Web Dynpro Componentization

SAP NetWeaver ’04

Bertram Ganz, Christian Georgi, Michael Wenz – SAP AG

SAP TechEd ‘05

SAP TECHED

Section Overview – Three Building Blocks

1. Component Basics
2. Component Usage Relations
3. Demo
4. Component-Based Application Architecture
5. Hands-On Excercise
6. NetWeaver Development Infrasturcture
7. Referencing Components
8. Component Interface Definitions
9. Hands-On Excercise

SAP TECHED

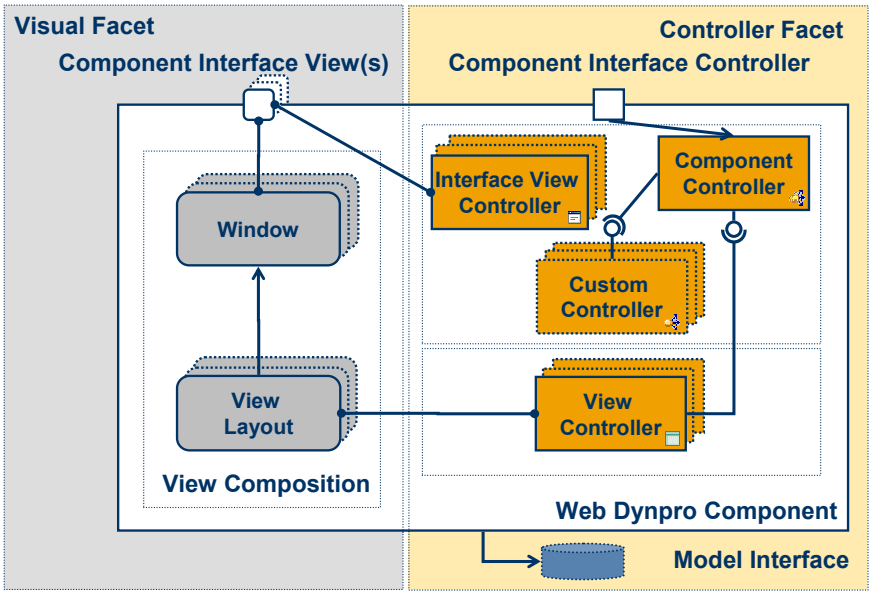
Web Dynpro Component Concept

Web Dynpro Components ...

1. are the fundamental blocks which build Web Dynpro applications
2. have a specific controller anatomy (white-box-approach)
3. have specific interfaces for fulfiling visual, programmatic and context-specific needs (black-box-approach)
4. are the only unit of reuse in Web Dynpro

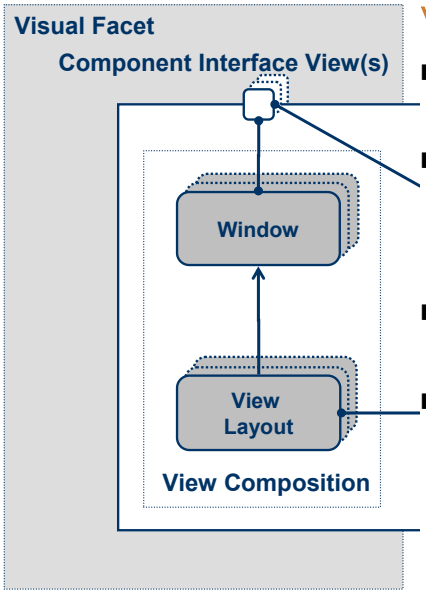
SAP TECHED

Web Dynpro Components – A White-Box Approach



SAP TECHED

Web Dynpro Components – A White-Box Approach



Visual Facet

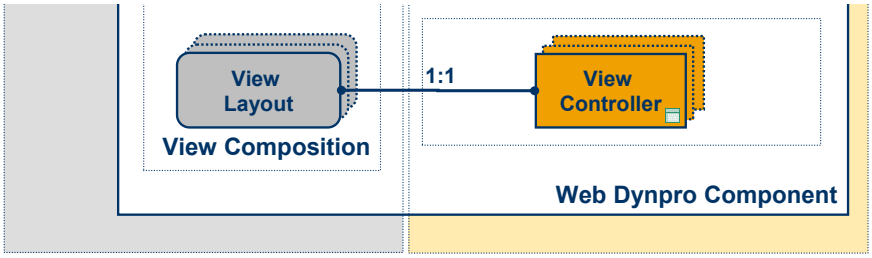
- The Component UI is modeled inside one or many windows
- A window contains the view composition (views, view sets and the navigation graph consisting of linked plugs)
- A view layouts contains a tree of Web Dynpro UI-elements
- There is a 1:1 relation between a window and its component interface view.

SAP TECHED

Web Dynpro Components – A White-Box Approach

View Layout and View Controller

- Every View-Layout has a corresponding View Controller
- The View-Controller is there for …
  - storing and referencing (UI-relevant) context data (data-binding, context mapping)
  - handling user actions
  - triggering navigation steps
  - dynamically modifying the view layout

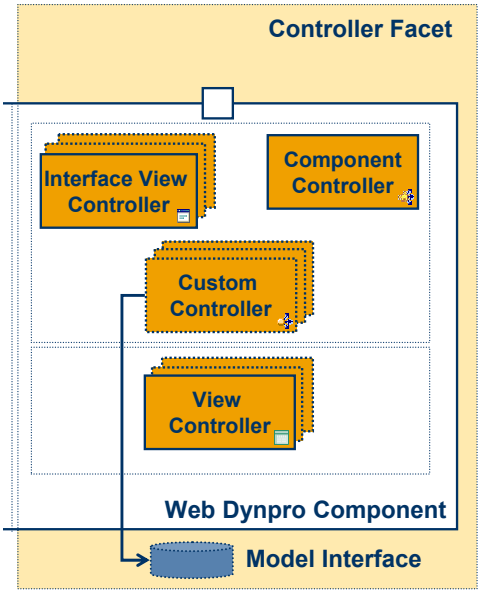


SAP TECHED

Web Dynpro Components – A White-Box Approach

Non-View Controllers

- Component Controller (1)
- Custom Controllers (0..n)
- Provide global services
- Store global context data
- Eventing
- Context-to-model binding
- Provide public interfaces to other controllers

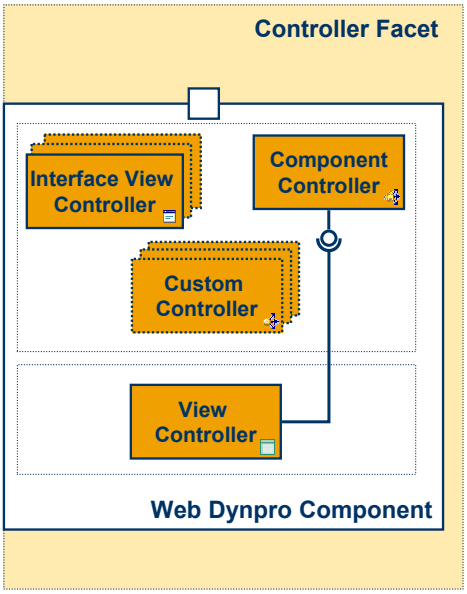


SAP TECHED

Web Dynpro Components – A White-Box Approach

Controller Interfaces

- Non-View Controllers can expose an interface (for accessing the context, calling methods and subscribing to events) to other controllers
- Controllers have to define a controller usage relation in order to access the interface of another controller
- View controllers do not expose such an interface to other controllers

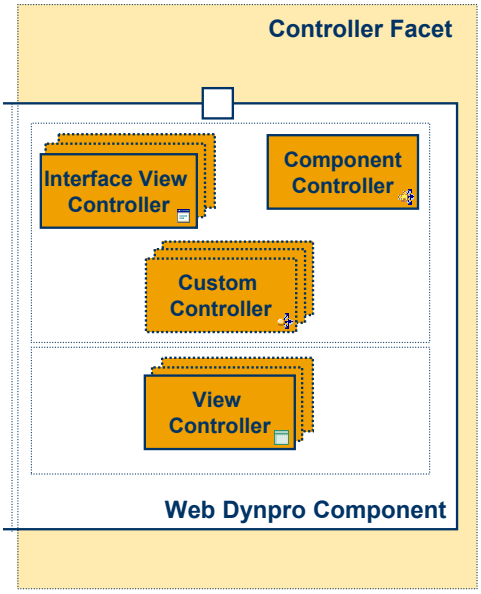


SAP TECHED

Web Dynpro Components – A White-Box Approach

Interface View Controllers

- Implement event handlers which are called when ..
  - starting (start-up plugs) Web Dypro applications
  - a component is reached via navigation (inbound plugs)
- Allow fireing outbound plugs (navigation)
- Allow firing exit plug
- Have no own context, public methods or events



SAP TECHED

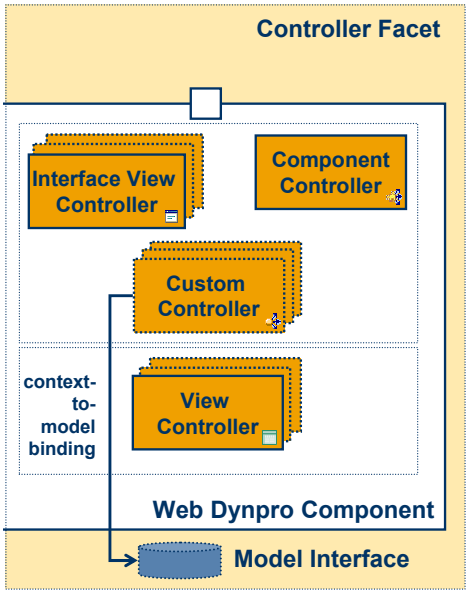
Web Dynpro Components – A White-Box Approach

Components and their Model Usages

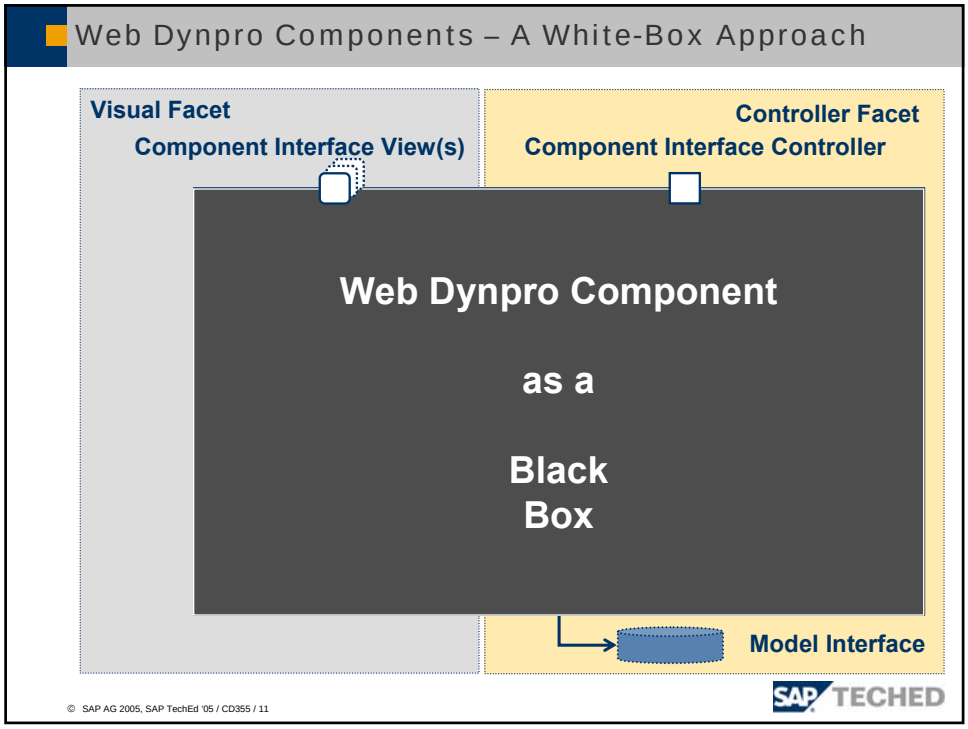
Proxys connect to the backend system like mySAP or Web Services

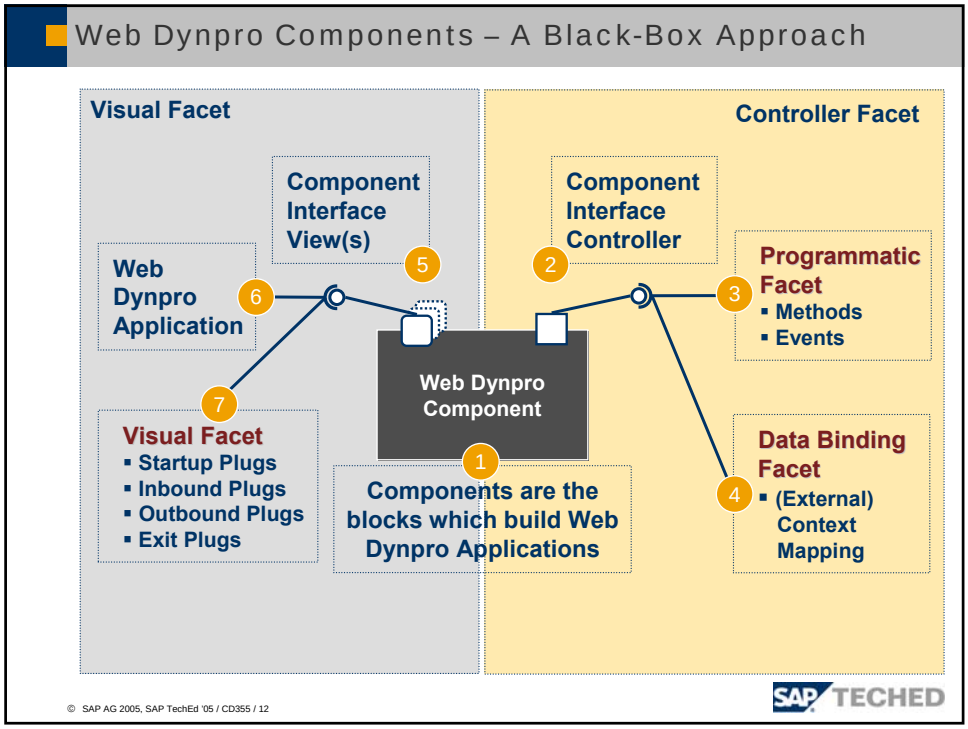
Model interfaces provide access to the backend

Data transfer between controller context and model objects is based on a **context-to-model binding** relation



SAP TECHED

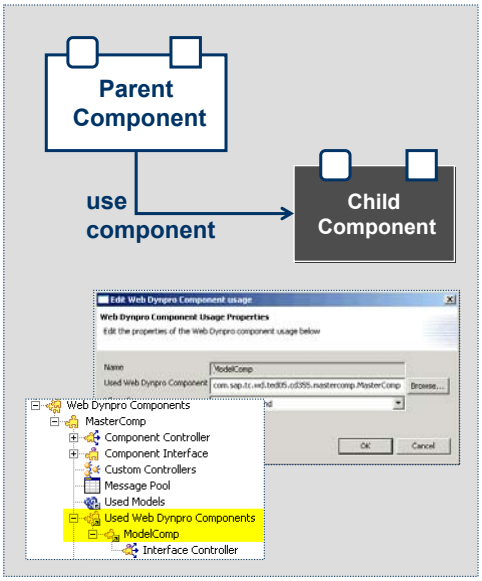

Web Dynpro Components – A White-Box Approach
Visual Facet
Component Interface View(s)
Controller Facet
Component Interface Controller
Web Dynpro Component
as a
Black
Box
Model Interface
© SAP AG 2005, SAP TechEd '05 / CD355 / 11
SAP TECHED


Web Dynpro Components – A Black-Box Approach
Visual Facet
Controller Facet
Component Interface View(s)
Component Interface Controller
Web Dynpro Application
Web Dynpro Component
Programmatic Facet
▪ Methods
▪ Events
Visual Facet
▪ Startup Plugs
▪ Inbound Plugs
▪ Outbound Plugs
▪ Exit Plugs
Data Binding Facet
▪ (External) Context Mapping
Components are the blocks which build Web Dynpro Applications
1
2
3
4
5
6
7
© SAP AG 2005, SAP TechEd '05 / CD355 / 12
SAP TECHED

Component Usage Relations

Component Usage

- The parent component must explicitly define a component usage relation for embedding another child component
- It acts like a variable referring to component or component interface definition
- At runtime it is associated with a concrete component instance

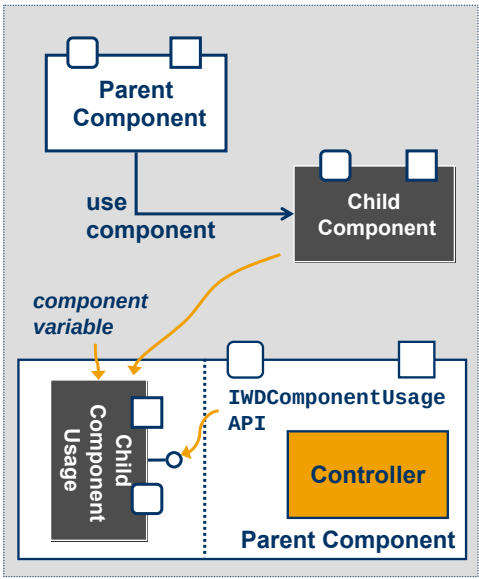


SAP TECHED

Component Usage Relations

Component Usage

- Declared inside a Web Dynpro Component (parent)
- Provides two alternative ways of lifecycle management for the associated component instance
- Handle to component factory
- Related API: IWDComponentUsage

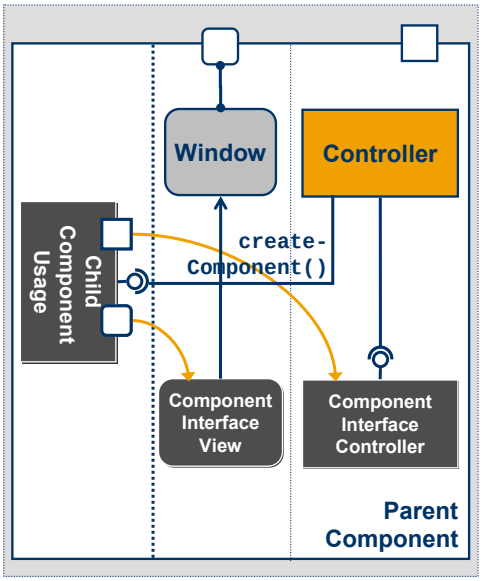


SAP TECHED

Component Usage Relations

Component Usage

- Provides access to instance of embedded child component
- Enables access to the interface controller
- Enables embedding the component's interface view(s)
  - Optional: e.g. faceless components do not have interface views
  - In the embedding window, interface views behave like "normal" views



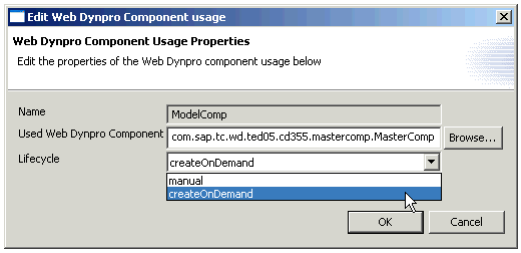
SAP TECHED

Managing Component Lifecycles

Lifecycle Property of a Component Usage

`createOnDemand`: Web Dynpro Runtime automatically creates a component instance on demand (e. g. when the component gets visible on the UI) and destroys it again

`manual`: component instance lifecycle must be programmatically managed by the the application developer using the **`IWDComponentUsage`**-API



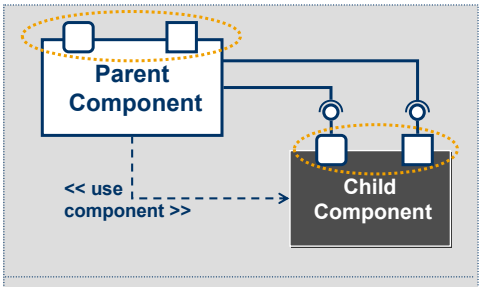
SAP TECHED

Web Dynpro Component Diagram

We propose the following diagram type for visualizing Web Dynpro component relations (based on the UML 2.0)

Ports of a Web Dynpro Component

- A Web Dynpro component has special entry points (ports) between the component's outer environment an its inner part.
- A port can be seen as the specification of an interaction point on the hull of a Web Dynpro component.



Visual Ports:
Component Interface Views **(0..n)**
(inbound plugs, outbound plugs, paramters for the inbound plug event handler)

Service Port:
Component Interface Controller **(1)**
(context, methods, events)

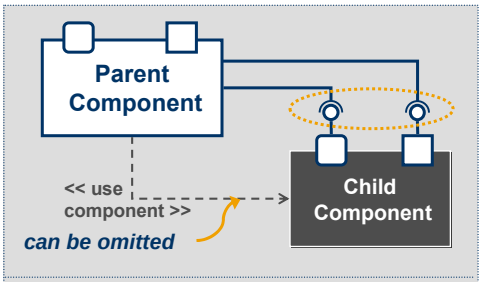
SAP TECHED

Web Dynpro Component Diagram

Connectors between Web Dynpro Components

- A connector links a Web Dynpro Component with a port of another component
- The connector visualized by linking a socket symbol with a ball symbol.
- The socket symbol starts from the component user
- The ball symbol starts from the component interface port of the used component

Note:
The graphical representation must not explicitly contain the relation dependency (usage relation)



Socket notation for a required interface

IExternal Interface (Comp. Interf. C:
Ball notation for an exposed interface

SAP TECHED

Section 2 – Component Based Application Architecture

1
- Component Basics
- Component Usage Relations
- Demo

2
- Component-Based Application Architecture
- Hands-On Excercise

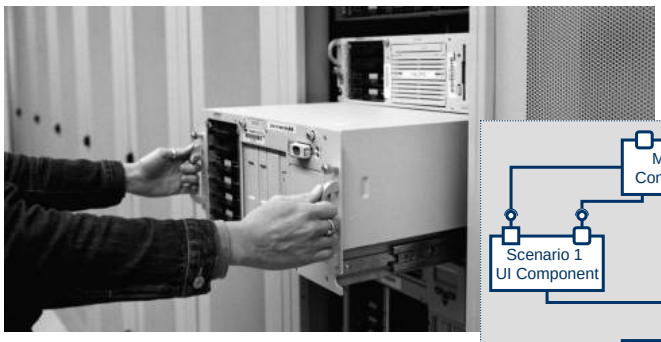
3
- NetWeaver Development Infrasturcture
- Referencing Components
- Component Interface Definitions
- Hands-On Excercise

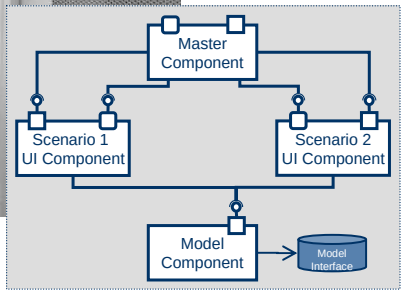
SAP TECHED

Breaking Applications into Parts: Components

Components are the blocks

which build Web Dynpro Applications.

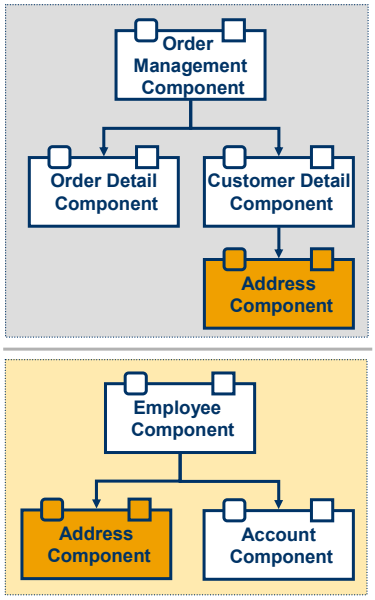




SAP TECHED

Breaking Applications into Parts: Components

Components hide implementation behind an interface (black box approach)



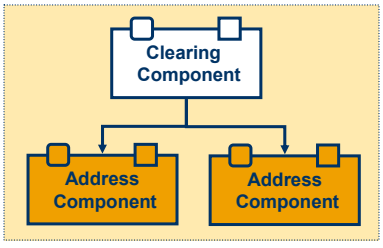
Benefits

- Break down applications into manageable parts
  - Distribute work between developers
  - Stay independent of implementation details (black box approach)
- Construct reusable entities
  - A component can be used in several other components

SAP TECHED

Breaking Applications into Parts: Components

Components concentrate reuse in one element (components, not views ...)

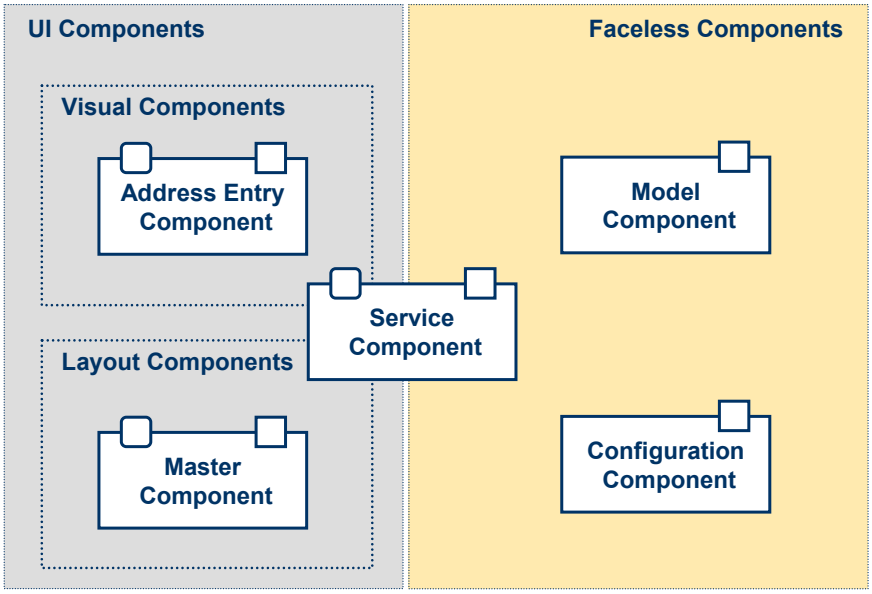


Benefits

- Multi-instance capability
  - A component can embed another one more than once
- Relieve developer from instance management
  - No multi-instance capability for parts of a component (views, controllers)
  - Create (non-view) controllers on demand
  - Get access to controller by specifying its type name

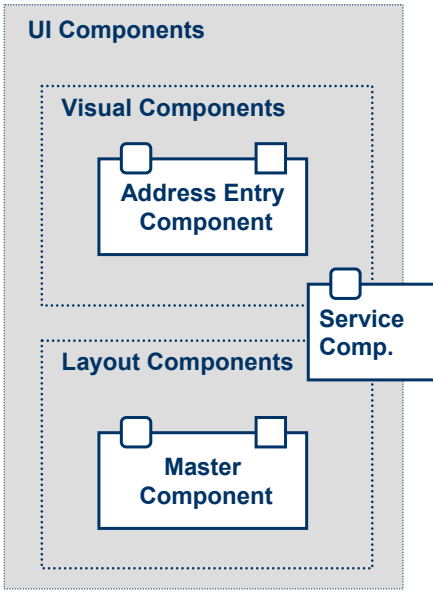
SAP TECHED

Web Dynpro Component Types



Component types are only an application design concept

Web Dynpro Component Types – UI-Components



User Interface Components

Visual Components:

Provide user interaction

- **Example: address entry form**

Layout Components:

- **Arrange the visual components into screens**
- **Only needed for providing uniformly structured screens in large applications**

Service Components (with UI):

Provide reusable UI service

Component types are only an application design concept

Web Dynpro Component Types

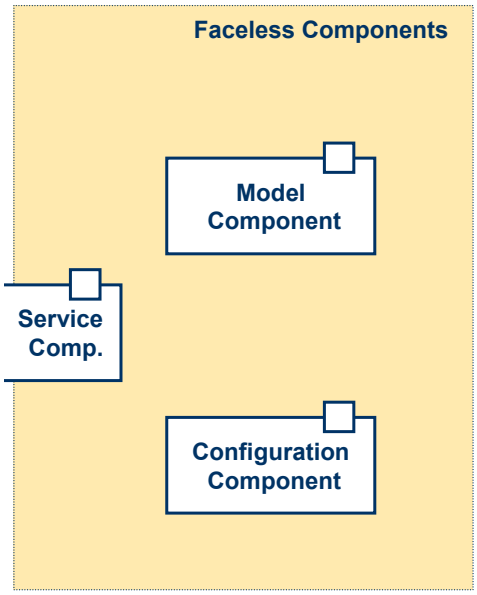
Faceless Component

Model Components:

Embed models and expose them in their component interface context

Configuration Components:

Special type of model component for storing and retrieving configuration data



Component types are only an application design concept

SAP TECHED

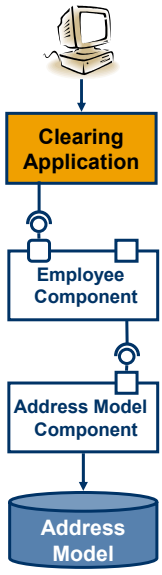
Components: Component Types

Use specialized Web Dynpro Component Types

- User Interface Components
  - Visual components
  - Layout components
- Faceless / Model Components
- Service Components

Component types are only an application design concept

Keep UI components and faceless components independent of each other to be able to exchange the other part if this is necessary



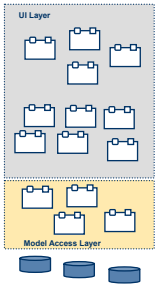
Component types are only an application design concept

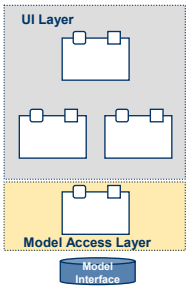
SAP TECHED

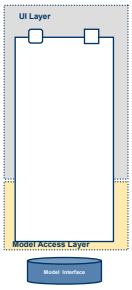
Allowing Reuse and Extensibility: Granularity

Finding the right component granularity

1 Too many small components	2 Right component granularity	3 Too large components
the application might not be performant resulting „sea“ of components might reduce maintainability	best reusability, extensibility and maintainability allows distributing work between several developers	reusability is reduced distributing work between several developers is harder

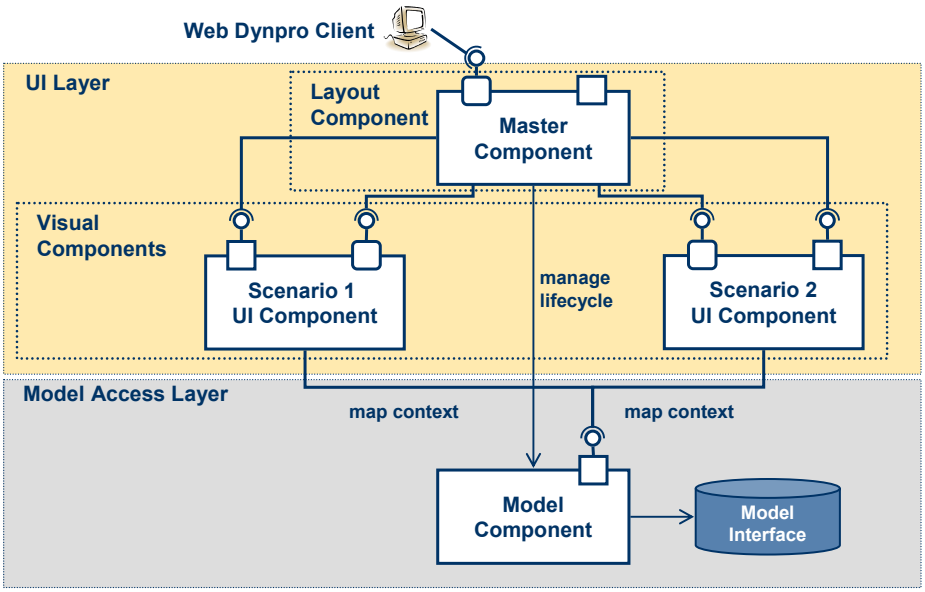






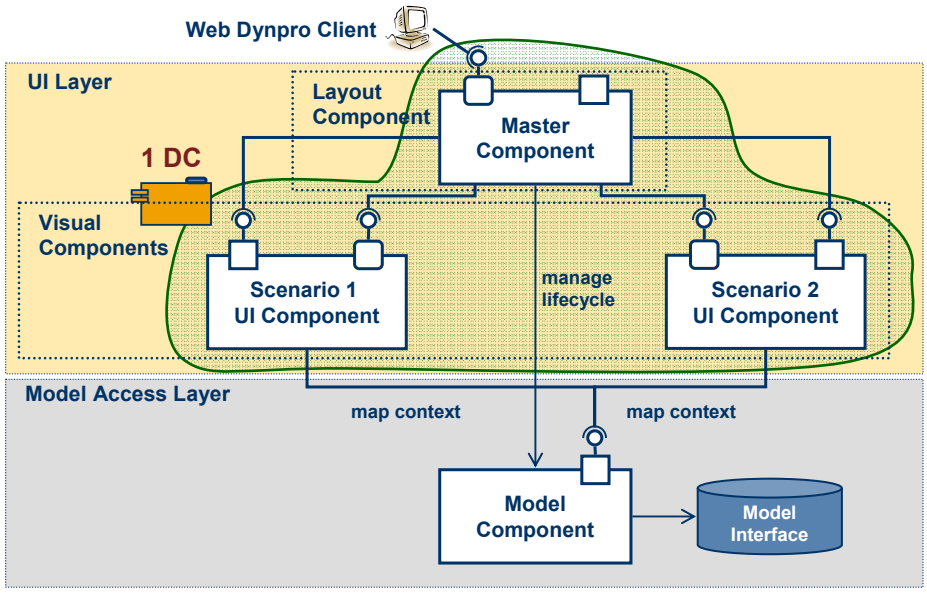
SAP TECHED

Example: A Simple Web Dynpro Component Architecture



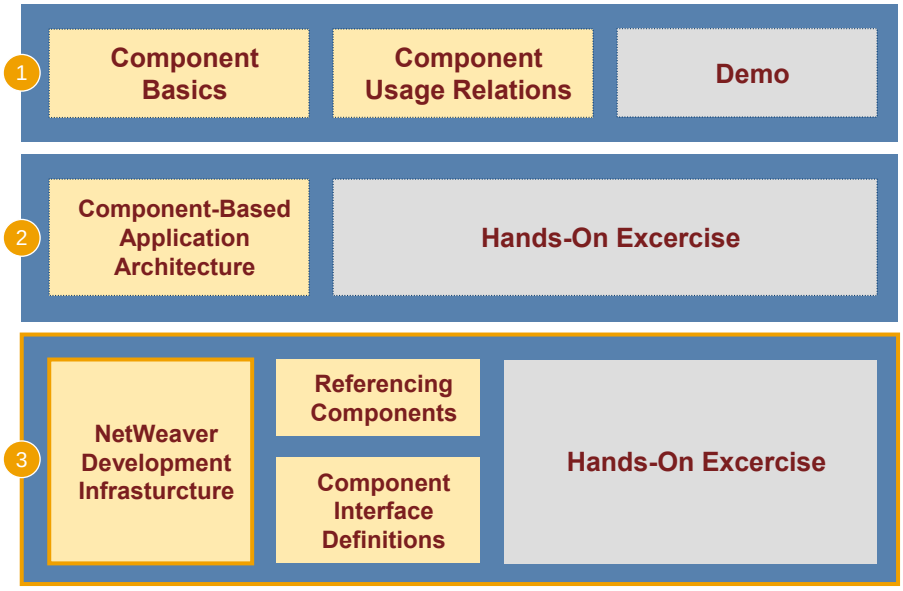
SAP TECHED

Example: A Simple Web Dynpro Component Architecture

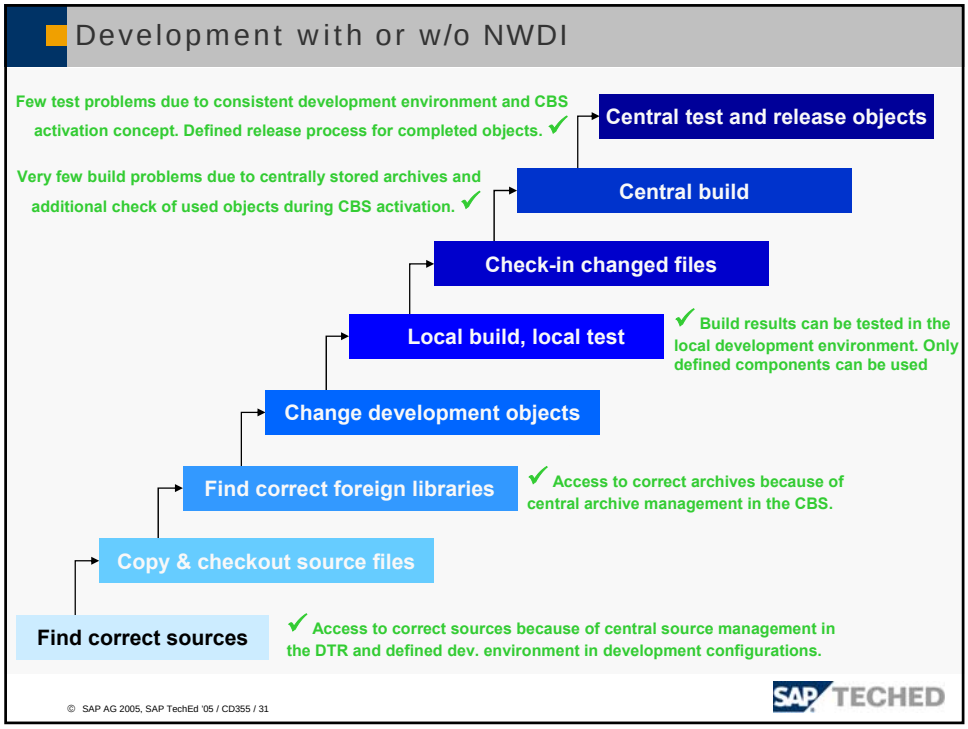


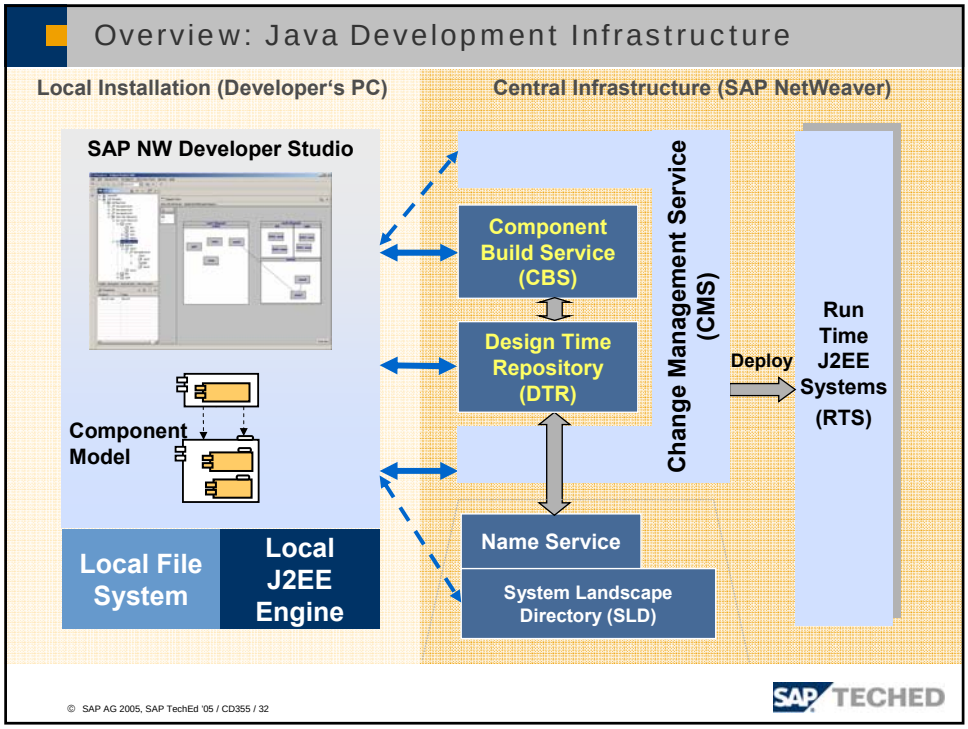
SAP TECHED

Section 2 – Component Based Application Architecture



SAP TECHED


Development with or w/o NWDI
Few test problems due to consistent development environment and CBS activation concept. Defined release process for completed objects.
Central test and release objects
Very few build problems due to centrally stored archives and additional check of used objects during CBS activation.
Central build
Check-in changed files
Local build, local test
Build results can be tested in the local development environment. Only defined components can be used
Change development objects
Find correct foreign libraries
Access to correct archives because of central archive management in the CBS.
Copy & checkout source files
Find correct sources
Access to correct sources because of central source management in the DTR and defined dev. environment in development configurations.
© SAP AG 2005, SAP TechEd '05 / CD355 / 31
SAP TECHED


Overview: Java Development Infrastructure
Local Installation (Developer‘s PC)
Central Infrastructure (SAP NetWeaver)
SAP NW Developer Studio
Component Model
Local File System
Local J2EE Engine
Component Build Service (CBS)
Design Time Repository (DTR)
Change Management Service (CMS)
Name Service
System Landscape Directory (SLD)
Deploy
Run Time J2EE Systems (RTS)
© SAP AG 2005, SAP TechEd '05 / CD355 / 32
SAP TECHED

DTR – Basics

Distributed, concurrent development

- Distributed versioning that spans repositories
- Conflict detection and resolution

Files and Folders

- DTR knows only files and folders
- No knowledge about objects of programming model (classes, tables, screens etc)

Check in/check out model

- Files are checked out from DTR and modified "offline"
- Files are checked in again when change is complete

SAP TECHED

CBS – Features

- Storage of all archives of a software component
- Immediate build of changes on request by developers
- Automatic rebuild of dependent DCs
- Provides access to latest build results for developers
- Web UI (administration)
- Parallel execution of build requests

SAP TECHED

CBS – Activation Concept

Inactive Workspace

- Changes are done only here

Active Workspace

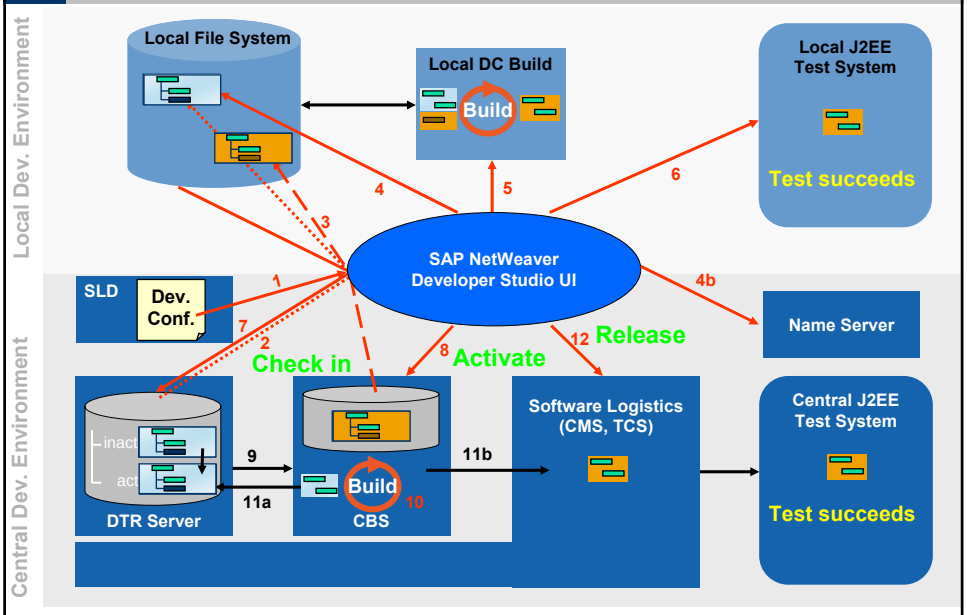
- Has only successfully built sources
- Always in sync with archives available from CBS

Activating Changes

- Additional step after check in
- Each code line (DEV, CONS) represented by two DTR workspaces

SAP TECHED

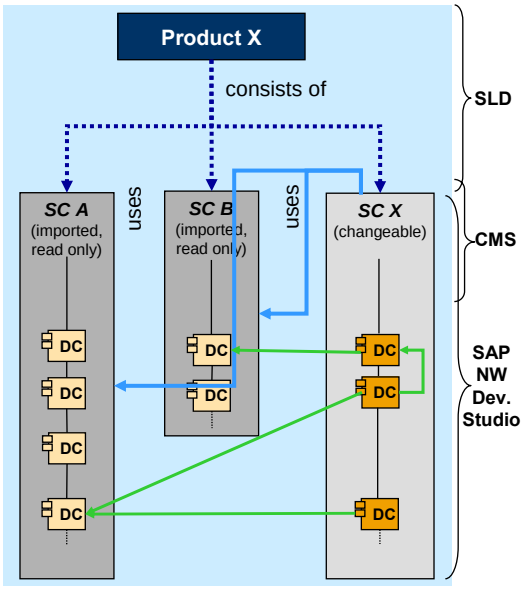
NWDI Based Development: Process



SAP TECHED

Structuring Applications Using the Component Model

1. Define a product
2. Define used SCs (in the development configuration)
3. Define new SCs
4. Define top-level DCs
5. Create DCs (default use dependencies to DCs of used SCs are set automatically)
6. Create development objects
7. Create public parts of DCs
8. Set use dependencies

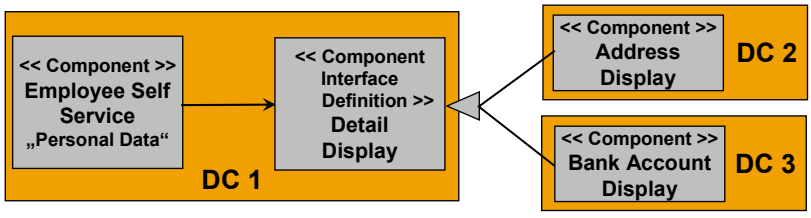


SAP TECHED

Allowing Reuse and Extensibility: Delivery

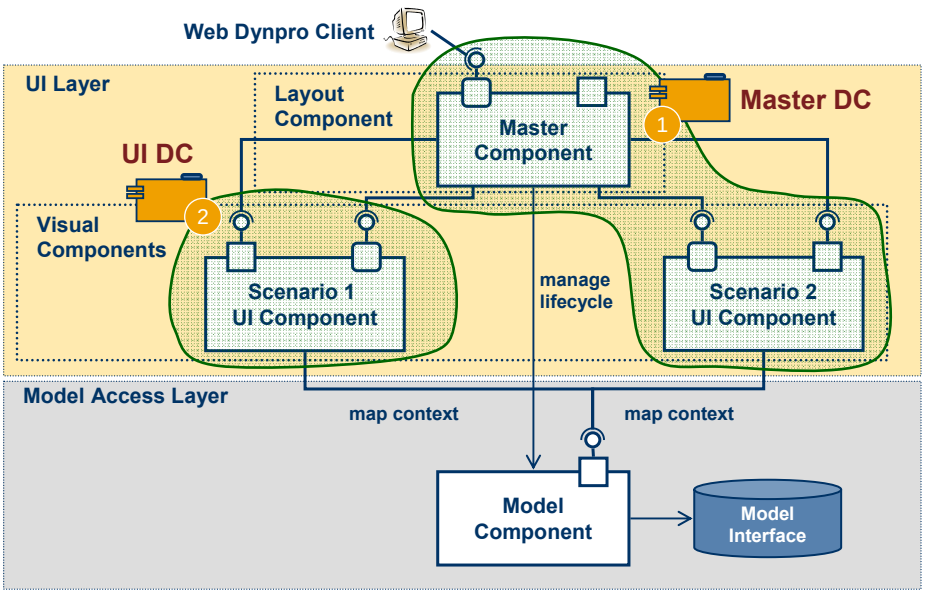
Package Web Dynpro Components into Development Components

- A development component is
  - unit for packaging, building and transportation in SAP's infrastructure (NWDI)
  - mapped to a project in the NetWeaver Developer Studio
- Distribute objects over development components (reuse, size)
  - component interface definitions and their implementations
  - models
- Generally keep development components small
  - improved build and deploy performance
  - watch out for increased overhead for maintaining references between them



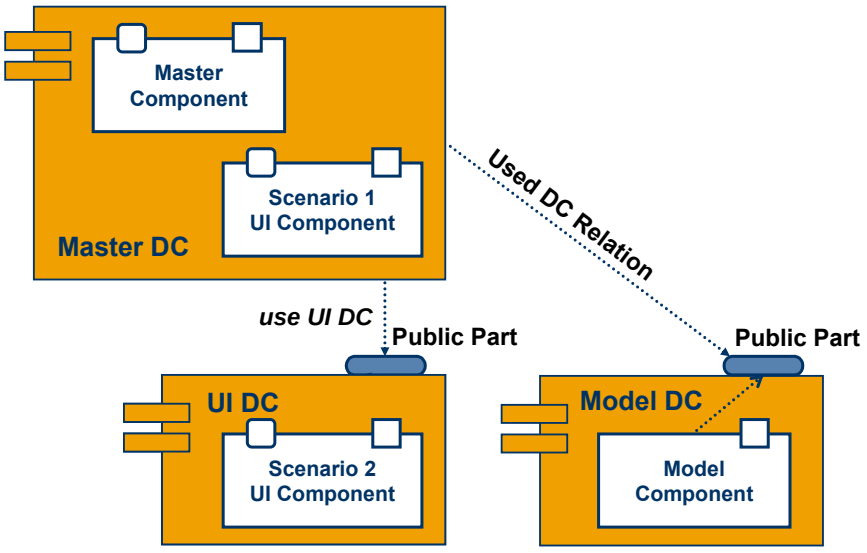
SAP TECHED

A Simple Web Dynpro Component Architecture



SAP TECHED

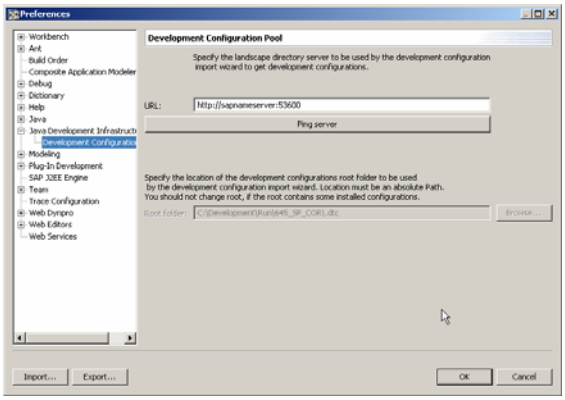
A Simple Web Dynpro Component Architecture



SAP TECHED

SLD Location

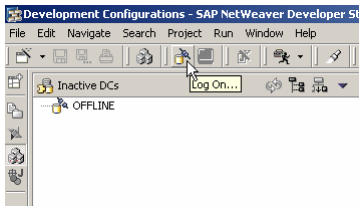
- The URL for the system landscape directory has to be entered in the IDE preferences
- Enable IDE to access defined configurations stored within the SLD



SAP TECHED

Logon to Development Infrastucture

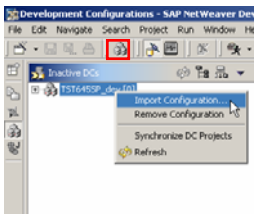
- Logon needed to work with Java development infrastructure
  - After every start of the IDE
  - Preference setting to enable auto logon after IDE start

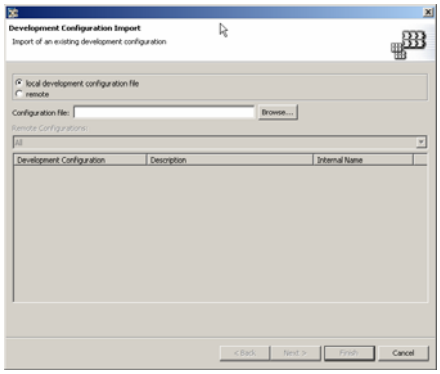


SAP TECHED

Import configuration

- Start wizard from context menu or toolbar
- Import either
  - Locally stored configuration file
  - Configuration from SLD (remote)



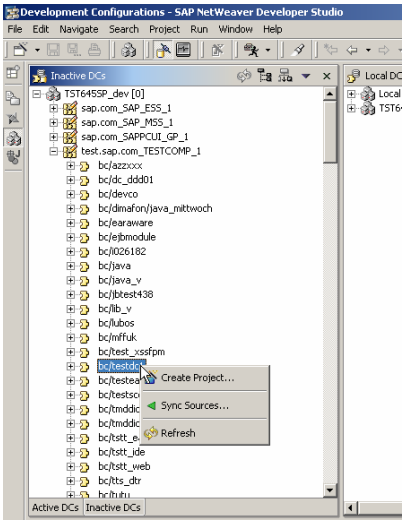


SAP TECHED

Browsing and Syncing Development Components

- Use the development configurations perspective
- Global and local views
  - Inactive DCs view: global state of inactive DTR workspace
  - Active DCs view: global state of active DTR workspace
  - Local DCs view: Local view on available DCs
- Use context menu in the tree to
  - Sync DCs
  - Unsync DCs
  - Create projects for existing DCs to import them into the IDE

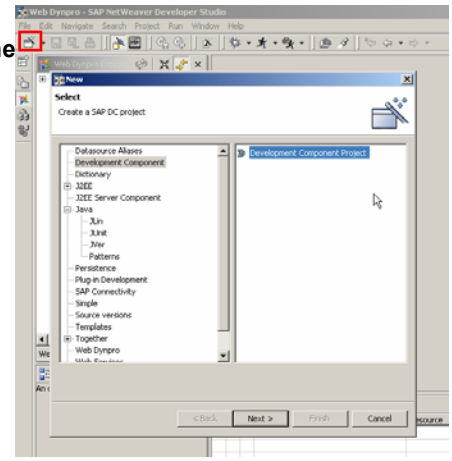
    → Do NOT use Eclipse project import for DCs !

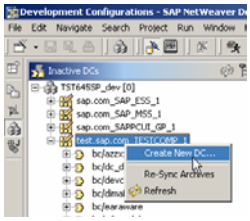


SAP TECHED

Create a new DC

- Create a new DC together with an Eclipse project with the help of the
  - Eclipse new object wizard
  - Context menu in the inactive DCs view
- First step: Assign new DC to an existing SC within an imported configuration
- Second step: Name the DC

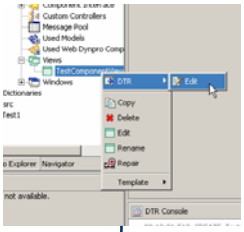


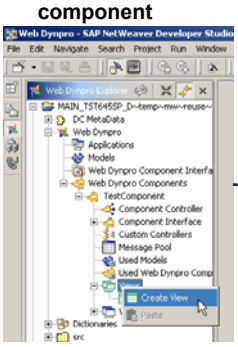


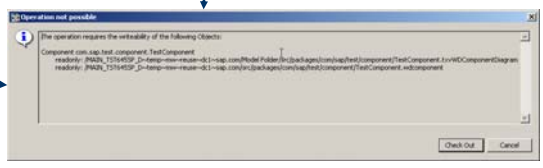
SAP TECHED

Check-out

- Check-out can be explicitly triggered by chosing Edit from the DTR submenu of the context menu of the desired object
- Check-out for Web Dynpro objects will also be implicitly triggered by trying to modify an object
  - E.g. creating a view within a read-only component will first check-out the component



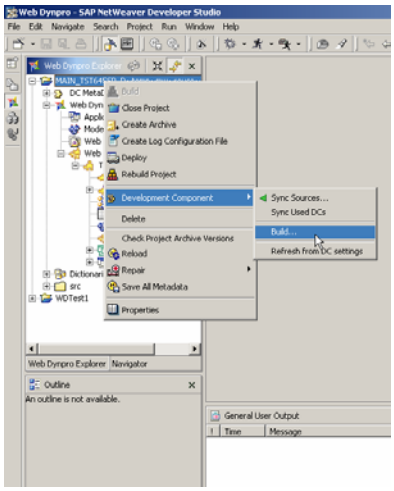




SAP TECHED

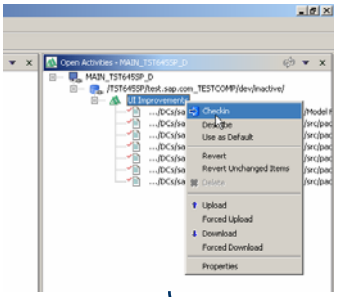
Performing DC Build

- Start DC build from the context menu of one or more DC projects
  - Considers the correct build order when called for more than one DC
- Performs
  - Full build of the DC
  - Creates a deployable archive
  - Creates public parts defined by the DC
- Build results are displayed the general user output view
  - Status message in view
  - Double click on message to get details
  - No Eclipse task entries written!
- DC build is necessary
  - When an object beloging to a public part has been changed
  - Prior to checking in to assure correctness

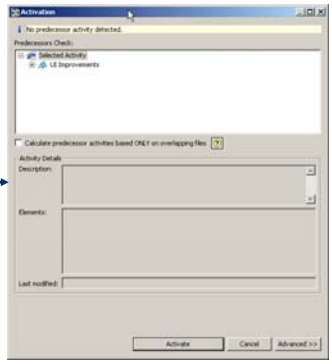


SAP TECHED

Check-in and Activating



- Check-in your changes to DTR by using the context menu entry in the open activities view in the development configuration perspective
- Status and progress messages will appear in general user output view

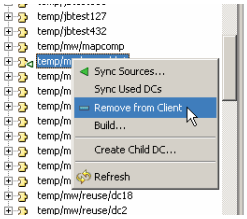


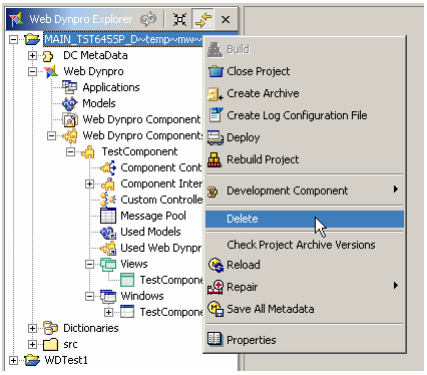
- After check-in an activation request can be triggered
- Sources and created archives will be moved from inactive to active workspace

SAP TECHED

Remove DC

- Remove DC by simply deleting the DC project including its files
  - Use delete in context menu of Web Dynpro explorer
  - Use remove from client in local or global DCs view





- Remove DC by simply deleting the DC project including its files
  - Use delete in context menu of Web Dynpro explorer
  - Use remove from client in local or global DCs view
- Local DCs can be deleted completely
- DCs stored in DTR can only be removed from the IDE

SAP TECHED

Section 2 – Component Based Application Architecture

1. Component Basics
2. Component Usage Relations
3. Demo
4. Component-Based Application Architecture
5. Hands-On Excercise
6. NetWeaver Development Infrasturcture
7. Referencing Components
8. Component Interface Definitions
9. Hands-On Excercise

SAP TECHED

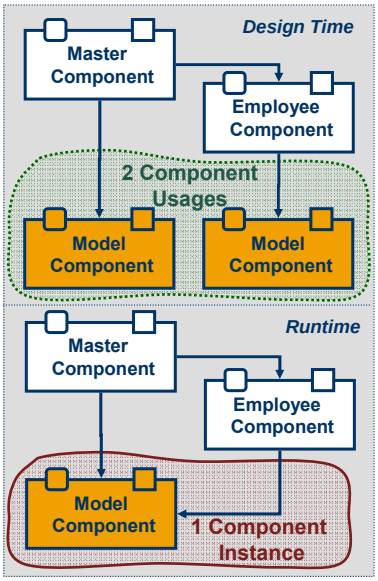
Sharing Component Instance between Component Usages

A component usage can be seen as a variable for a Web Dynpro component instance.

When using another Web Dynpro component a corresponding component usage relation must be declared.

Referencing Mode

- component usages can reference the same component instance
- instance life cycle cannot be controlled in referencing mode
- look on API *IWDComponentUsage*



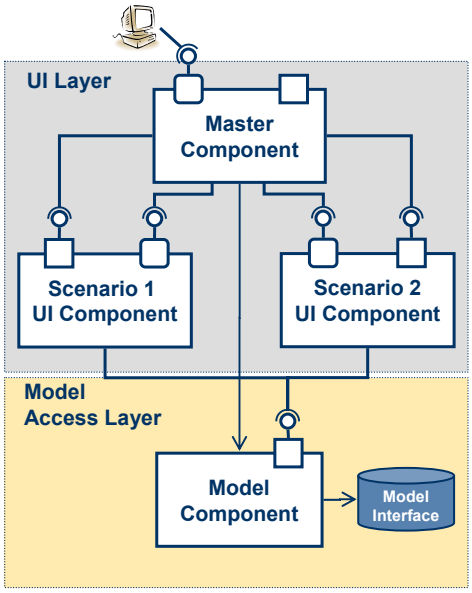
SAP TECHED

A Simple Web Dynpro Component Architecture

- How can both UI components use the same model component instance?
- But both have their own, independant component usages.

Solution

- The lifecycle of the model component is managed by the master component
- The model component usages in both UI components must define `Lifecycle = manual`.
- The reference to the model component instance is injected by the master component by calling a component interface method of the UI components.
- The UI components can then enter the model component in referencing mode.

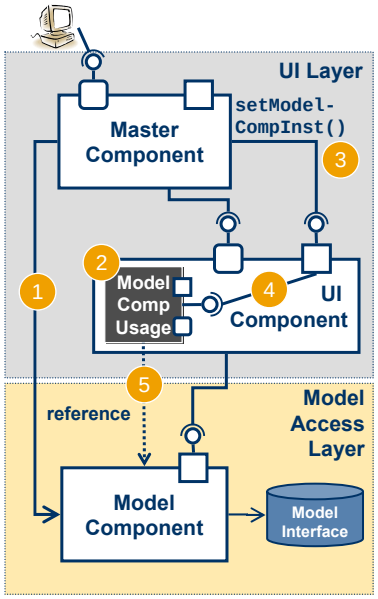


SAP TECHED

Using a Component Instance in Referencing Mode

Entering a Referencing Mode

1. The master component creates an instance of the model component.
2. The UI component has defined a *component usage relation* to the model component. It does not manage the lifecycle (`= manual`) of the used model component itself!
3. The master component passes a reference of the model component usage to the UI component (via a public component interface method).
4. The UI component uses this reference for entering the model component in *referencing mode* (`IWDComponentUsage`-API).
5. The model component usage is now related to the concrete runtime instance of the model component.

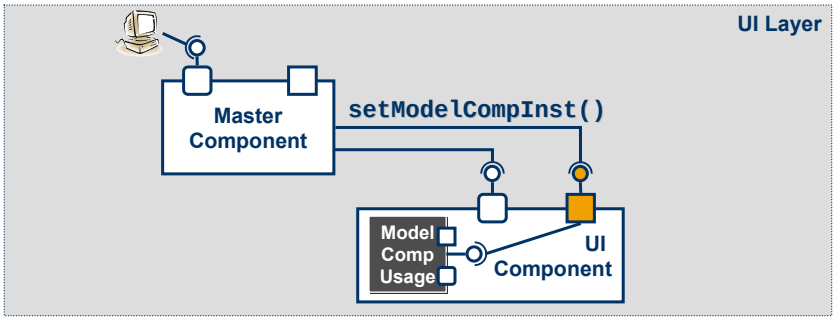


SAP TECHED

Entering another Component in Referencing Mode

UI Component Interface Controller must expose method

```
setModelComponentInstance(
  IWDComponentUsage modelCompUsg)
{
   wdThis.wdGet<ModelCompUsageName>
      .enterReferencingMode(modelCompUsg)
}
```



SAP TECHED

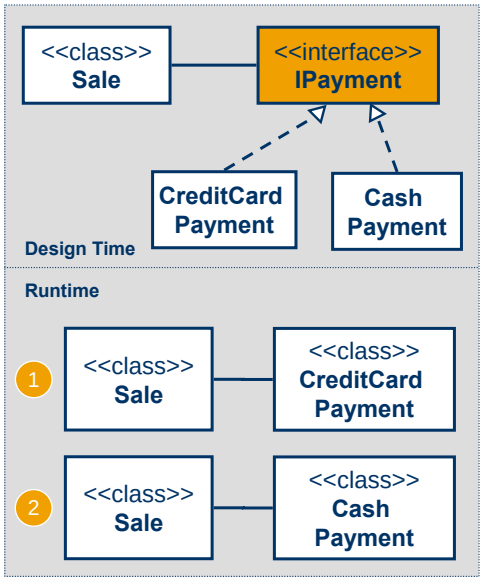
Java Interfaces and the Strategy Design Pattern

Java Interfaces

In Java an interface is a type, just as a class is a type.

Like a class an interface defines methods but never implements them.

An interface is an abstraction of a class independent from its implementation.



SAP TECHED

Java Interfaces and the Strategy Design Pattern

The Strategy Design Pattern in Java

Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.*

It embodies two main OO design principles:

- ***encapsulate the concept that varies***
- ***program to an interface, not an implementation.***

Good software = loosely coupled collection of interchangeable parts

Bad software = monolithic, tightly coupled system

Loose coupling makes your software much more extensible, maintainable, and reusable.

** Chapter 1 of the Gang of Four's (GOF) Design Patterns*

SAP TECHED

Stratey Pattern and Component Interface Definitions

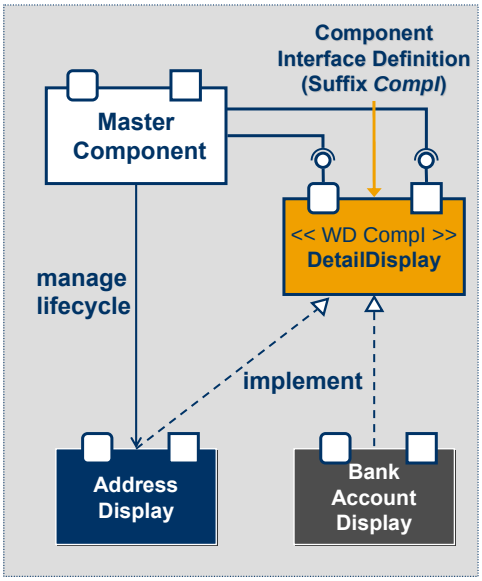
Component Interface Definitions

Allow decoupling Web Dynpro Components

- Define a component interface without an implementation
- Interface can be implemented by one or more components

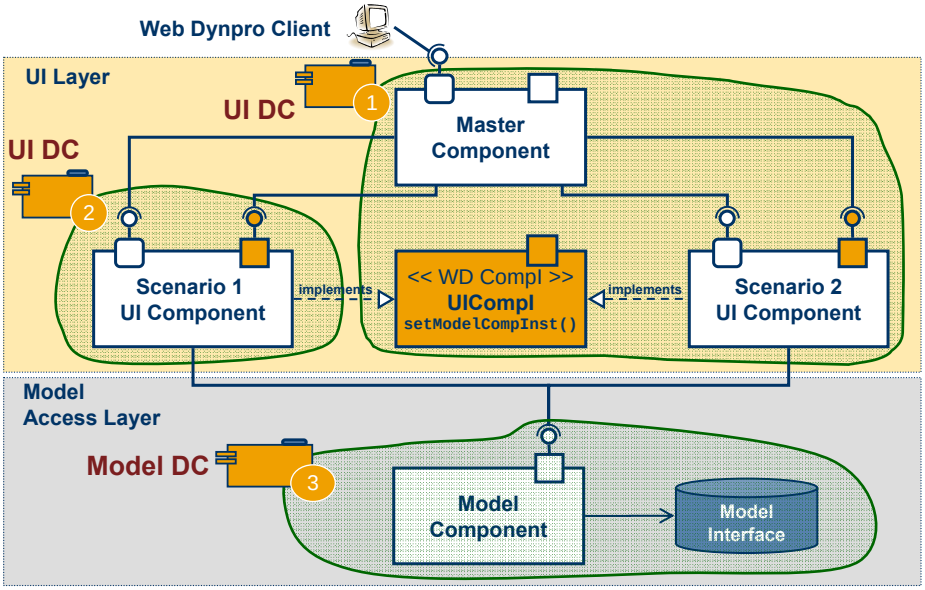
Benefits → Decoupling

- Component user can program against separately defined component interface (implementation is hidden)
- Component implementation can be provided (or selected) later on (plugin concept)



SAP TECHED

Example: Adding a Component Interface Definition



SAP TECHED

Summary

- **Web Dynpro components are the key to extensible, reusable and managable Web Dynpro software**
- **A state-of-the-art Web Dynpro application architecture can be achieved by using specialized component types**
- **The NetWeaver Development Infrastucture provides functions for developing professional, component-based Web Dynpro applications**

SAP TECHED

Further Information

Public Web:

www.sap.com

SAP Developer Network: www.sdn.sap.com ➔ Web Application Server ➔ Web Dynpro (➔ Code Samples ➔ Sample Applications and Tutorials)

SAP Customer Services Network: www.sap.com/services/

Related SAP Education Training Opportunities

http://www.sap.com/education/

Related Workshops/Lectures at SAP TechEd 2005

CD210, Transaction Handling in Web Dynpro, 1-Hour Lecture

CD213, Web Dynpro Meets JSF – A New Level of Interoperability , 1-H L.

CD258 Putting It All Together – Application Development in SAP NetWeaver, 4-H Workshop

CD255 Component-Based Java Development with SAP NetWeaver, 4-H W.

CD207 SAP NetWeaver Development Infrastructure – An Overview, 1-H L.

SAP TECHED