

# **Automating the build and deploy phases with WSAD and WebSphere together with open source tools CVS and Ant**

Magnus Larsson

Callista Enterprise AB

magnus.larsson@callista.se

<http://www.callista.se/enterprise>

Source code examples from this presentation can be downloaded from  
<http://www.callista.se/enterprise/resources/>

# Objectives and Prerequisites

## □ Objectives

- Share some experiences regarding how to automate the build and deploy phases when developing large and complex J2EE-applications

## □ Non-objectives

- Learning J2EE 1.3, WebSphere 5.0, WSAD 5.0, CVS or Ant...
- Covering automation of unit tests during build and deploy
  - See specific session on "Unit Testing"

## □ Prerequisites

- Basic understanding of the non-objectives (except Unit Testing...)

# Agenda

- Objectives and Prerequisites
- Problem definition
- Concepts and Terminology
  - A Reference Architecture (build and deploy view)
- Experiences
  - Automation of the build and deploy phases
  - Different versions of Utility Components
  - Different versions of Business Components
  - Many Utility components
- Summary

# Problem definition

- In general
    - WSAD's "Export EAR-file" only works for very small projects...
      - Use Ant and CVS!
        - Both Ant and CVS are well integrated with WSAD
  
  - When building large enterprise systems you must also avoid
    - Building monolithic systems
      - Use Components!
  
    - Big Bang projects
      - Ensure that Components can be released individually!
- ➔ Howto use CVS and Ant to build and deploy large enterprise systems based on **versioned components**?

## Problem definition, cont...

### □ When

- ...the number of components and developers increase
- ...time flies and each component has been released in a number of versions

### □ Questions

- What component-versions should I use?
  - Specifically a problem with open source!
- Which component-versions work together?
- What components should I package in my assembly and what components should I expect to be pre-installed?

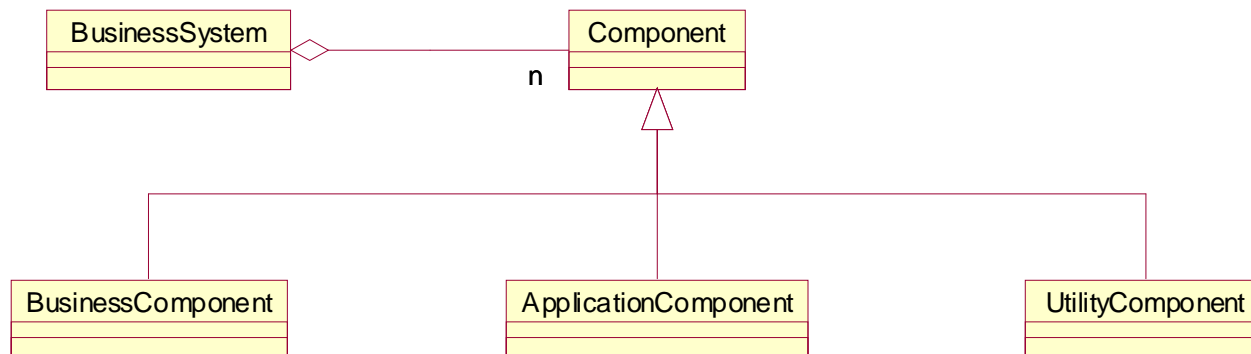
# Concepts and Terminology

- Architecture and modelling references
  - Reference Architecture based on
    - Hertzum/Sims: "Business Component Factory"
  - Callista Enterprise refinements
    - "Improve development of J2EE components Using Sun's Core J2EE Patterns"
      - <http://www.callista.se/enterprise/resources/>
  - Modelling guidelines
    - Eeles/Houston/Kozaczynski: "Building J2EE Applications with the Rational Unified Process"

# Reference Architecture: Component Model

*The Reference Architecture is described from a build and deploy view, only!*

*Component = unit of versioning and deployment*

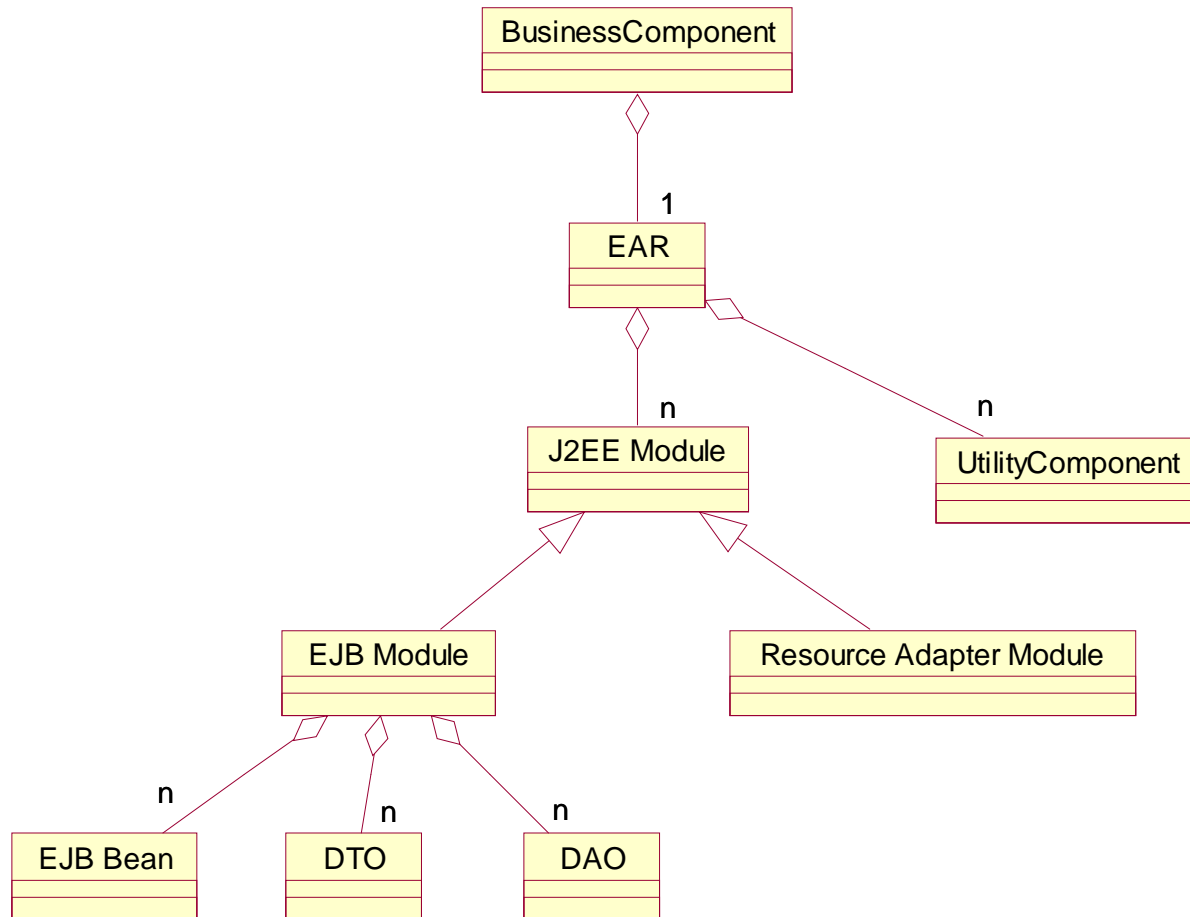


5-20 Session Beans (Facades)  
10-50 DAO or Entity Beans  
10-50 Database Tables.

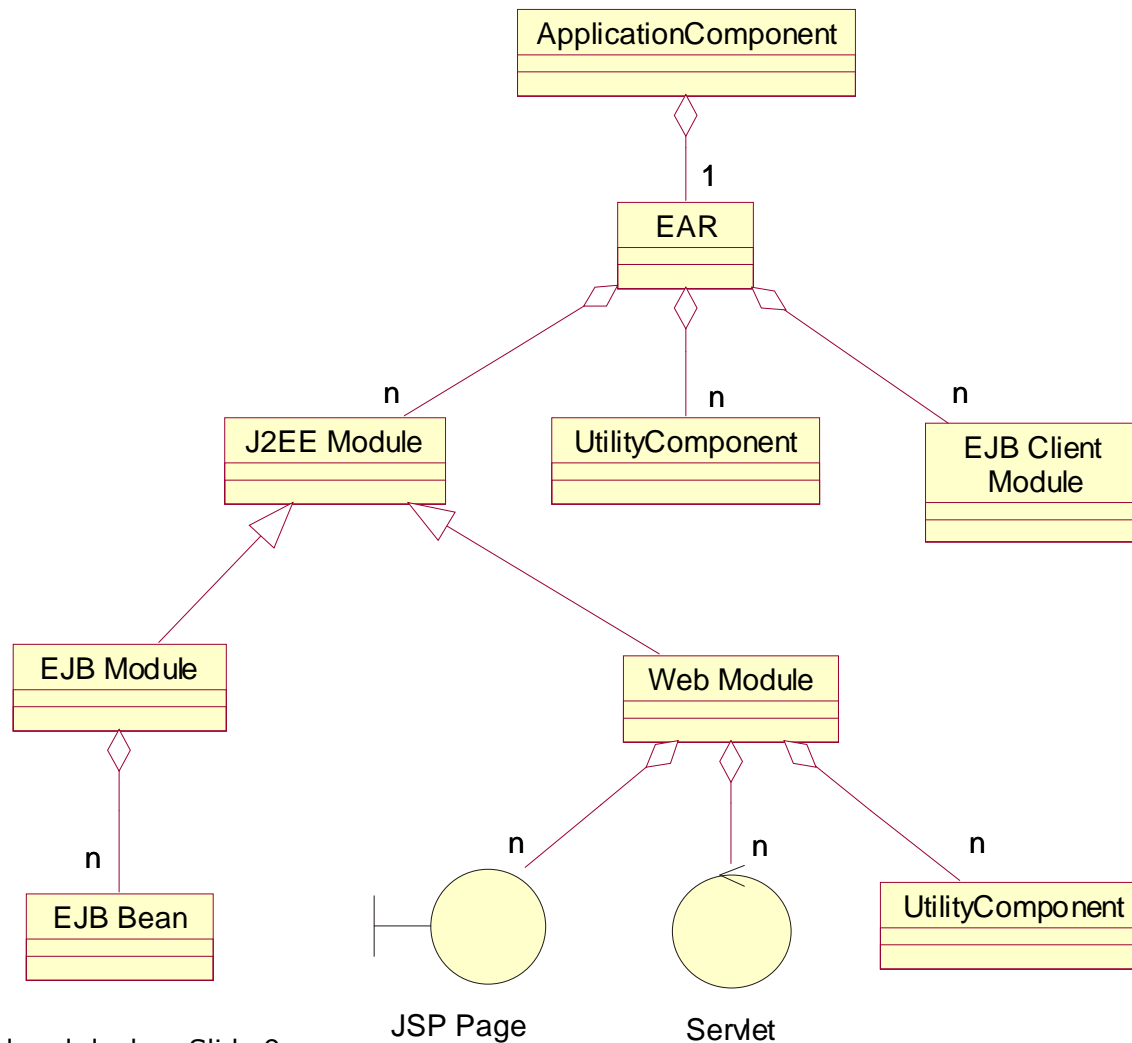
One "application"  
(several use cases)

Libraries & Frameworks  
- 3PPs  
- Open Source  
- In-house developed

# The Business Component in J2EE terms

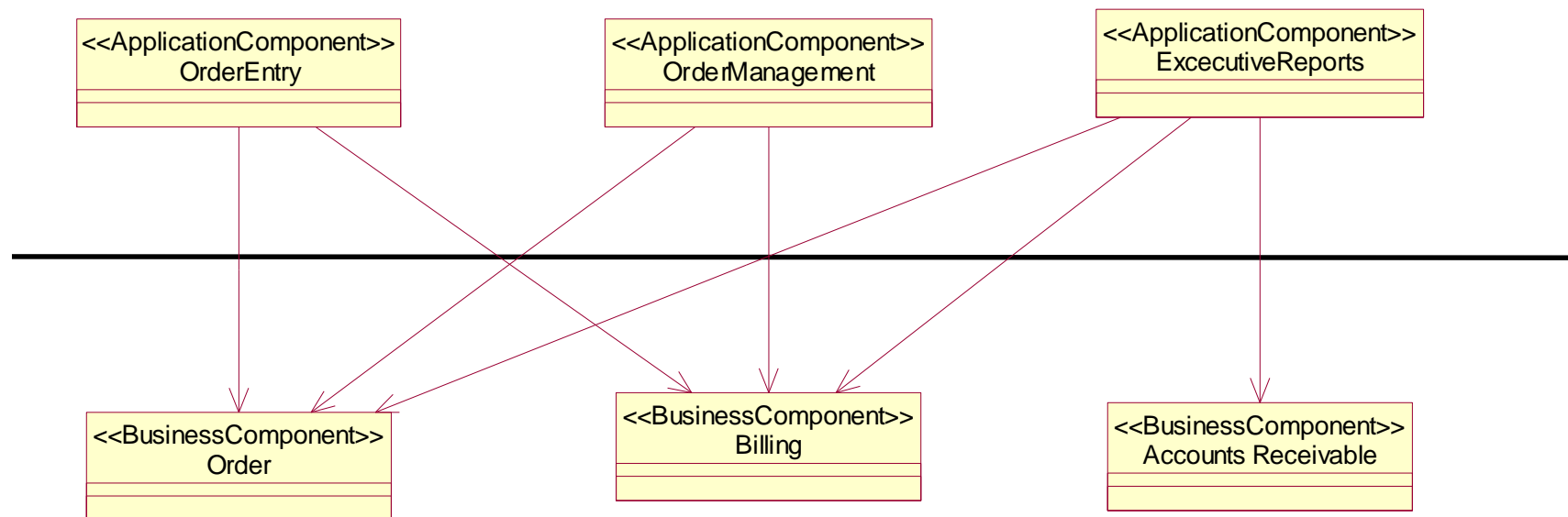


# The Application Component in J2EE terms



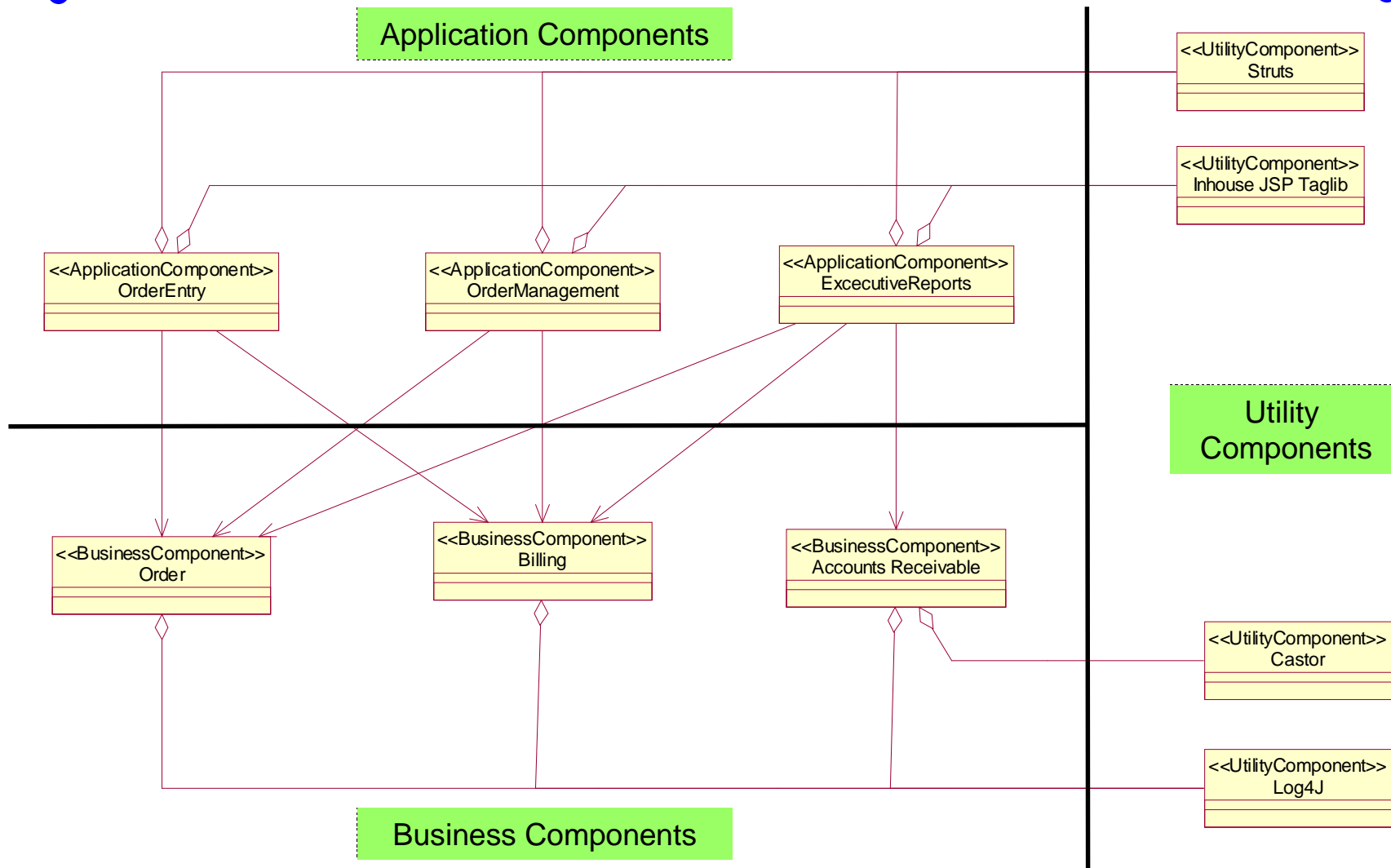
# Reference Architecture: An Example

## Application Components



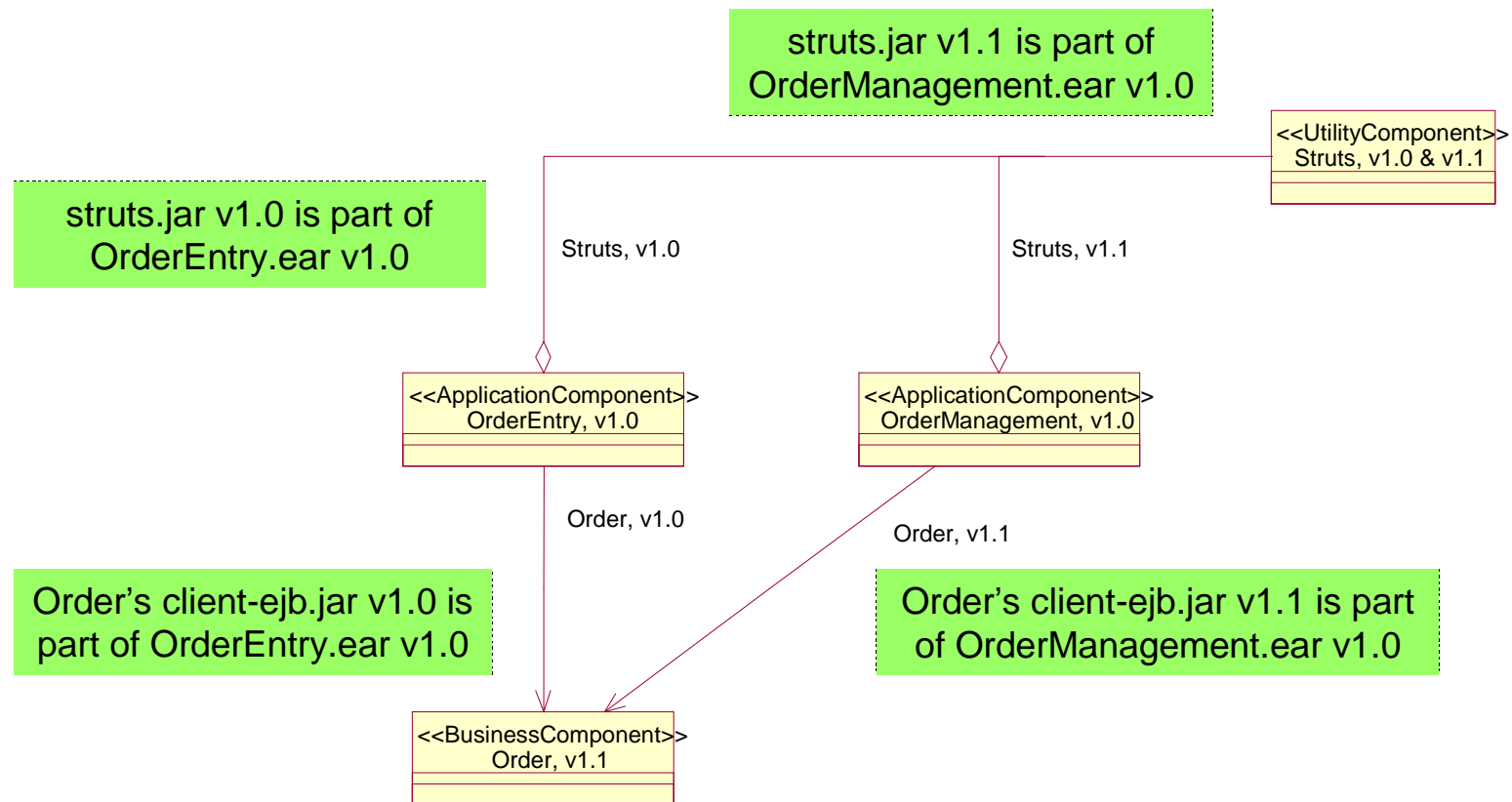
## Business Components

# Reference Architecture: An Example, cont...



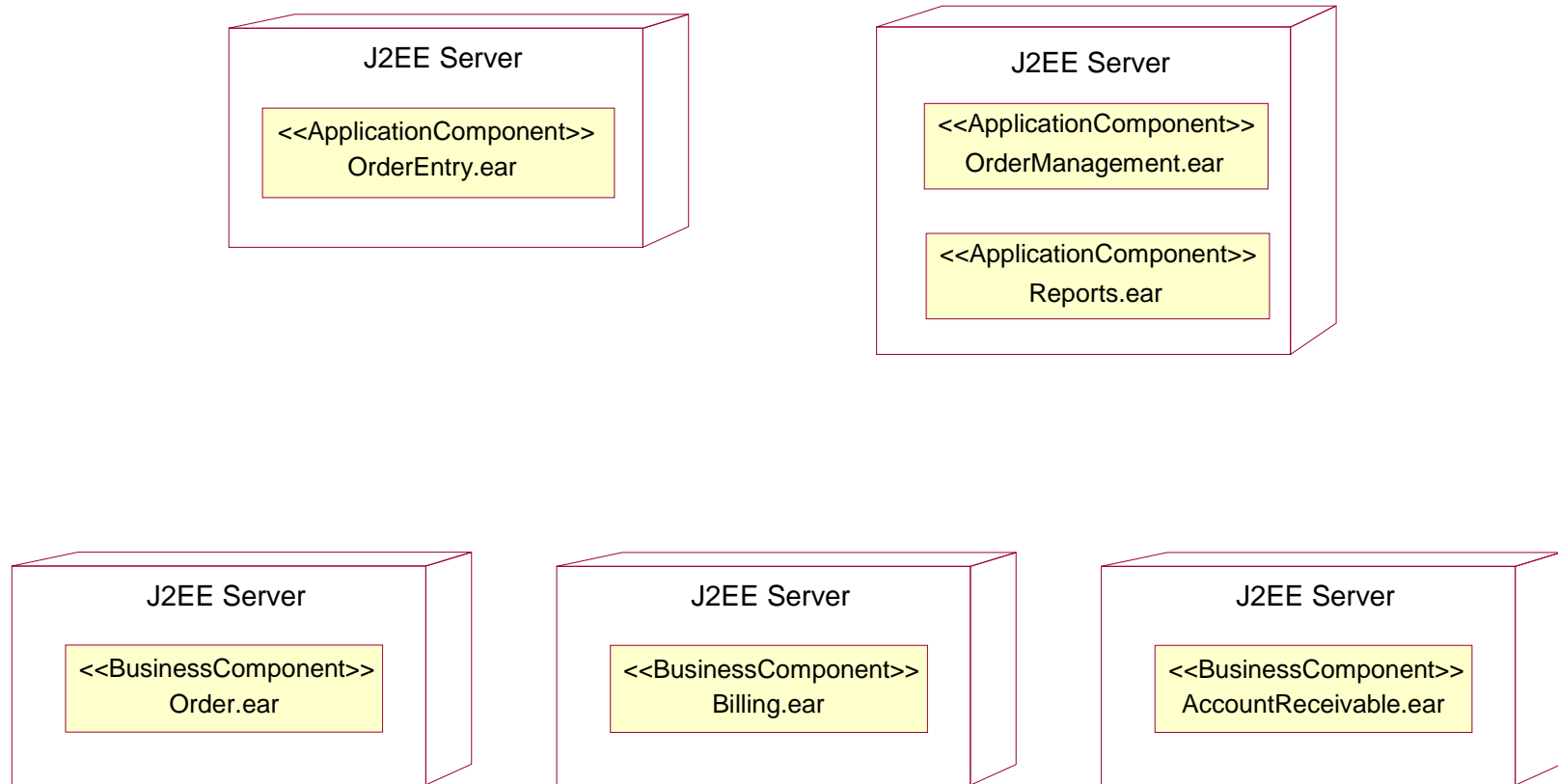
# Reference Architecture: An Example, cont...

## Component Version View



# Reference Architecture: An Example, cont...

## Deployment View



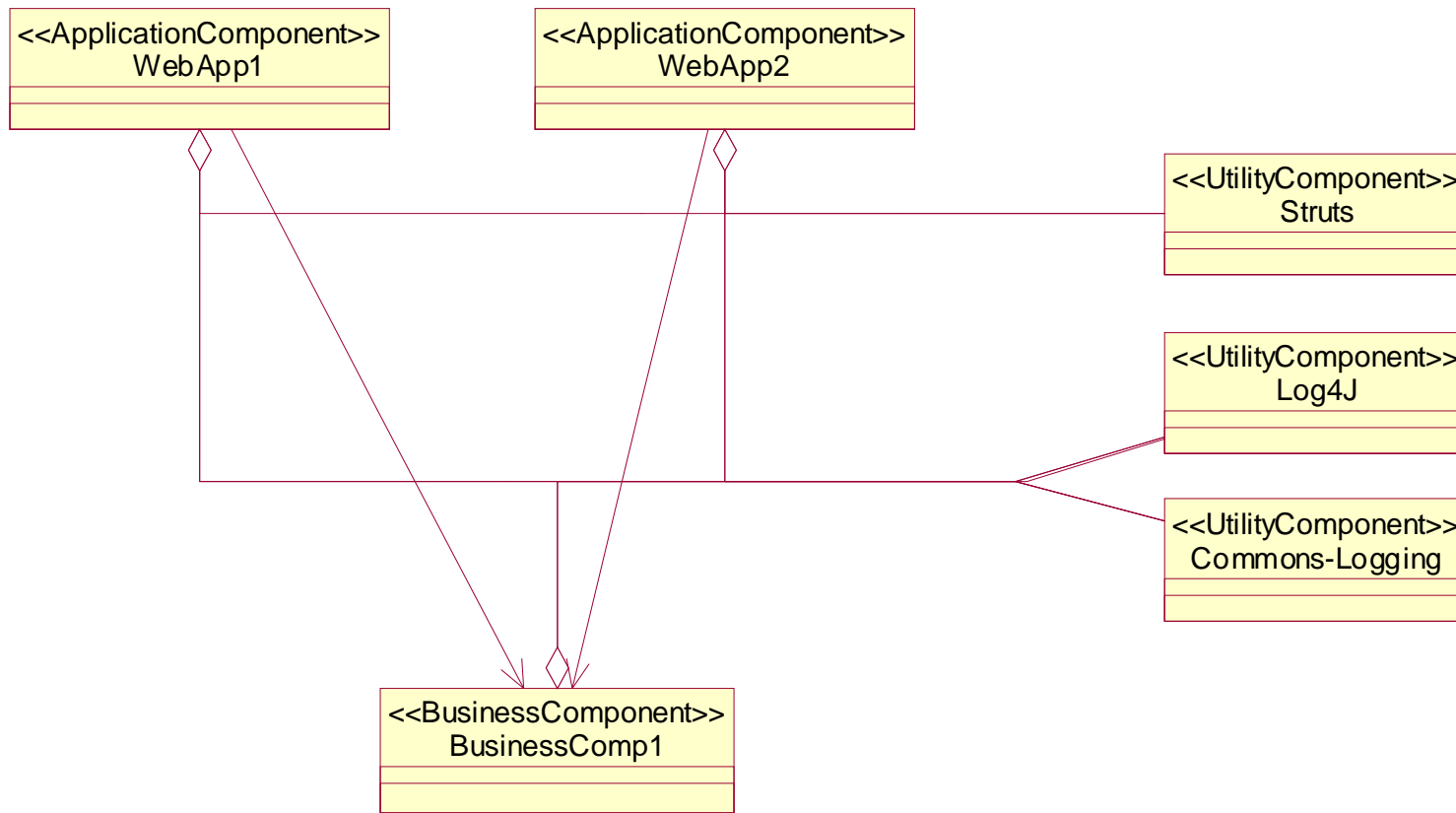
# Agenda, where are we?

- Objectives and Prerequisites
- Problem definition
- Concepts and Terminology
  - A Reference Architecture (build and deploy view)
- **Experiences**
  - Automation of the build and deploy phases
  - Different versions of Utility Components
  - Different versions of Business Components
  - Many Utility components
- Summary

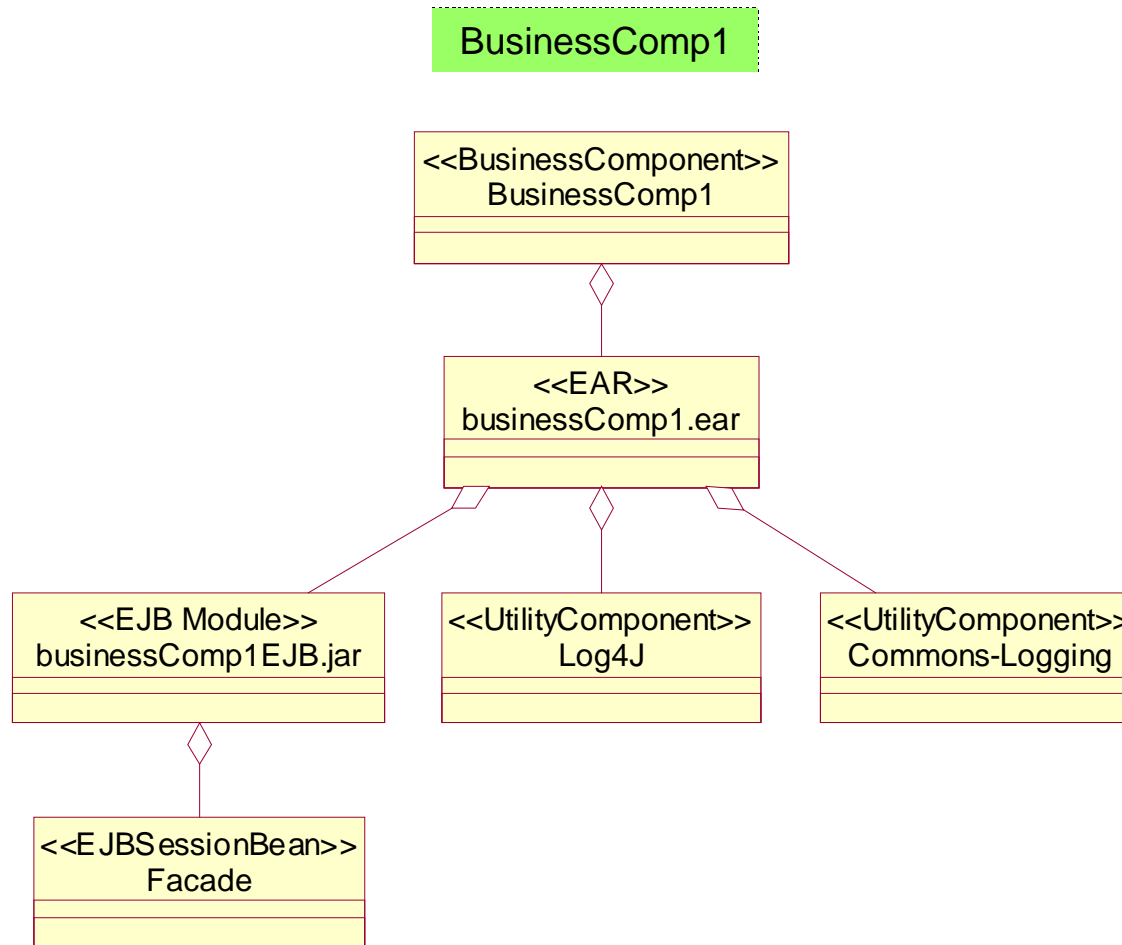
# Experiences

- Howto automate the build and deploy phases
  - Howto ensure that the right versions are used
  
- Howto handle concurrent use of different versions of Utility Components
  - Can two Application Components use different versions of Struts at the same time?
  
- Howto handle concurrent use of different versions of Business Components
  - Can two Application Components use different versions of the `ejb-client.jar` – file for a Business Component at the same time?
  
  - Howto handle extensions of the interfaces of a Business Component
  
- Howto handle the problem with many Utility components
  
- ➔ Let's define the smallest possible example to cover these aspects...

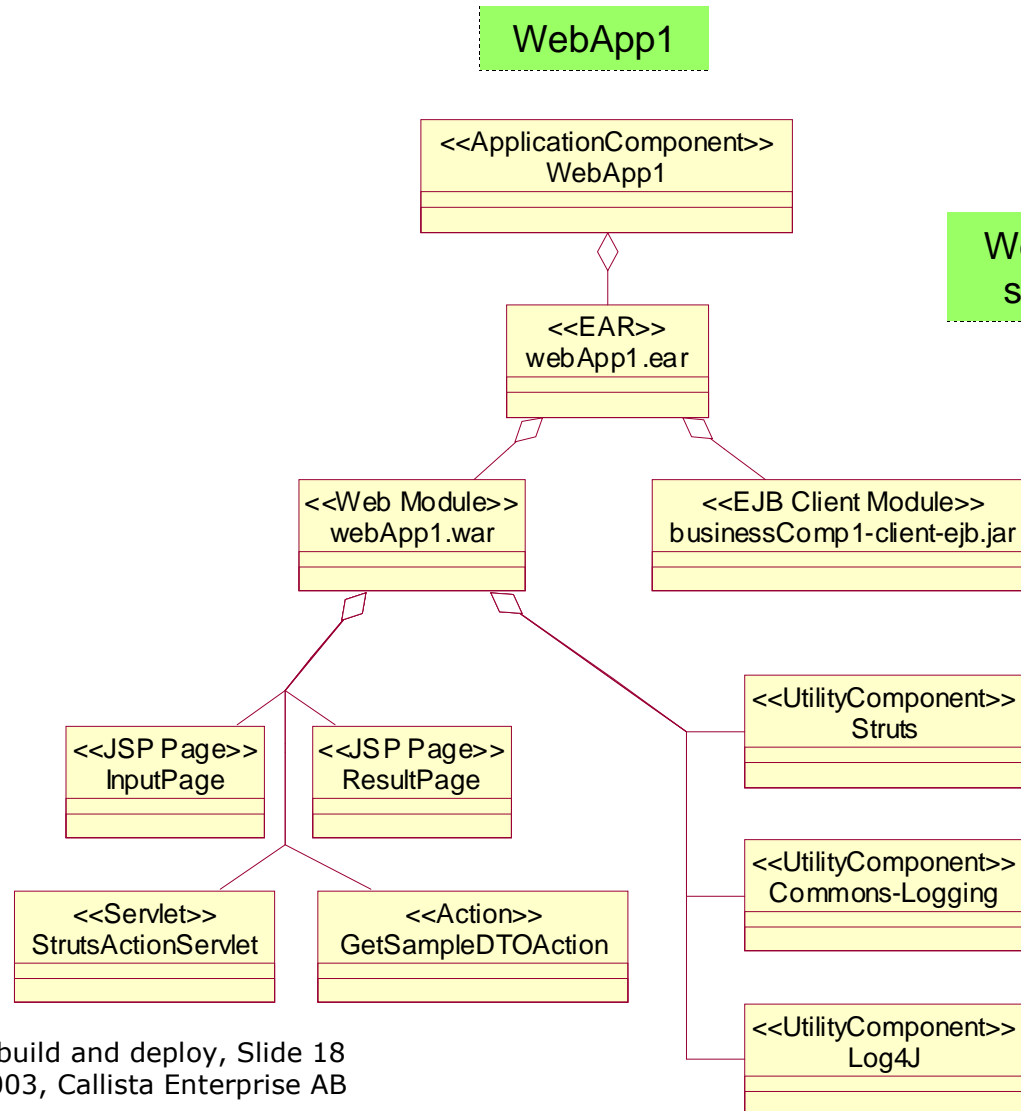
# Example



# Example - J2EE View

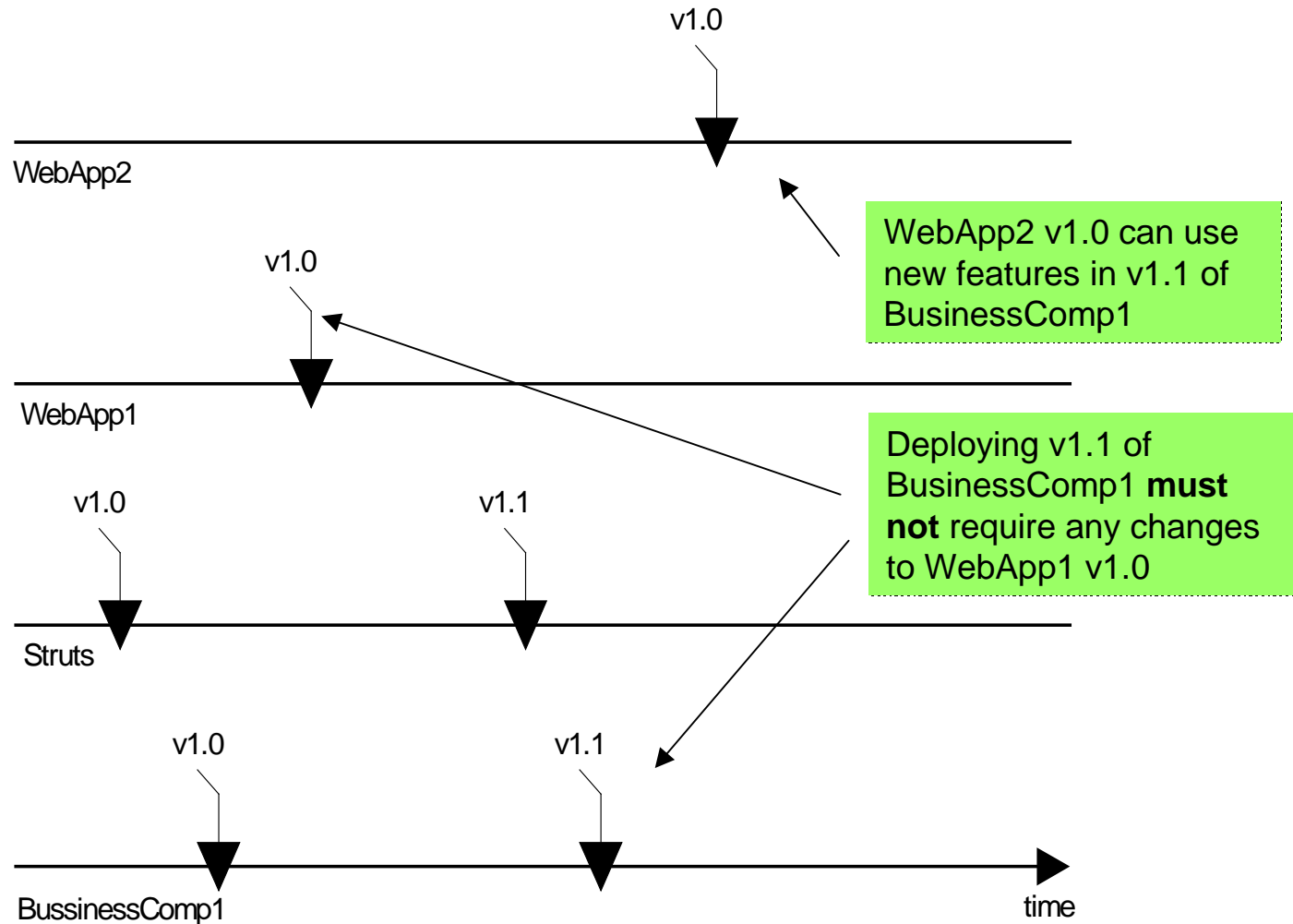


# Example - J2EE View

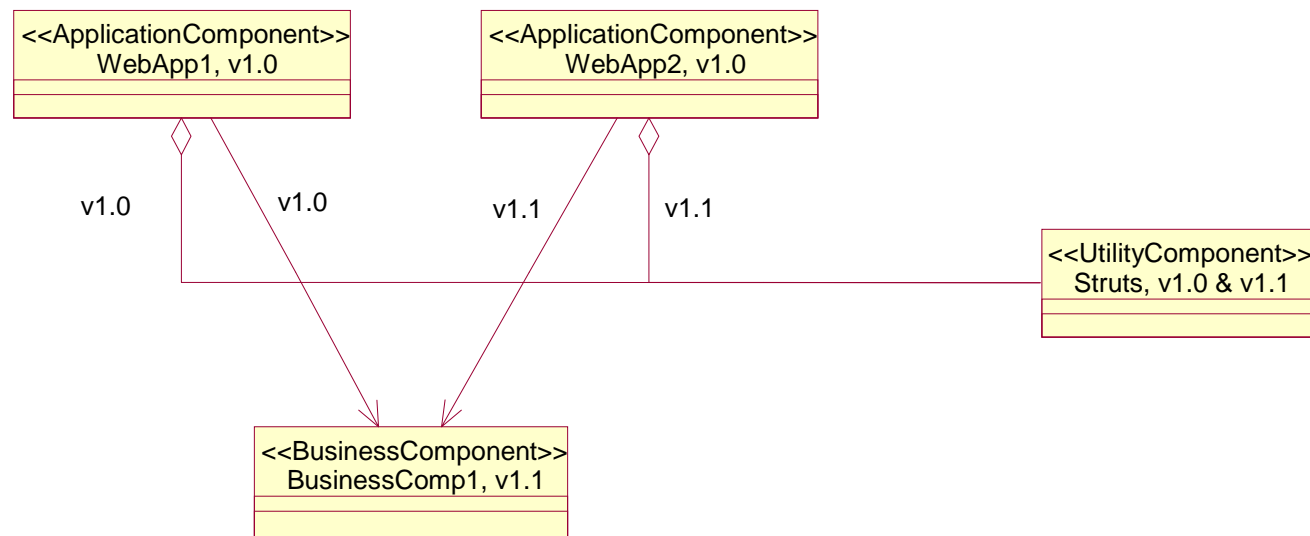


WebApp2 is has a similar structure

# Example - Timeline

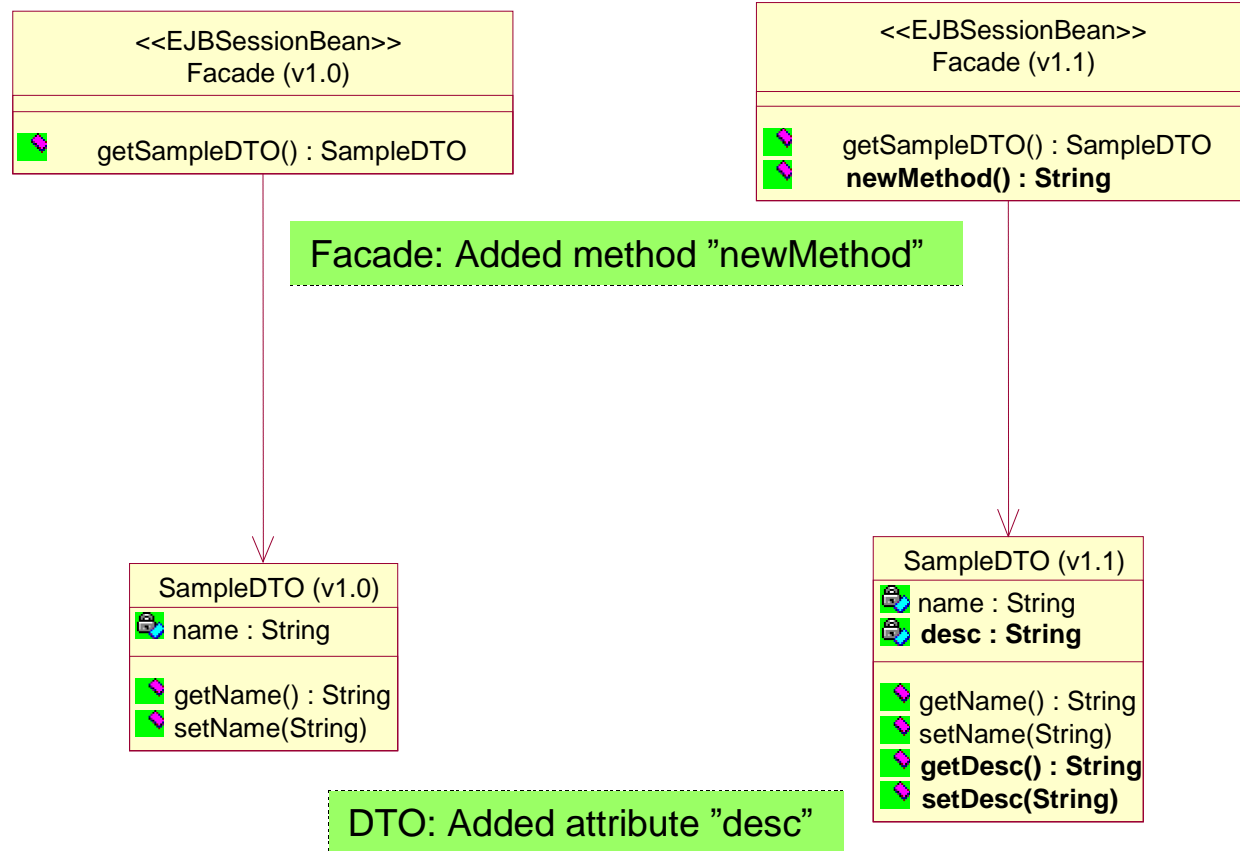


# Example – Version dependencies

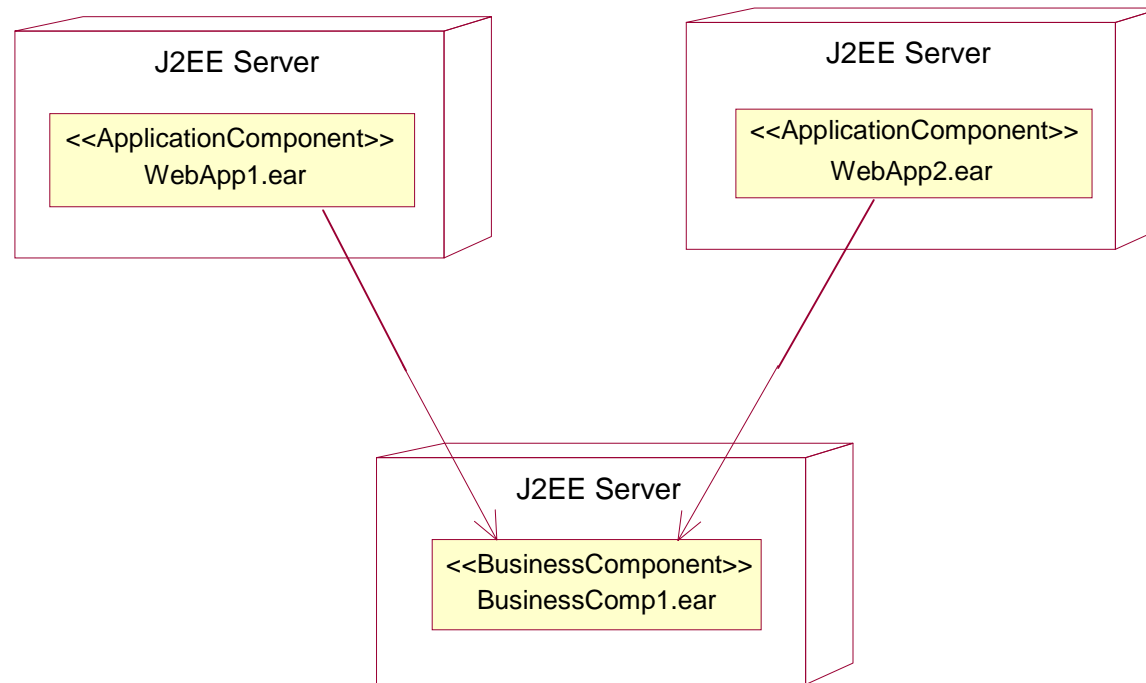


- Different versions of Struts
  - Bundle different versions at build-time
- Different versions of BusinessComp1
  - Special attention is required, see next slide!

# Example – BusinessComp1 v1.0 vs. v1.1



# Example – Deployment View

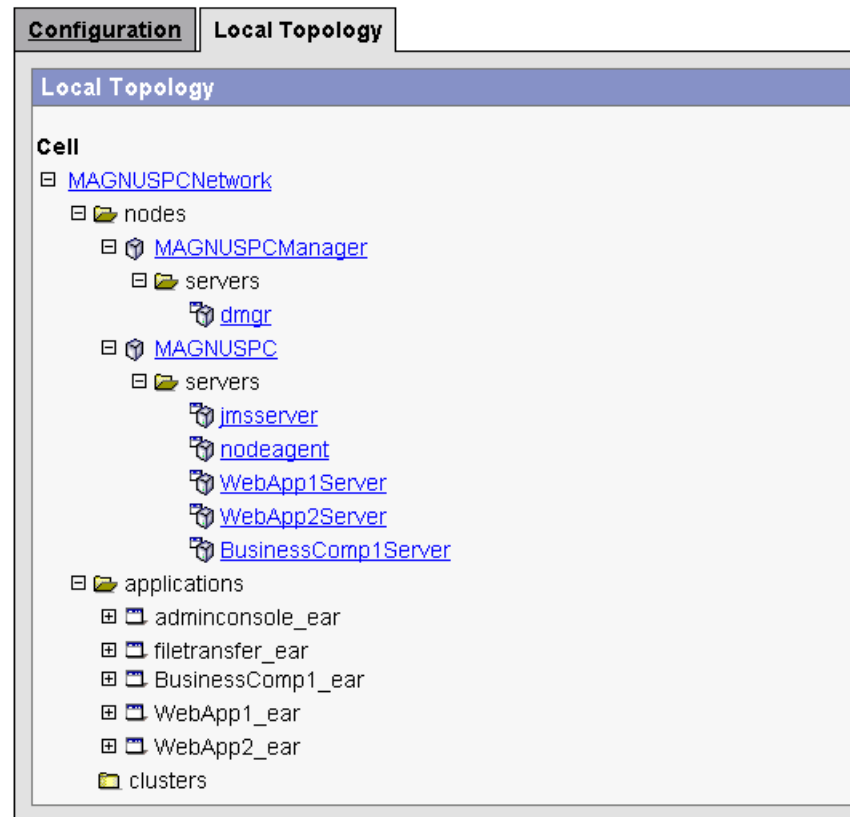


# Example – Deployment View

From the Admin Console in WebSphere 5.0 Network Deployment

## MAGNUSPCNetwork

A logical grouping of WebSphere nodes for the purposes of common administration.



# Agenda, where are we?

- Objectives and Prerequisites
- Problem definition
- Concepts and Terminology
  - A Reference Architecture (build and deploy view)
- Experiences
  - **Automation of the build and deploy phases**
  - Different versions of Utility Components
  - Different versions of Business Components
  - Many Utility components
- Summary

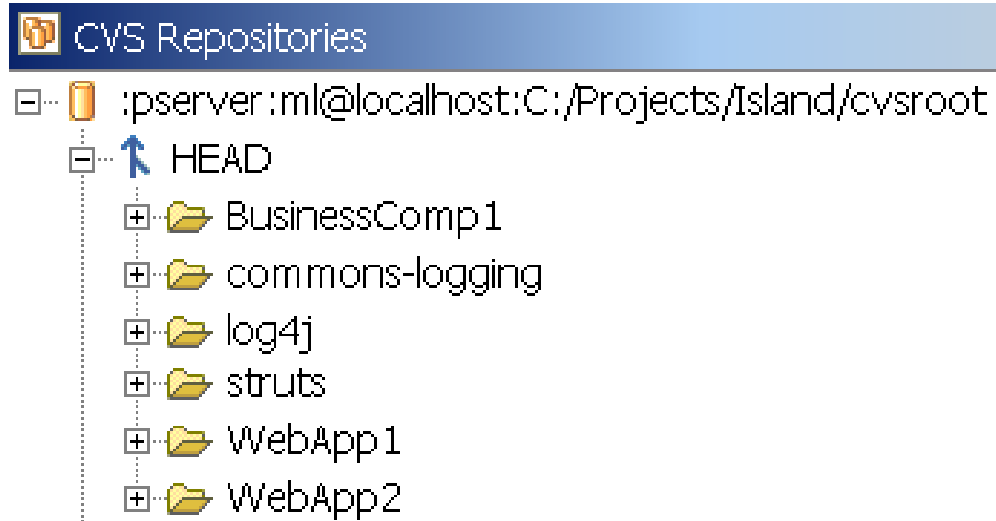
# Automation of the build and deploy phases

## □ Examples

- CVS – structure
- Build - script
- Script support in WebSphere 5.0
- J2EE Infrastructure setup and teardown scripts
  - Efficient setup of test environments
- Deploy - script

# Automation of the build and deploy phases

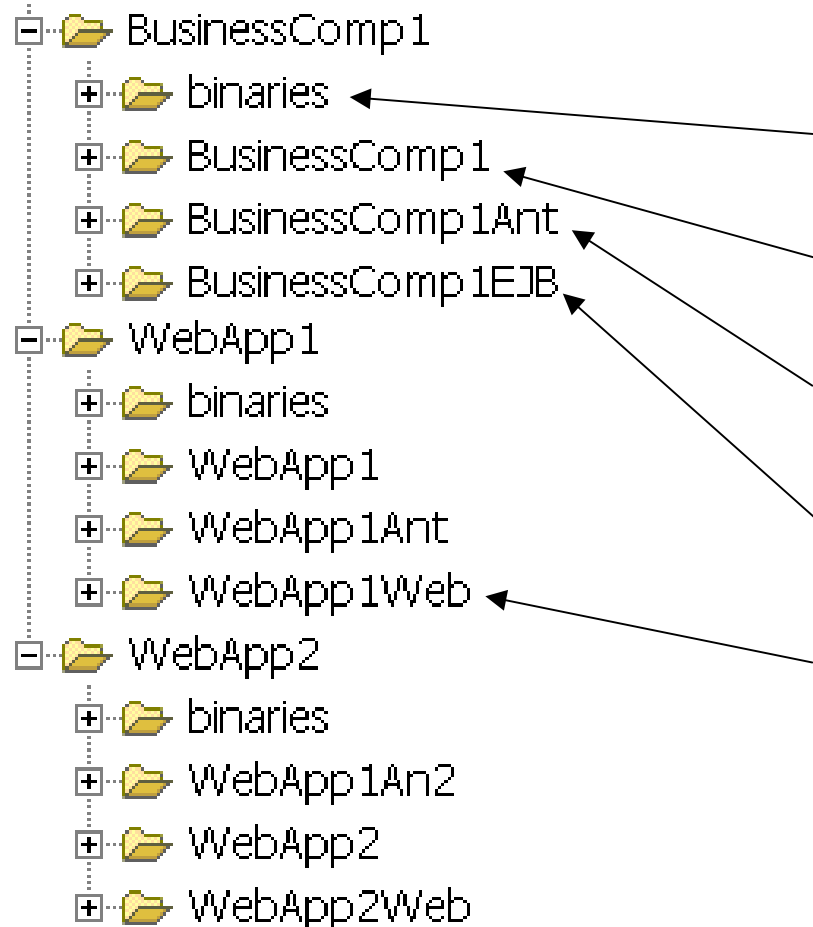
- CVS – structure, Component level



Each Component has its own top-module under the CVS-root

# Automation of the build and deploy phases

## □ CVS – structure, Project level



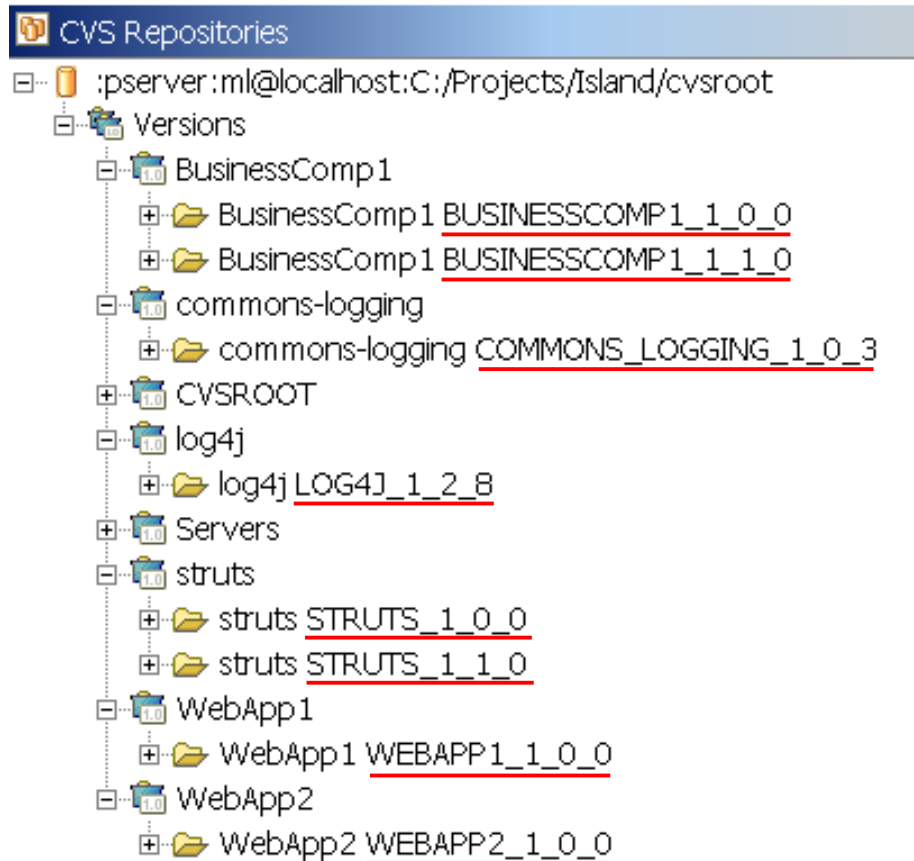
Each Component has a number of projects:

- **Binaries**  
Contains the deliverables for the component
- **Ear**  
Contains the Ear - project
- **Ant**  
Contains the Ant-scripts
- **Ejb**  
Contains EJB module - projects
- **Web**  
Contains Web module - projects

WSAD "Check Out" is done on the Project level and never on the Component level

# Automation of the build and deploy phases

## □ CVS – structure, Component Versions



Versions are only tagged on the Component level and not on single projects within a Component.

# Automation of the build and deploy phases

## □ Ant-script fragments for building WebApp1

- Read version-information

```
<property file = "versions.properties"/>
```

- Checkout WebApp1 and the binaries of the components it uses

```
<cvspackage = "xxx" tag = "${xxx.version}".../>
```

- Copy binary files from the used components

```
<copy todir = "${webLib.dir}".../>
```

- Compile the source code in WebApp1

```
<javac srcdir = "${webSource.dir}".../>
```

- Create the war-file and the ear-file

```
<jar jarfile = "${earProject.dir}/WebApp1Web.war".../>
```

```
<jar jarfile = "${binaries.dir}/${appComp.name}.ear".../>
```

- Checkin the ear-file into the WebApp1/binaries – project in CVS

```
<cvscommand="commit ${appComp.name}.ear".../>
```

# Automation of the build and deploy phases

## □ `versions.properties` for WebApp1

# Dependencies to Utility Components

`struts.version` = `STRUTS_1_0_0`

`commonsLogging.version` = `COMMONS_LOGGING_1_0_3`

`log4j.version` = `LOG4J_1_2_8`

# Dependencies to Business Components

`businessComp1.version` = `BUSINESSCOMP1_1_0_0`

## □ Differences in `versions.properties` for WebApp2

`struts.version` = `STRUTS_1_1_0`

`businessComp1.version` = `BUSINESSCOMP1_1_1_0`

## Automation of the build and deploy phases

- WebSphere v5.0 adds new scripting support
  - Internal object model exposed using JMX
    - Inline with the "soon to be released..." J2EE v1.4
  - All configurable entities in a Cell are programmatically accessible as JMX MBeans
  - A JMX client only need to communicate with the Deployment Manager
  - Bean Scripting Framework (BSF) based on JMX
  - wsadmin is a script-tool based on BSF
  - wsadmin supports today only the script language JACL
  - JACL is a Java version of TCL (Tool Command Language)

# Automation of the build and deploy phases

□ Sample JACL for setup and teardown of a J2EE test environment

□ Create a J2EE server

```
$AdminConfig create Server $nodeName $attributes  
$AdminConfig save  
$AdminControl startServer $serverName $nodeName
```

□ Remove a J2EE server

```
$AdminControl stopServer $serverName $nodeName  
$AdminConfig remove $server  
$AdminConfig save
```

# Automation of the build and deploy phases

- Sample Ant- and JACL-script for deployment
  - Checkout binaries of the components to be deployed

```
<property file = "versions.properties"/>  
<cvstag package = "WebApp1/binaries"  
tag = "${WebApp1.version}".../>
```

- Deploy an J2EE application

```
$AdminApp install WebApp1.ear  
{-node MAGNUSPC -server WebApp1Server}  
$AdminConfig save  
$AdminControl invoke $appManager startApplication WebApp1
```

- Undeploy an J2EE application

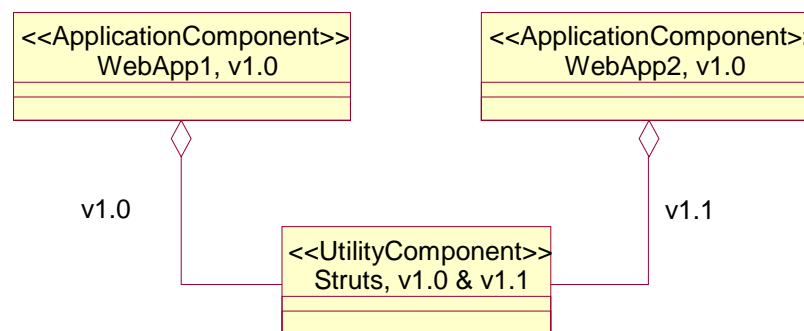
```
$AdminApp uninstall $app  
$AdminConfig save
```

# Agenda, where are we?

- Objectives and Prerequisites
- Problem definition
- Concepts and Terminology
  - A Reference Architecture (build and deploy view)
- Experiences
  - Automation of the build and deploy phases
  - **Different versions of Utility Components**
  - Different versions of Business Components
  - Many Utility components
- Summary

# Different versions of Utility Components

- No problem!
- The Utility Components are packaged into the EAR-file of each Application and Business Component
- WebApp1 v1.0 can use Struts v1.0 and WebApp2 v1.0 can use Struts v1.1 at the same time



# Agenda, where are we?

- Objectives and Prerequisites
- Problem definition
- Concepts and Terminology
  - A Reference Architecture (build and deploy view)
- Experiences
  - Automation of the build and deploy phases
  - Different versions of Utility Components
  - **Different versions of Business Components**
  - Many Utility components
- Summary

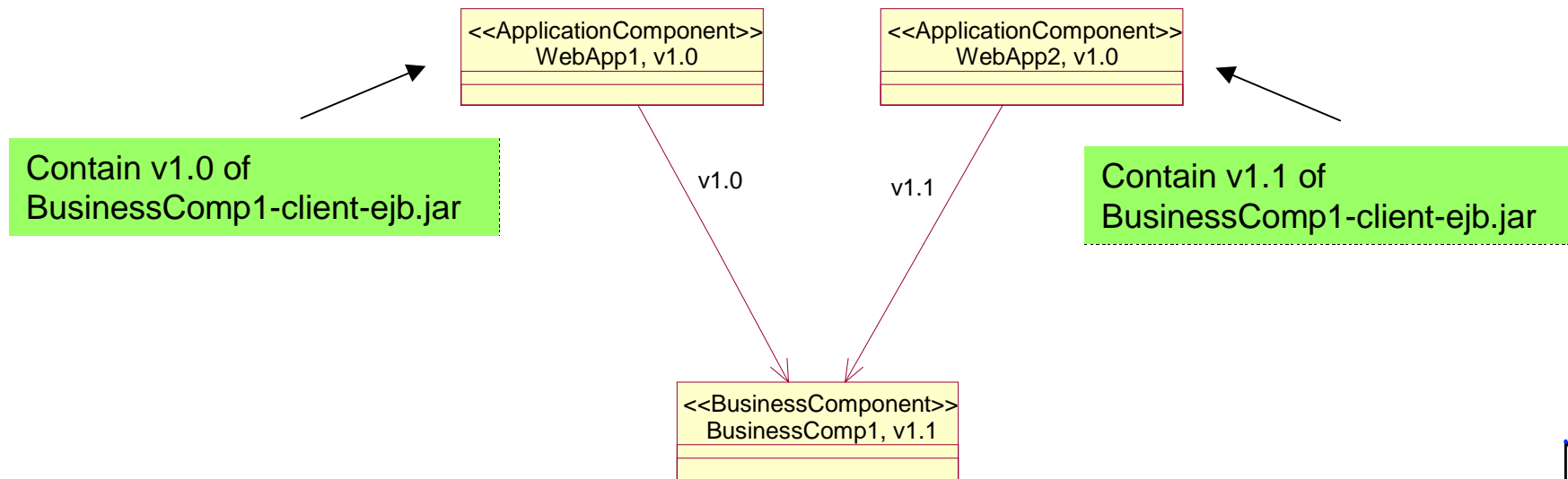
# Different versions of Business Components

- Problems with **Versioned** Business Components
  - Only one version of a Business Component can be installed at the same time
  - When a new version of a Business Component is deployed will existing Application Components contain “old” versions of the `client-ejb.jar` - file.
  - Application Owners are not willing to deploy a new version of the Applications every time a new version of a Business Component is deployed
  
- ➔ We must be able to live with old `client-ejb.jar` - files.

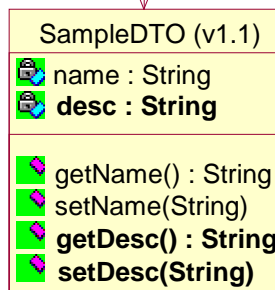
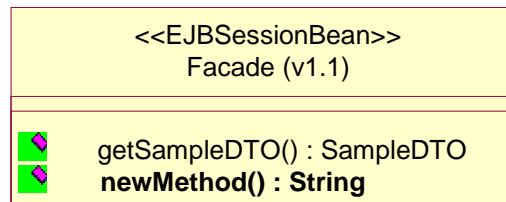
# Different versions of Business Components

## □ Problems with **Versioned** Business Components, cont...

- Don't break the interfaces, only extend!
  - It must be possible to
    - Add methods to Facades - EJB Session beans
    - Add [optional] attributes to DTO's



# Different versions of Business Components



- Adding methods to Facade - EJB Session ok
  - Doesn't cause any problems for existing Applications
- Adding attributes to DTO's will make existing applications to fail
  - Due to "old" client-ejb.jar files in the Applications will ClassCastExceptions (or similar...) be thrown

# Different versions of Business Components

## □ Adding attributes to DTO's, alternative #1

- Use standard Java mechanism "Serial UID" to make it work

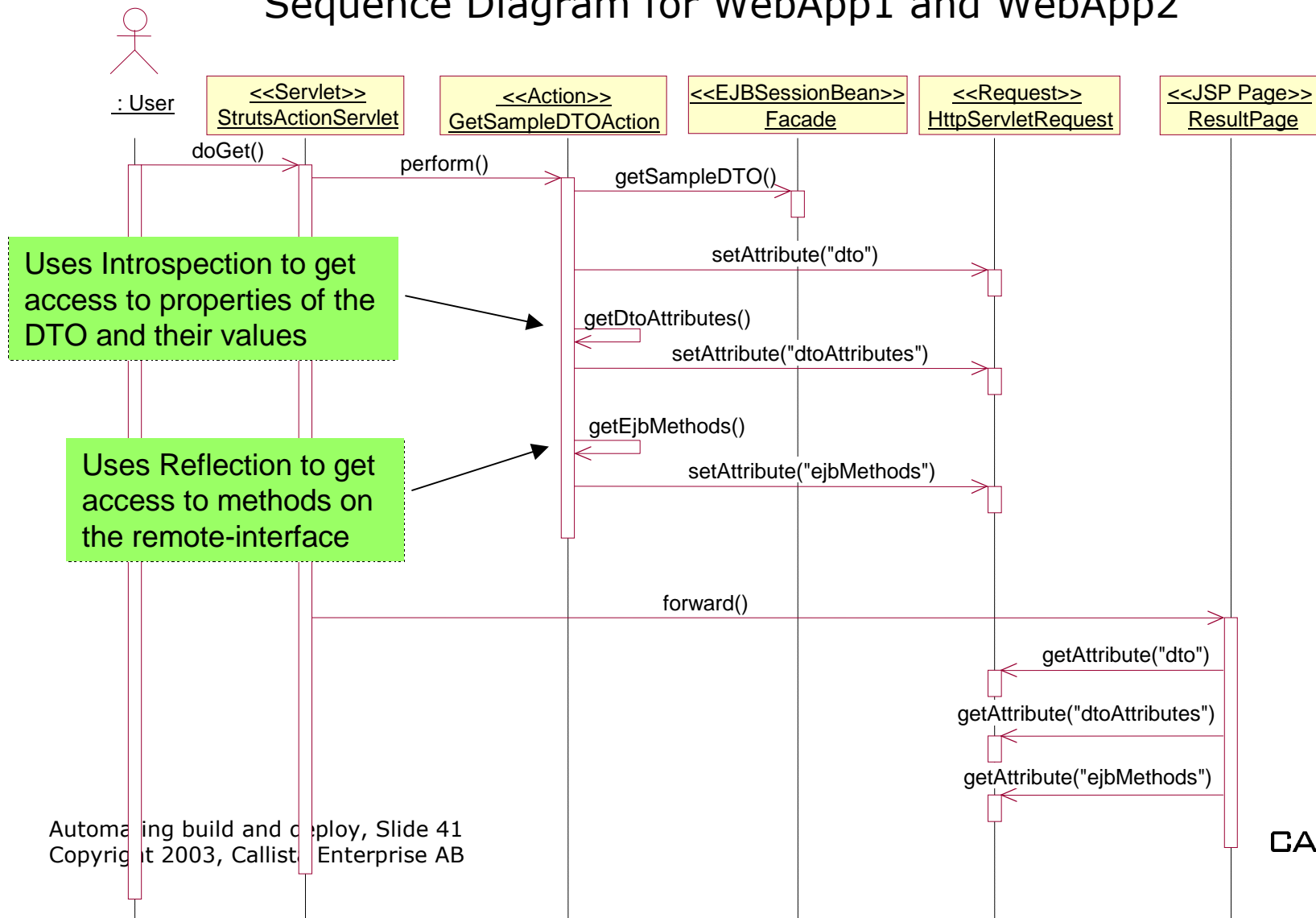
```
public class SampleDTO implements Serializable {  
    static final long serialVersionUID = -976579209635629459L;  
}
```

## □ Does it work?

- Is the "BusinessComponent v1.0" - view really retained for WebApp1 v1.0 when "BusinessComponent v1.1" is deployed?
- Let us test with some introspection and reflection...
  - WebApp1 v1.0 use the old v1.0 ejb-client.jar file
  - WebApp2 v1.0 use the new v1.1 ejb-client.jar file

# Different versions of Business Components

## Sequence Diagram for WebApp1 and WebApp2



# Different versions of Business Components

*Output from WebApp2  
using the new v1.1  
ejb-client.jar - file*

The new attribute and the  
new method in v1.1 is  
there as expected!

```
WebApp2 - ResultPageDTO

Name = a name
Desc = a description

Content of a dto of type = se.callista.businesscomp1.SampleDTO
- name = a name
- class = class se.callista.businesscomp1.SampleDTO
- desc = a description

Methods in a ejb-bean of type = se.callista.businesscomp1._Facade_Stub
- Method name = newMethod
- Method name = getSampleDTO
- Method name = isIdentical
- Method name = getHandle
- Method name = remove
- Method name = getPrimaryKey
- Method name = getEJBHome
- Method name = _ids

Timestamp = Wed Aug 20 22:42:56 CEST 2003
```

# Different versions of Business Components

*Output from WebApp1  
using the old v1.0  
ejb-client.jar - file*

The new attribute and the  
new method in v1.1 is  
**NOT** there as expected!

```
WebApp1 - ResultPage  
Name = a name  
  
Content of a dto of type = se.callista.businesscomp1.SampleDTO  
- name = a name  
- class = class se.callista.businesscomp1.SampleDTO  
  
Methods in a ejb-bean of type = se.callista.businesscomp1._Facade_Stub  
- Method name = getSampleDTO  
- Method name = isIdentical  
- Method name = getHandle  
- Method name = remove  
- Method name = getPrimaryKey  
- Method name = getEJBHome  
- Method name = _ids  
  
Timestamp = Wed Aug 20 22:42:56 CEST 2003
```

# Different versions of Business Components

- Adding attributes to DTO's, alternative #2
  - If you don't want to play around with "Serial UID's"
    - Use a HashMap as the single member inside the DTO
    - Provide getter and setter methods as before as the interface

```
public class SampleDTO implements Serializable {
    private HashMap map = new HashMap();

    public String getName() {return (String)map.get("name");}

    public void setName(String name) {map.put("name", name);}

    public String getDesc() {return (String)map.get("desc");}

    public void setDesc(String desc) {map.put("desc", desc);}
}
```

# Agenda, where are we?

- Objectives and Prerequisites
- Problem definition
- Concepts and Terminology
  - A Reference Architecture (build and deploy view)
- Experiences
  - Automation of the build and deploy phases
  - Different versions of Utility Components
  - Different versions of Business Components
  - **Many Utility components**
- Summary

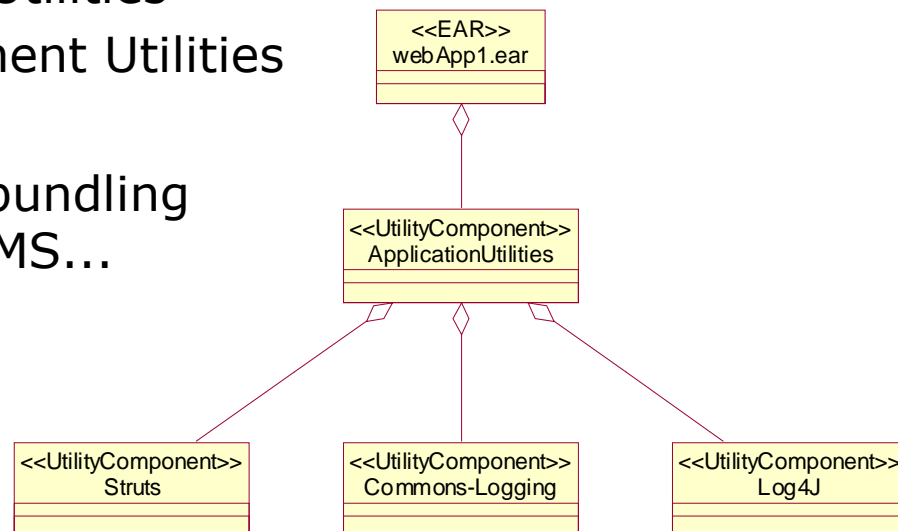
# Many Utility components

## □ Problem

- When a Component depends on >10 Utility Components
  - Keeping track of all component versions
  - What version of the utility components work together?

## □ Solution

- Bundle into larger versioned Utility Components, e.g.
  - Web Application Utilities
  - Business Component Utilities
- Compare with J2EE bundling Servlets, JSP, EJB, JMS...



# Many Utility components

- `versions.properties` for ApplicationUtilities v1.0.0

# Dependencies to Utility Components

`struts.version` = `STRUTS_1_0_0`

`commonsLogging.version` = `COMMONS_LOGGING_1_0_3`

`log4j.version` = `LOG4J_1_2_8`

- Simplified `versions.properties` for WebApp1

# Dependencies to Utility Components

`applicationUtilities.version` = `APPLICATION_UTILITIES_1_0_0`

# Dependencies to Business Components

`businessComp1.version` = `BUSINESSCOMP1_1_0_0`

# Agenda, where are we?

- Objectives and Prerequisites
- Problem definition
- Concepts and Terminology
  - A Reference Architecture (build and deploy view)
- Experiences
  - Automation of the build and deploy phases
  - Different versions of Utility Components
  - Different versions of Business Components
  - Many Utility components
- **Summary**

# Did we meet the objectives?

## □ Objectives

- Share some experiences regarding how to automate the build and deploy phases when developing large and complex J2EE-applications

## □ Bottom Line

- Increased quality and shortened delivery time in the build and deploy phase

## □ **Final Note**

- This presentation only describe one possible way of doing it. There are many other alternatives!
  - Download the examples and try them out
    - <http://www.callista.se/enterprise/resources/>

# Questions?

