class citizens

❯ RESTful ABAP PROGRAMMING MODEL

RESTful ABAP programming model big picture

The conceptual view

PLANNED INNOVATION



Entry-point of service implementation

Expose service definition protocol specifically

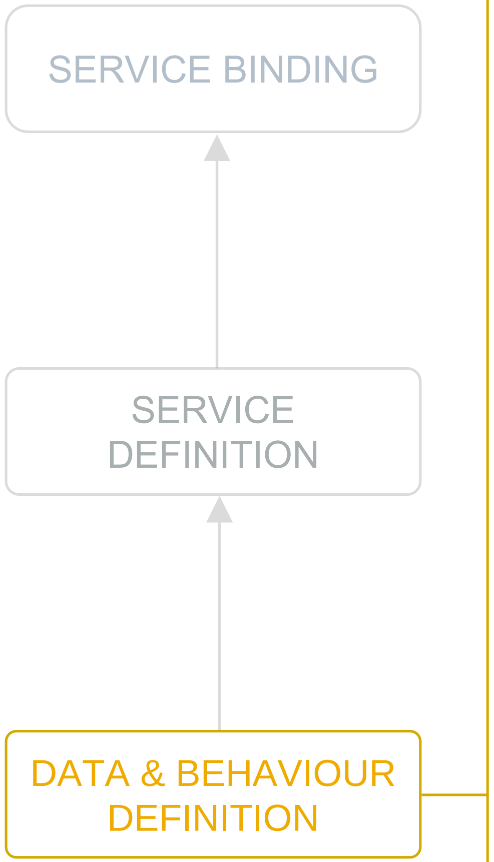
Service definition via projection of data and behaviour

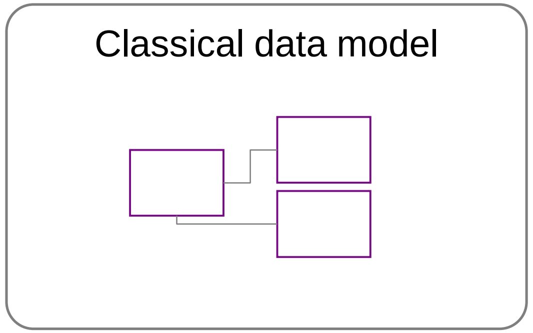
Universe of transactional & analytical data models and their business logic

RESTful ABAP programming model big picture

Data model: CDS entities

PLANNED INNOVATION

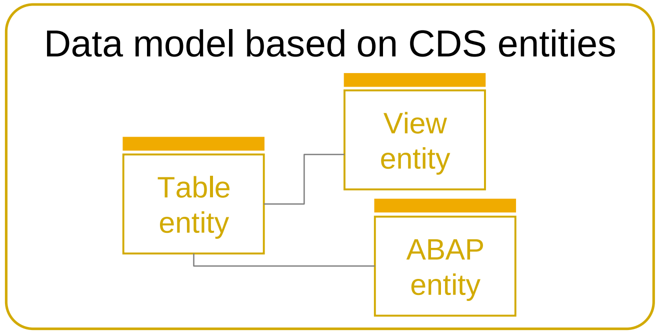




No separation of signature and implementation

Implementation currently always database dependent

today



CDS signatures for separation of signature (data modelling) and implementation (e.g. database view)

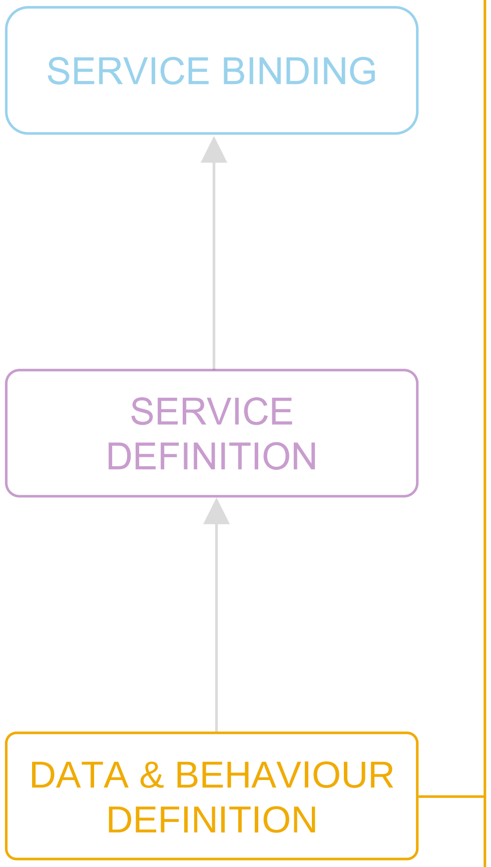
ABAP implemented entities via CDS custom entities

future

PLANNED INNOVATION

RESTful ABAP programming model big picture

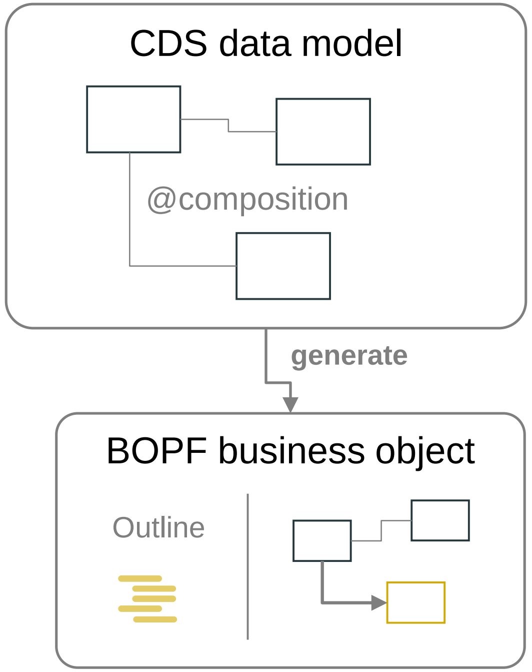
Data model: Business objects



SERVICE BINDING

SERVICE DEFINITION

DATA & BEHAVIOUR DEFINITION



CDS data model

@composition

generate

BOPF business object

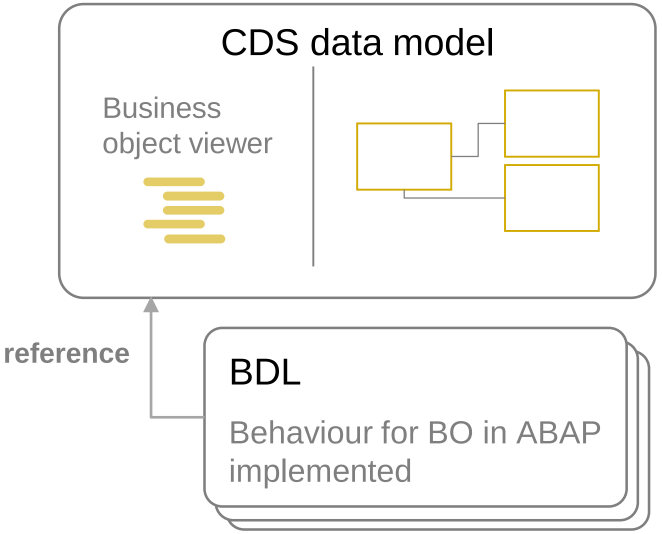
Outline

Development flow not intuitive
(CDS and BOPF tooling)

Generation of artefacts
(BOPF metadata generation based on CDS artefacts)

Untyped signature
(BOPF consumer and provider API)

today



CDS data model

Business object viewer

reference

BDL

Behaviour for BO in ABAP implemented

Clear development flow
(CDS for data modelling with ABAP language extensions for behaviour)

No redundancy and no generation of artefacts
(clear references on language level)

Typed APIs
(For BO consumer and provider)

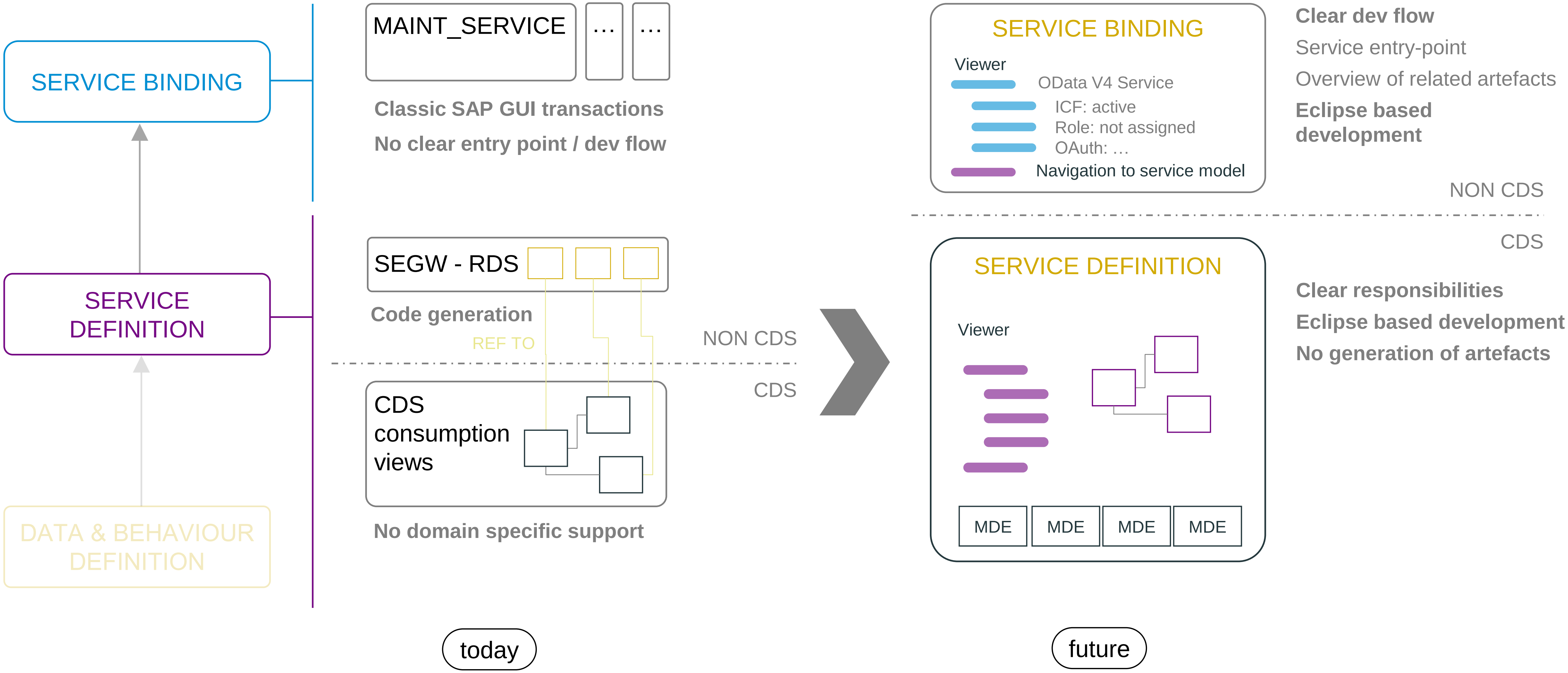
➔ **The Business Object becomes an ABAP first class citizen**
(Known by ABAP compiler, language and tools including static checks)

future

RESTful ABAP programming model big picture

Service definition and service binding

PLANNED INNOVATION



SERVICE BINDING

SERVICE DEFINITION

DATA & BEHAVIOUR DEFINITION

MAINT_SERVICE … …

Classic SAP GUI transactions

No clear entry point / dev flow

SEGW - RDS

Code generation

REF TO

NON CDS

CDS

CDS consumption views

No domain specific support

today

SERVICE BINDING

Viewer

OData V4 Service

ICF: active

Role: not assigned

OAuth: …

Navigation to service model

Clear dev flow

Service entry-point

Overview of related artefacts

Eclipse based development

NON CDS

CDS

SERVICE DEFINITION

Viewer

MDE MDE MDE MDE

Clear responsibilities

Eclipse based development

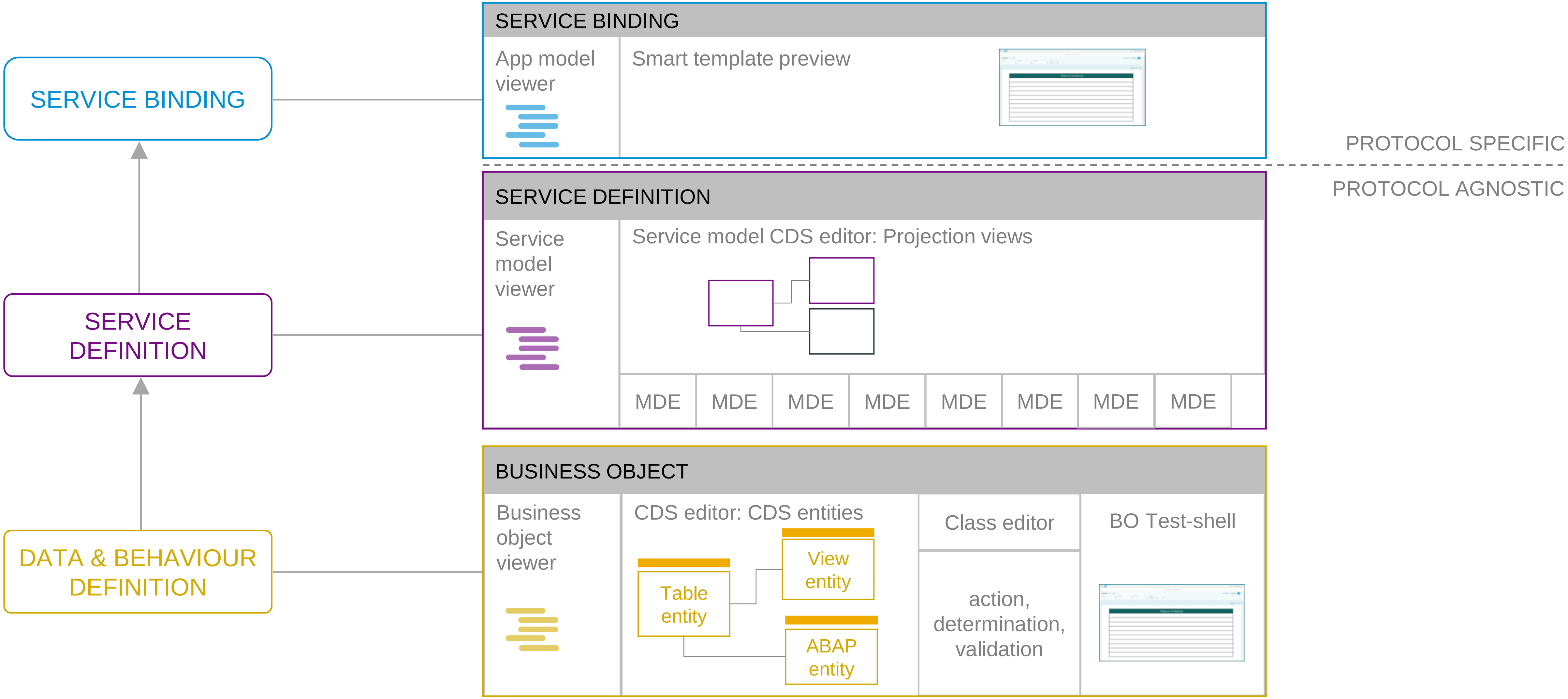
No generation of artefacts

future

RESTful ABAP programming model big picture

Summary

PLANNED INNOVATION



SERVICE BINDING

SERVICE DEFINITION

DATA & BEHAVIOUR DEFINITION

SERVICE BINDING

App model viewer

Smart template preview

PROTOCOL SPECIFIC

PROTOCOL AGNOSTIC

SERVICE DEFINITION

Service model viewer

Service model CDS editor: Projection views

MDE MDE MDE MDE MDE MDE MDE MDE

BUSINESS OBJECT

Business object viewer

CDS editor: CDS entities

Table entity

View entity

ABAP entity

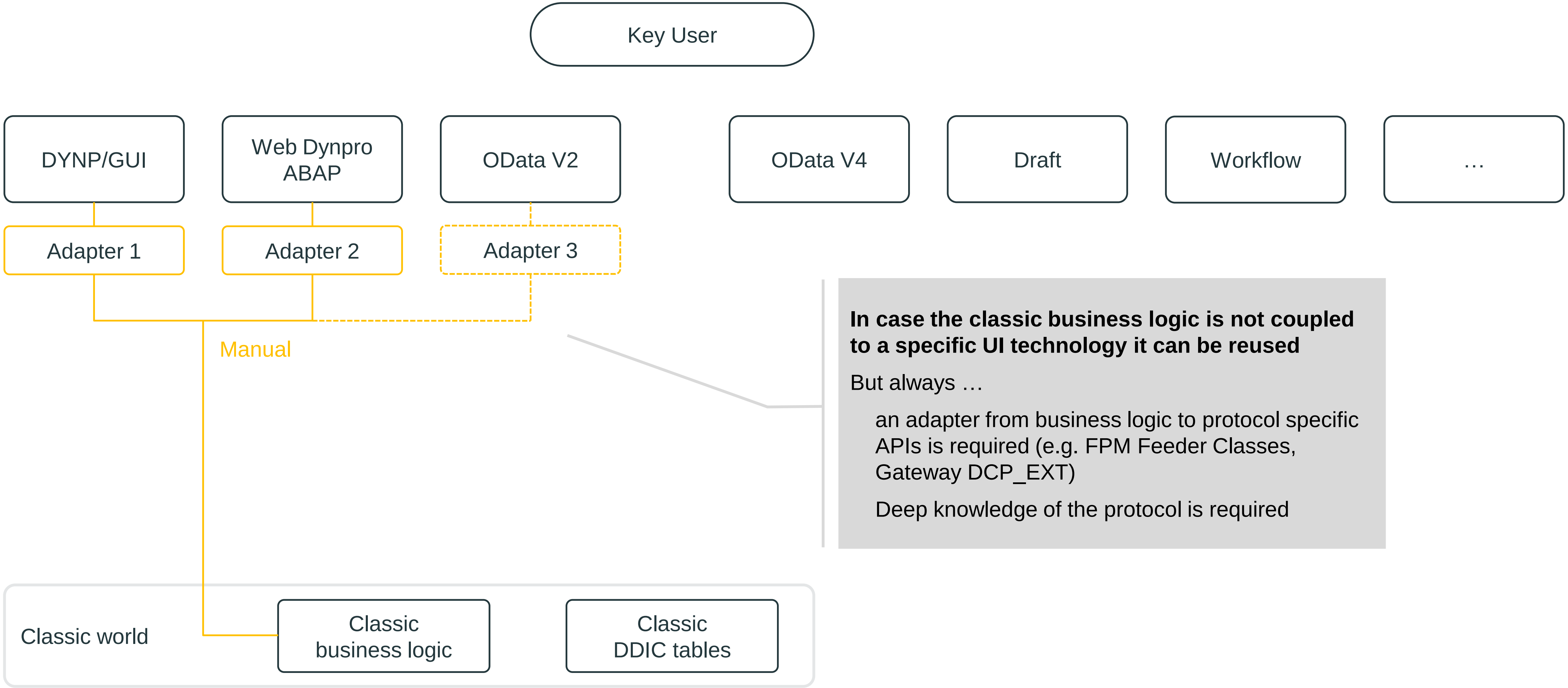
Class editor

action, determination, validation

BO Test-shell

RESTful ABAP programming model service enablement

PLANNED INNOVATION



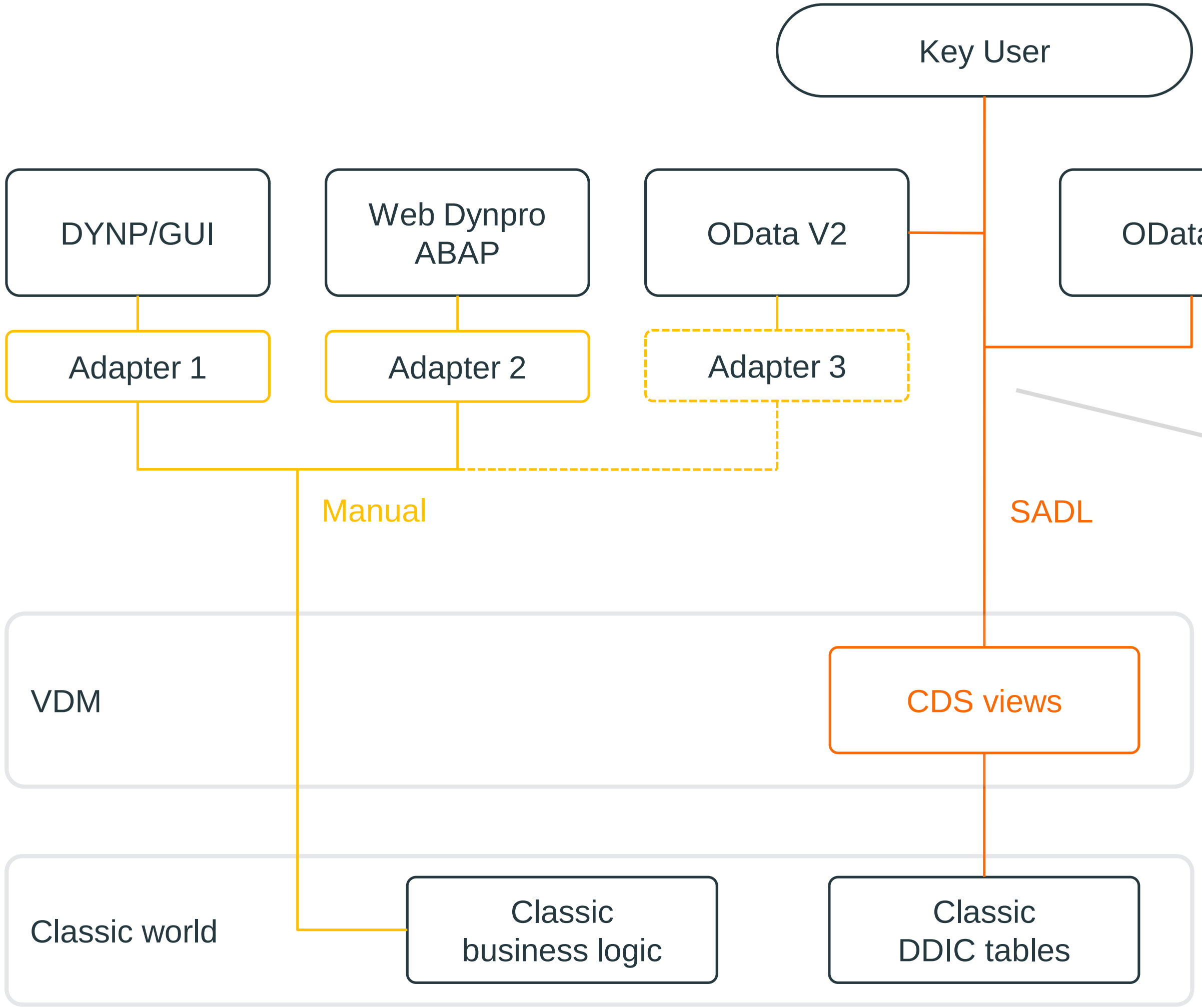
In case the classic business logic is not coupled to a specific UI technology it can be reused

But always …

- an adapter from business logic to protocol specific APIs is required (e.g. FPM Feeder Classes, Gateway DCP_EXT)
- Deep knowledge of the protocol is required

RESTful ABAP programming model service enablement

PLANNED INNOVATION



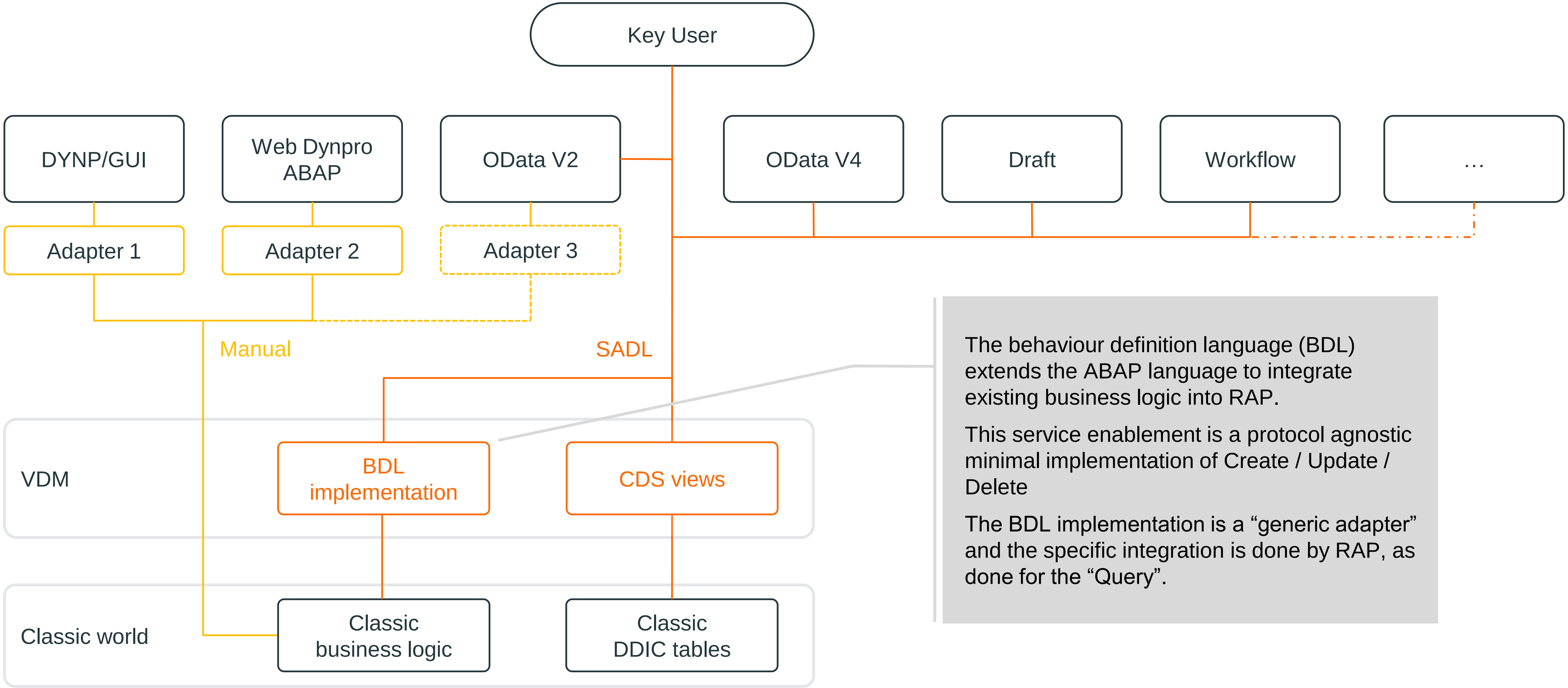
With CDS and SADL it is common sense that the “query/read” is always implemented generically done by experts

Key user,.. can easily create new read APIs based on the VDM

What about the transactional changes?

RESTful ABAP programming model service enablement

PLANNED INNOVATION



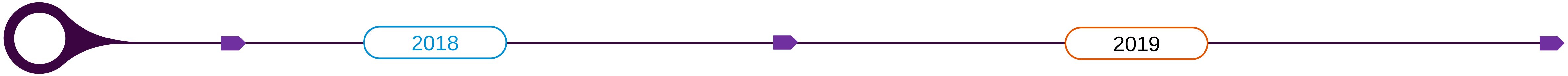
The behaviour definition language (BDL) extends the ABAP language to integrate existing business logic into RAP.

This service enablement is a protocol agnostic minimal implementation of Create / Update / Delete

The BDL implementation is a “generic adapter” and the specific integration is done by RAP, as done for the “Query”.

Main investment areas for the ABAP programming model

PLANNED INNOVATION



2018

2019

A2X COMMUNICATION

- Integration of legacy code into the RESTful ABAP Programming Model
- First end-2-end OData V4 support
- OData clients

NON FUNCTIONAL IMPROVEMENTS

- Focus on security
- Focus on testability
- Supportability

ANALYTICAL SAP FIORI APPS

- Further harmonization of analytics and transactional applications
- OData V4 enabling for analytical apps

TRANSACTIONAL SAP FIORI APPS

- Business Objects as ABAP first class citizen

NON FUNCTIONAL IMPROVEMENTS

- Focus on supportability
- Focus on E2E development flow

SAP TechEd Online / Community

Access replays of

- Keynotes
- SAP TechEd live interviews
- Select lecture sessions

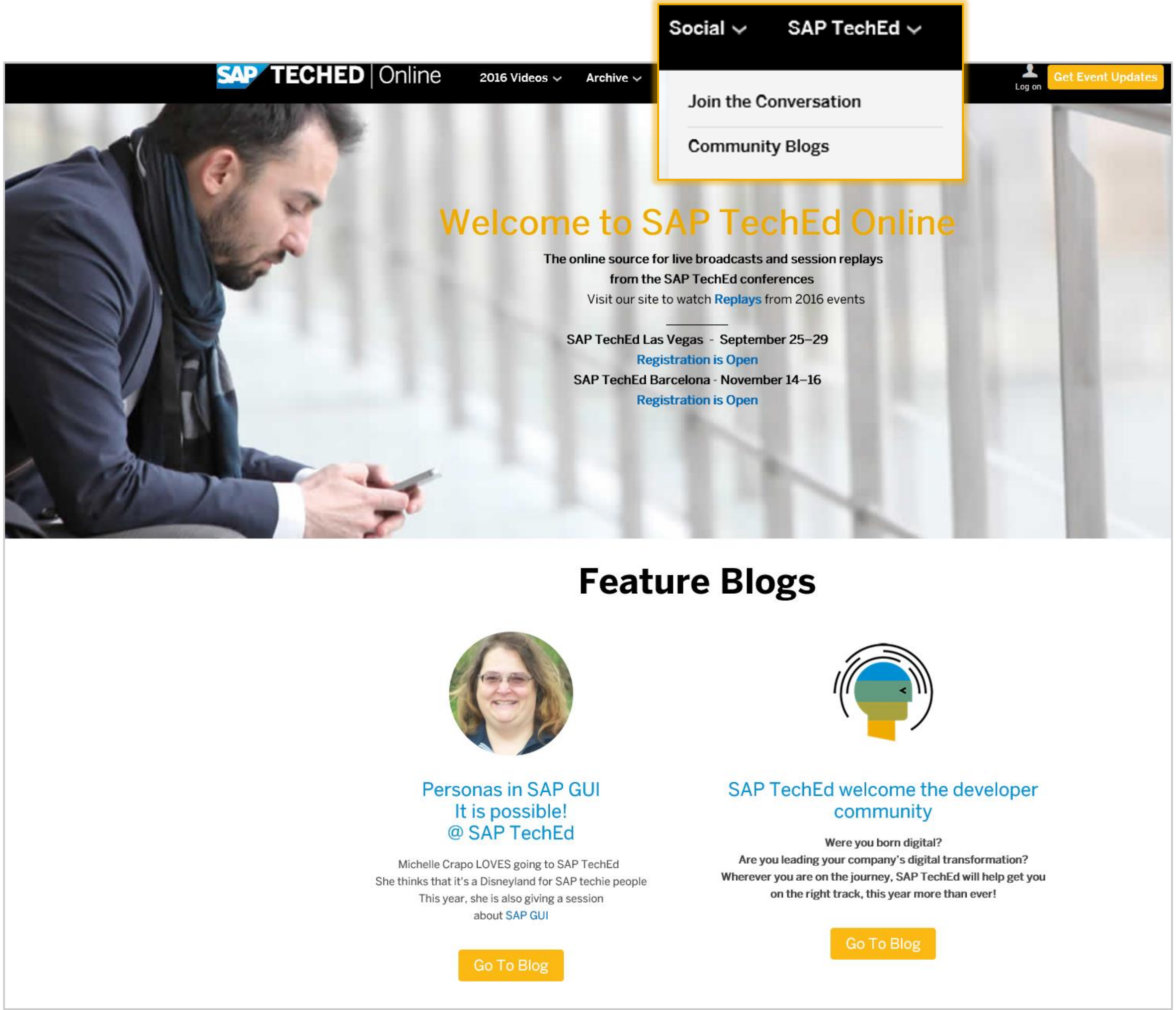
http://sapteched.com/online

Continue your SAP TechEd discussion after the event within the SAP TechEd Community!

- Read and reply to blogposts
- Ask your questions
- Join conversations

sap.com/community

See all SAP TechEd Blogposts



Welcome to SAP TechEd Online

The online source for live broadcasts and session replays
from the SAP TechEd conferences
Visit our site to watch Replays from 2016 events

SAP TechEd Las Vegas - September 25–29
Registration is Open
SAP TechEd Barcelona - November 14–16
Registration is Open

Feature Blogs

Personas in SAP GUI
It is possible!
@ SAP TechEd

Michelle Crapo LOVES going to SAP TechEd
She thinks that it's a Disneyland for SAP techie people
This year, she is also giving a session
about SAP GUI

SAP TechEd welcome the developer community

Were you born digital?
Are you leading your company's digital transformation?
Wherever you are on the journey, SAP TechEd will help get you
on the right track, this year more than ever!

Further information

Related SAP TechEd sessions

S4H106	ABAP Channels: Overview and Usage Scenarios
S4H112	ABAP Strategy
S4H130	Custom Code Adaptation for SAP S/4HANA (Lecture)
S4H164	Custom Code Adaptation for SAP S/4HANA (Handson)
S4H165	Troubleshoot Your SAP Fiori App with ABAP Development Tools for Eclipse
S4H222	OData V4 Services
S4H231	Optimize your ABAP Code for SAP HANA
S4H232	Automated Testing Within the ABAP Programming Model for SAP S/4HANA
S4H269	Authorizations Within the ABAP Programming Model for SAP S/4HANA
S4H274	Modern ABAP with Eclipse
S4H276	Build a Fiori List Report App: ABAP Programming Model for SAP S/4HANA
S4H279	Build a Transactional Fiori App: ABAP Programming Model for SAP S/4HANA
S4H630	SAP Fiori: Development on Core Data Services
S4H836	SAP NetWeaver: Road Map and Transition to SAP S/4HANA
S4H837	Road Map Q&A: ABAP Platform
S4H839	Integration of OData and SAP Fiori

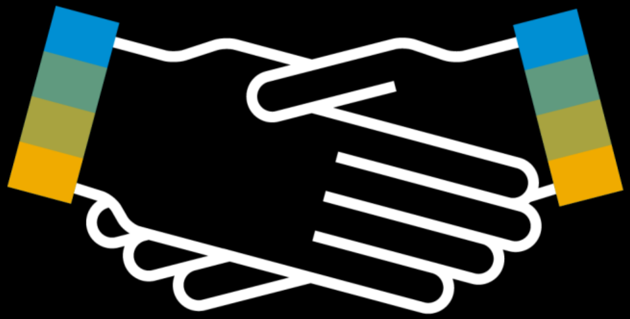
SAP Public Web

scn.sap.com
www.sap.com

SAP Education and Certification Opportunities

www.sap.com/education

Thanks for attending this session.



Feedback

Please complete your session evaluation for **S4H140**.

Presentation from SAP TechEd 2017

Contact information:

Marcel Hermanns

marcel.hermans@sap.com

Follow us

www.sap.com/contactsap

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platforms, directions, and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

THE BEST RUN SAP