


neo:developer 2002

s3 How to improve development of J2EE Components using...

## Sun's Core J2EE Patterns

Johan Eltes  
J2EE and EAI Architect




New: Typos fixed 2002-02-26

### The Speaker

- Path to J2EE
  - 2 years Mid-range, Mainframe (OOA + ??D + Cobol)
    - S/36 85-86
    - CICS 87
    - TSO 89-90
  - 10+ years of OOD/OOP
    - Smalltalk 88-93
    - Objective-C 93-96
    - Java 97 -
  - 7 years distributed objects
    - NeXT PDO 93-96
    - Corba 96-98
    - DCOM/MTS 97-00
    - RMI/EJB 98-02
- J2EE Experience
  - 8 EJB projects:
    - 3 BEA WebLogic
    - 5 WebSphere
  - JCA Project:
    - WebSphere / XA
  - JMS Project:
    - MQSeries
- EAI Experience
  - Platform evaluation and implementation
    - Industri and Utility
  - MQSI, Mercator, Tibco

How to improve ... with Sun's Core J2EE Patterns  
Copywrite Callista Enterprise AB 2002

2




Project Manager  
Designer  
Architect

## Agenda

- ✍ The Project
  - ✍ *The business case*
  - ✍ *Requirements*
  - ✍ *Staffing and process*
- ✍ Introduction to J2EE Patterns
  - ✍ *Background*
  - ✍ *Layers*
  - ✍ *Patterns*
- ✍ The Reference Architecture
  - ✍ *Definition (RUP)*
  - ✍ *Motivation*
  - ✍ *Chosen component model*
  - ✍ *Mapping to J2EE*
- ✍ Merging Ref. Arch and J2EE Patterns
  - ✍ *Applied patterns*
  - ✍ *Chosen strategies*
  - ✍ *Open issues*
- ✍ Extensions
  - ✍ *Platform abstraction*
  - ✍ *Component Configuration*
  - ✍ *Exception handling*
  - ✍ *Transaction control*
- ✍ The model-driven design process
  - ✍ *Design process*
  - ✍ *Business Components*
  - ✍ *Application Components*
  - ✍ *EJB*
  - ✍ *Assembly*
- ✍ Summary and Questions

How to improve ... with Sun's Core J2EE Patterns  
Copyright Callista Enterprise AB 2002




3

## Project Background: The Business case

- ✍ Volvo Information Technology AB
  - ✍ *Global application development and operations for AB Volvo*
  - ✍ *Global outsourcing partner for external customers*
  - ✍ *A staff of 4 000 employees and contractors*
  - ✍ *Turnover: Sek 5 billion*
- ✍ Future platforms are J2EE and MS .Net
- ✍ Driven by customer strategies
  - ✍ *AB Volvo is many customers!*

How to improve ... with Sun's Core J2EE Patterns  
Copyright Callista Enterprise AB 2002



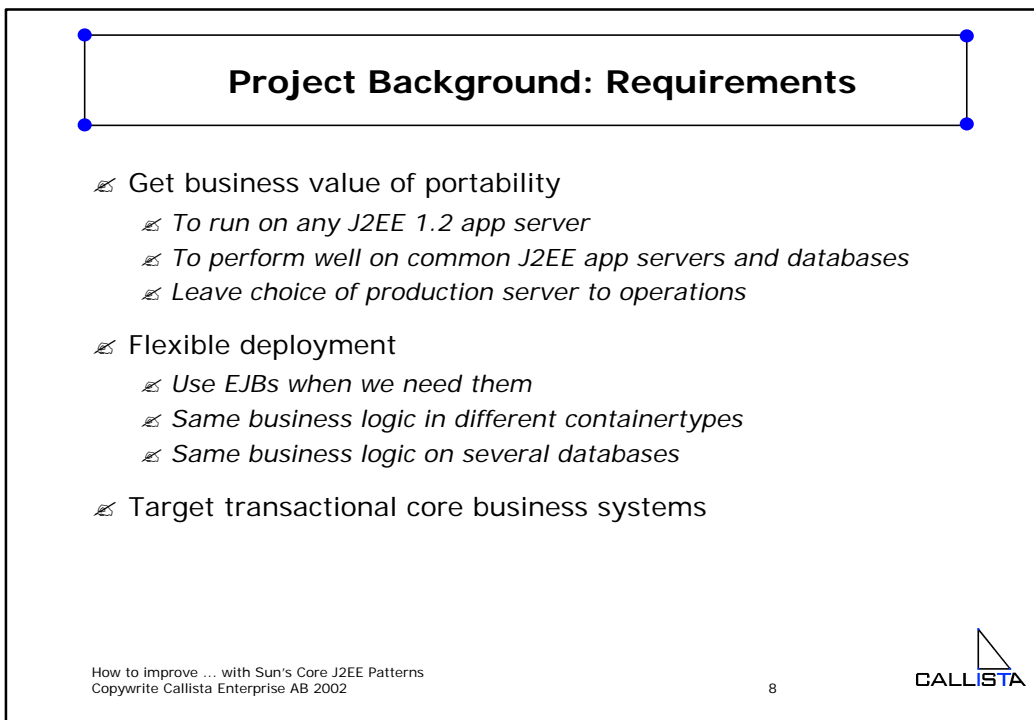
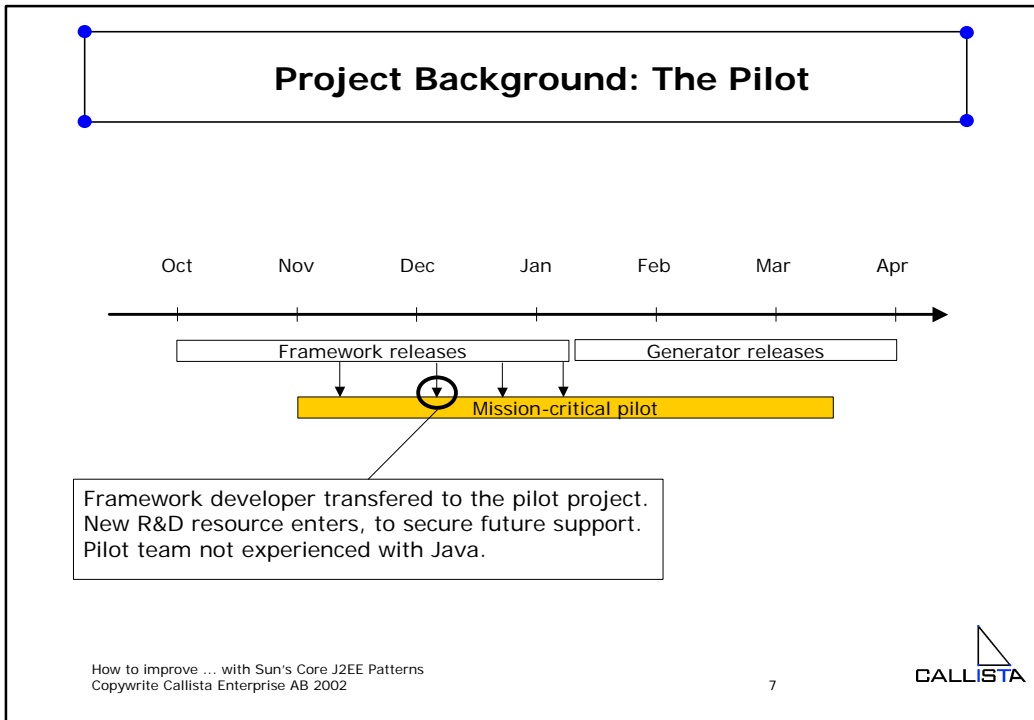
4

## Project Background: The Business case

- ⌘ The Java Platform Project – JAPP
  - ⌘ *Initially started to standardise the tool set...*
  - ⌘ *...and provide tool support / mentoring*
  - ⌘ *Rebirth as global project in September 2001 (SE, US, FR)*
  - ⌘ *Pure R&D project*
- ⌘ Project goal
  - ⌘ *Deliver an efficient development process for enterprise systems based on main-stream tools and best-practise*
- ⌘ The history
  - ⌘ *J2EE is too tough for developers with a pure application (customer!) focus*
  - ⌘ *All projects put far too much efforts on problems related to the technical domain*
  - ⌘ *Re-use of experience does not give the required jump-start to the next project*
  - ⌘ *Suns J2EE Blueprints did not keep up to its promise as a "project booster"*
  - ⌘ *EJBs avoided due to complexity and lack of experience*

## Project Background: The Project

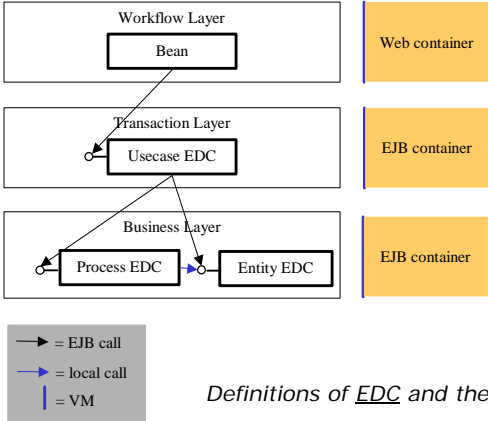
- ⌘ Phases
  - ⌘ *September: Project set-up*
    - ⌘ *Definition of a reference architecture*
    - ⌘ *Planning*
  - ⌘ *October – Januari: The none-automated process*
    - ⌘ *Design of the BCVM (platform abstraction)*
    - ⌘ *Design of implementation strategies - Pattern-by-pattern*
    - ⌘ *Design of "missing patterns"*
    - ⌘ *Design and development of supporting Java framework/library*
    - ⌘ *Development of a sample business system (quality assessment)*
  - ⌘ *February – March: The automated process*
    - ⌘ *Definition of UML-mappings (abstract design modeling)*
    - ⌘ *Design and development of generators*
    - ⌘ *Tool integrations*
    - ⌘ *Build process*
    - ⌘ *Stress test, purifying code*
    - ⌘ *Design- and programming guidelines*



### Project Background: Requirements

*Use EJBs when they add value!*  
*Scenario 3 VMs, 2 EJB layers*


- ✗ Enterprise systems
  - ✗ + All EDCs distributed
  - ✗ + Full EJB security
  - ✗ - Not optimized
  - ✗ - Not batch



→ = EJB call  
 → = local call  
 | = VM

*Definitions of EDC and the layers later...*

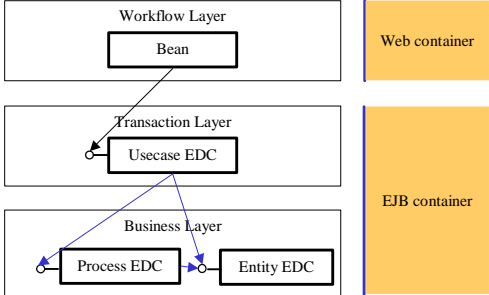
How to improve ... with Sun's Core J2EE Patterns  
 Copyright Callista Enterprise AB 2002

9 


### Project Background: Requirements

*Scenario: 2 VMs, 1 EJB layer*

- ✗ Large systems
- ✗ + Optimized to some extent
- ✗ - No EJB Security on business layer
- ✗ - No distr. access to Business layer
- ✗ - Not batch



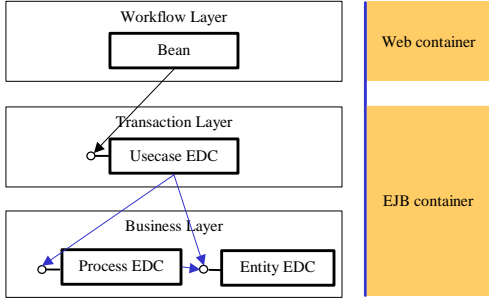
How to improve ... with Sun's Core J2EE Patterns  
 Copyright Callista Enterprise AB 2002

10 


## Project Background: Requirements

*Scenario: 1 VM, 1 EJB layer*

- ✍ Smaller systems
- ✍ + Fairly Optimized
- ✍ - Monolithic deploy
- ✍ - No EJB Security on business layer
- ✍ - Not batch



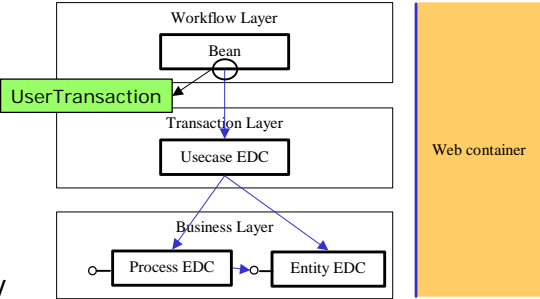
How to improve ... with Sun's Core J2EE Patterns  
 Copyright Callista Enterprise AB 2002

11 


## Project Background: Requirements

*Scenario: 1 VM, No EJBs, With JTA*

- ✍ Smaller systems
- ✍ + Optimized
- ✍ - Monolithic deploy
- ✍ - No EJB security
- ✍ - Not batch



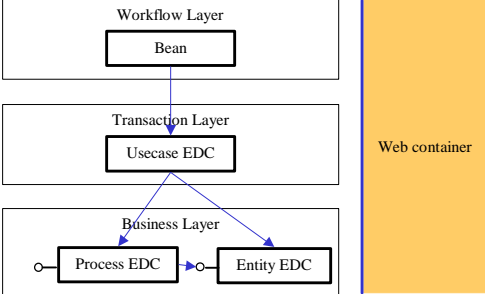
How to improve ... with Sun's Core J2EE Patterns  
 Copyright Callista Enterprise AB 2002

12 

## Project Background: Requirements


*Scenario: 1 VM, No EJBs, No JTA*

- ⌘ Tomcat on user-workstation
- ⌘ + Optimized
- ⌘ + Light-weight infrastructure
- ⌘ - Monolithic deploy
- ⌘ - Single-threaded
- ⌘ - Single-user
- ⌘ - No 2PhC (single database)
- ⌘ - No EJB security
- ⌘ - No batch



The diagram shows a vertical stack of three layers within a yellow 'Web container' box. The top layer is 'Workflow Layer' containing a 'Bean'. An arrow points down to the 'Transaction Layer' containing 'Usecase EDC'. From 'Usecase EDC', two arrows point down to the 'Business Layer', which contains 'Process EDC' and 'Entity EDC'. There is also a bidirectional arrow between 'Process EDC' and 'Entity EDC'.

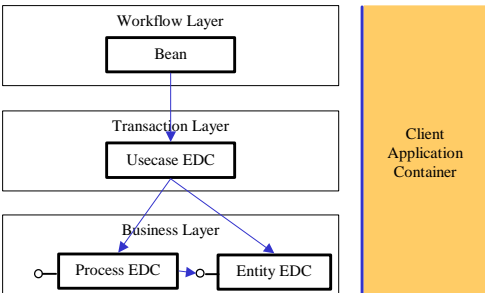
How to improve ... with Sun's Core J2EE Patterns  
 Copywrite Callista Enterprise AB 2002

13 

## Project Background: Requirements


*Scenario: 1 VM, No EJBs, No JTA*

- ⌘ Batch
- ⌘ + Light-weight infrastructure
- ⌘ - Monolithic deploy
- ⌘ - Single-threaded
- ⌘ - Single-user
- ⌘ - No 2PhC (single database)
- ⌘ - No EJB security



The diagram shows a vertical stack of three layers within a yellow 'Client Application Container' box. The top layer is 'Workflow Layer' containing a 'Bean'. An arrow points down to the 'Transaction Layer' containing 'Usecase EDC'. From 'Usecase EDC', two arrows point down to the 'Business Layer', which contains 'Process EDC' and 'Entity EDC'. There is also a bidirectional arrow between 'Process EDC' and 'Entity EDC'.

How to improve ... with Sun's Core J2EE Patterns  
 Copywrite Callista Enterprise AB 2002

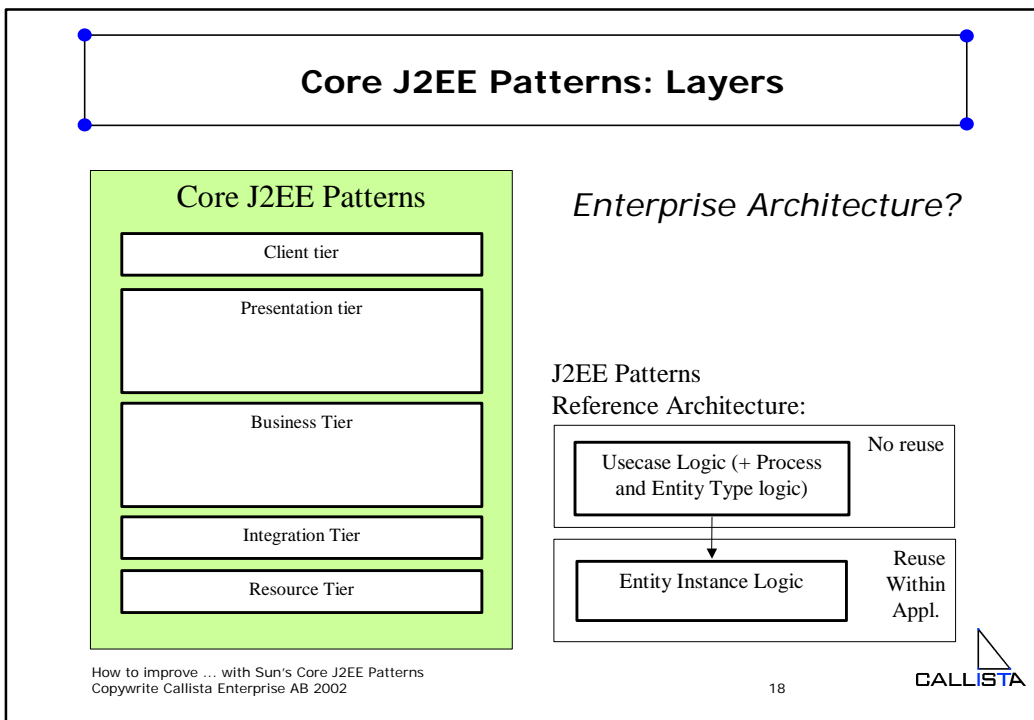
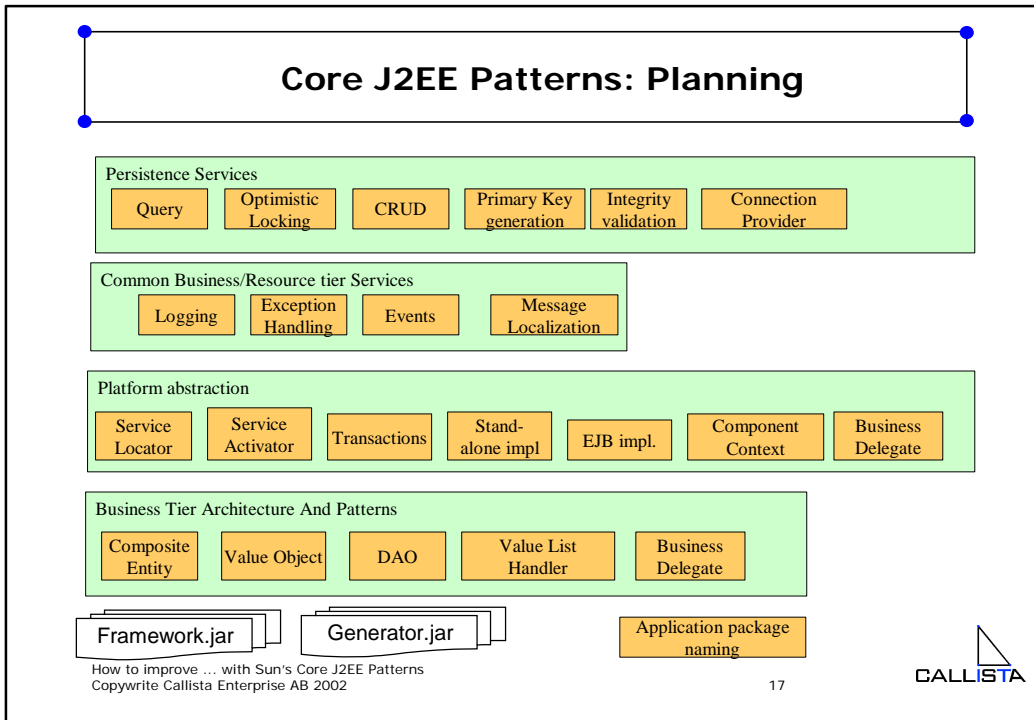
14 

## Project Background: The requirements led us to...

- ✗ No Entity Beans
  - ✗ *Would risk "business value portability"*
  - ✗ *Does not support a database-centric programming model*
    - ✗ In core transactional systems, database constraints are part of business logic
    - ✗ Entity beans generate system exceptions, which can not be handled by application
  - ✗ *Not mature in J2EE 1.2*
  - ✗ *Not critical to project success, assuming automated process*
  - ✗ *Too much complexity for the value – DAO pattern OK*
  - ✗ *Too large concept to manage/simulate without container*
- ✗ Platform abstraction
  - ✗ *Isolate app logic from EJB infrastructure*
  - ✗ *Positive side affects:*
    - ✗ Shields app developer from EJB Exception- and Transaction semantics
    - ✗ Shorter development cycles
    - ✗ Easier to achieve automated tests
- ✗ Look for best-practice
  - ✗ *Suns J2EE Patterns*
  - ✗ *Open Source frameworks and libraries (Struts, Log4j, xerces, junit etc)*

## Introduction to Sun's Core J2EE Patterns

- ✗ Give names to common problems
- ✗ Sun effort parallel to Blueprint Patterns
- ✗ Structured around layers
- ✗ Some names are already common vocabulary:
  - ✗ *DAO, Value Object, Session Facade, Business Delegate, Front Controller*
  - ✗ *...which is one of their most valuable properties*
- ✗ Worked well as subject for planning
- ✗ Not as formal (and well-defined) as GOF patterns



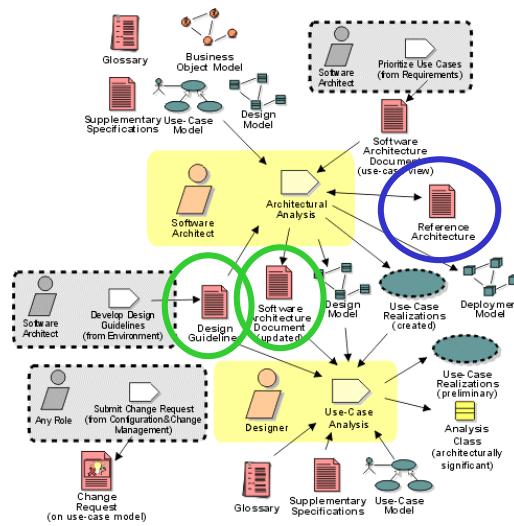
## Core J2EE Patterns: Layers

*Not sufficient as an enterprise business component architecture*

- ✗ No Business System View
- ✗ No domain layer – no reuse of business logic
  - ✗ Reuse of BMP entity (only instance logic)
- ✗ No rules for subsystems and subsystem collaboration.
- ✗ Implicit co-packaging of presentation tier and integration tier (DAO-access from presentation tier)

## Reference Architecture for J2EE business systems

RUP: "A Reference Architecture is, in essence, a predefined architectural pattern, or set of patterns, possibly partially or completely instantiated, designed and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects."



## Reference Architecture: Japp approach

- ⌘ Herzum/Sims "Business Component Factory"
  - ⌘ J2EE Sun's Core J2EE Patterns
  - ⌘ JAPP Business component model
  - ⌘ J2EE
  - ⌘ Architectural Framework
    - ⌘ JAPP framework / generators
    - ⌘ Struts Web Gui framework
    - ⌘ Log4J – Logging framework
- "A Reference Architecture is, in essence, a predefined architectural pattern or set of patterns possibly partially or completely instantiated designed and proven for use in particular business and technical contexts together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects."

## Reference Architecture: With less words...

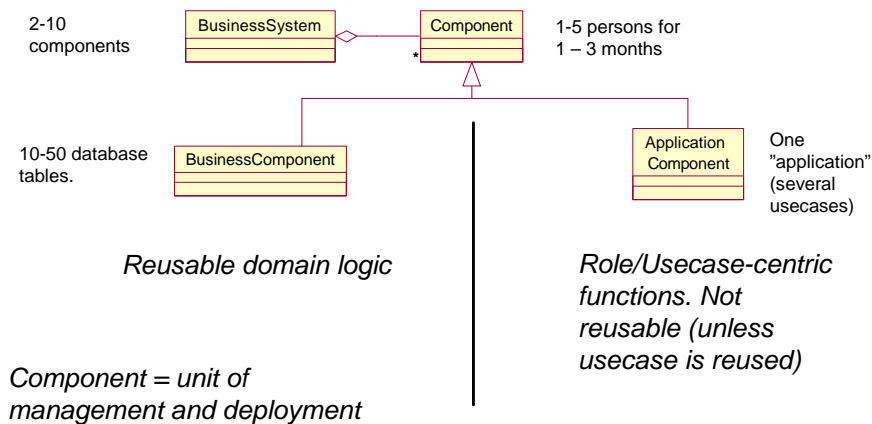
*"The principles and the means to most effectively achieve a design vision"*

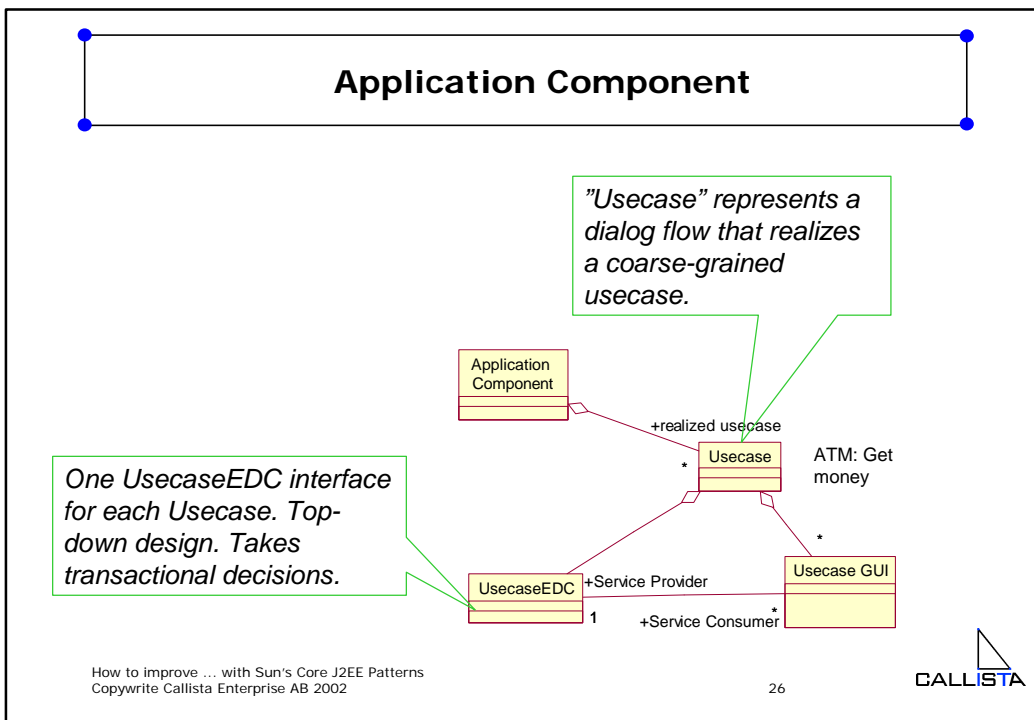
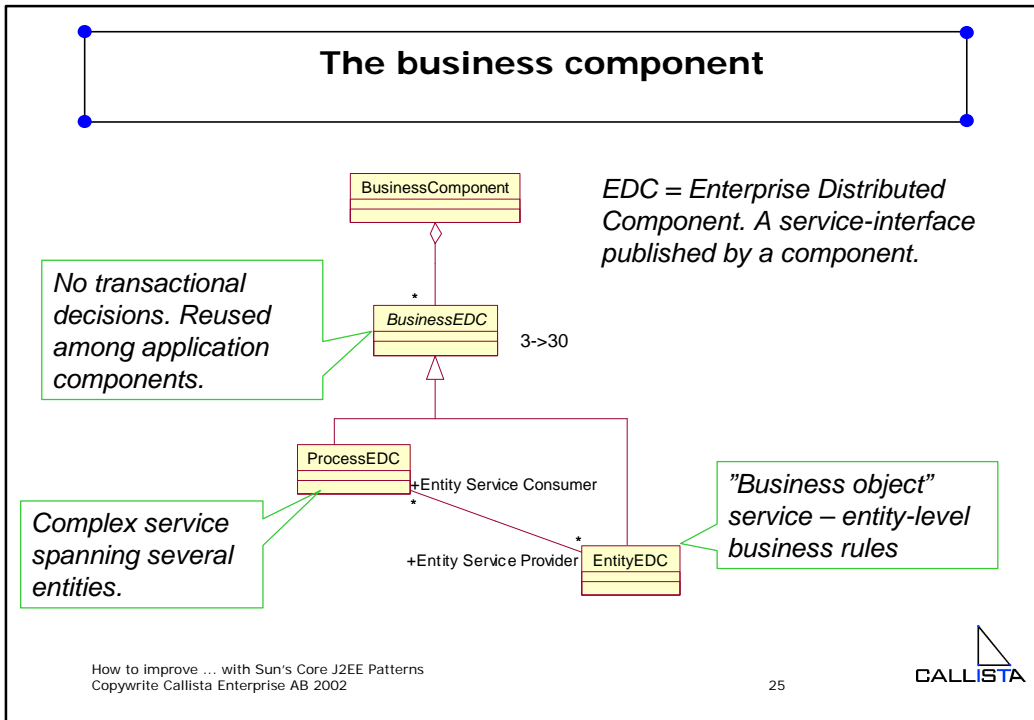
*Richard Hubert, OMG / Interactive Objects*

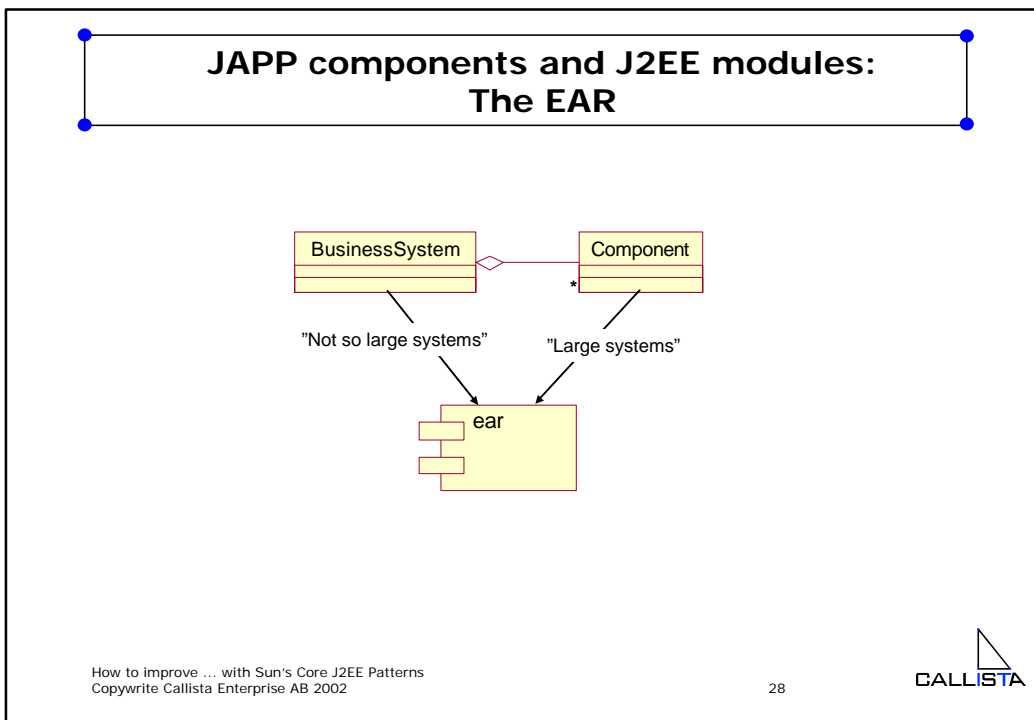
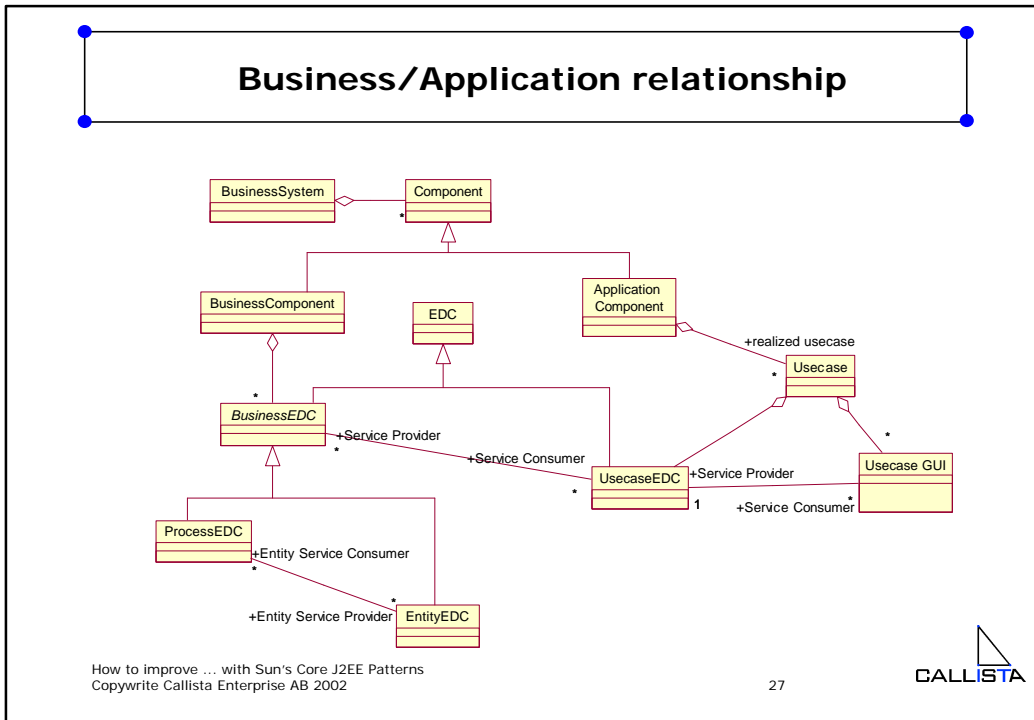
## Reference Architecture: Motivation

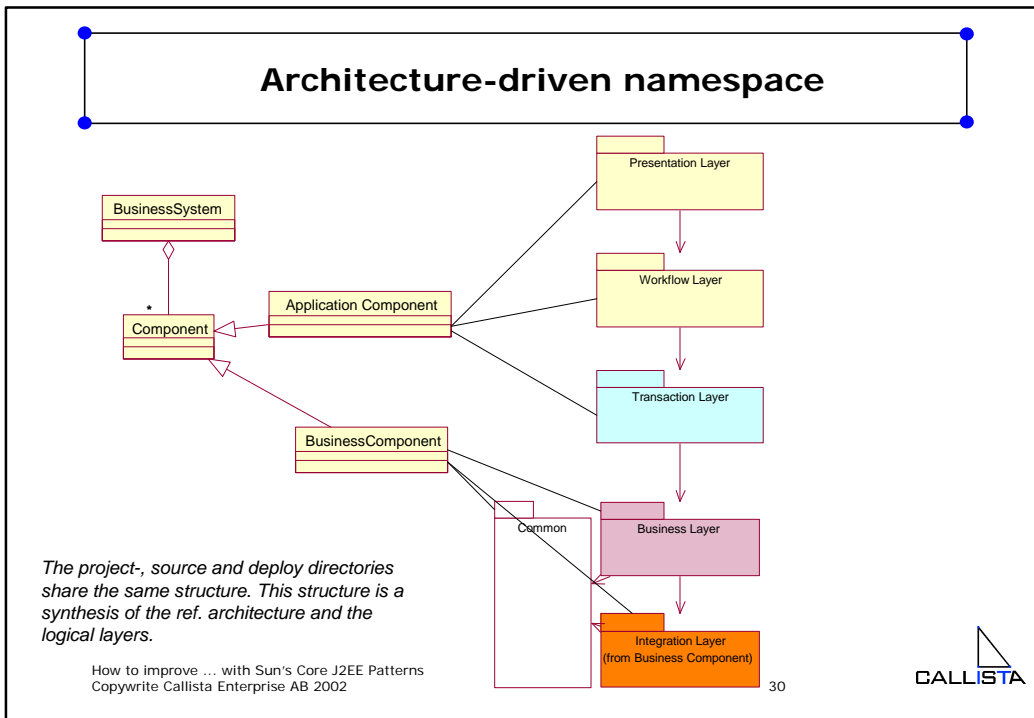
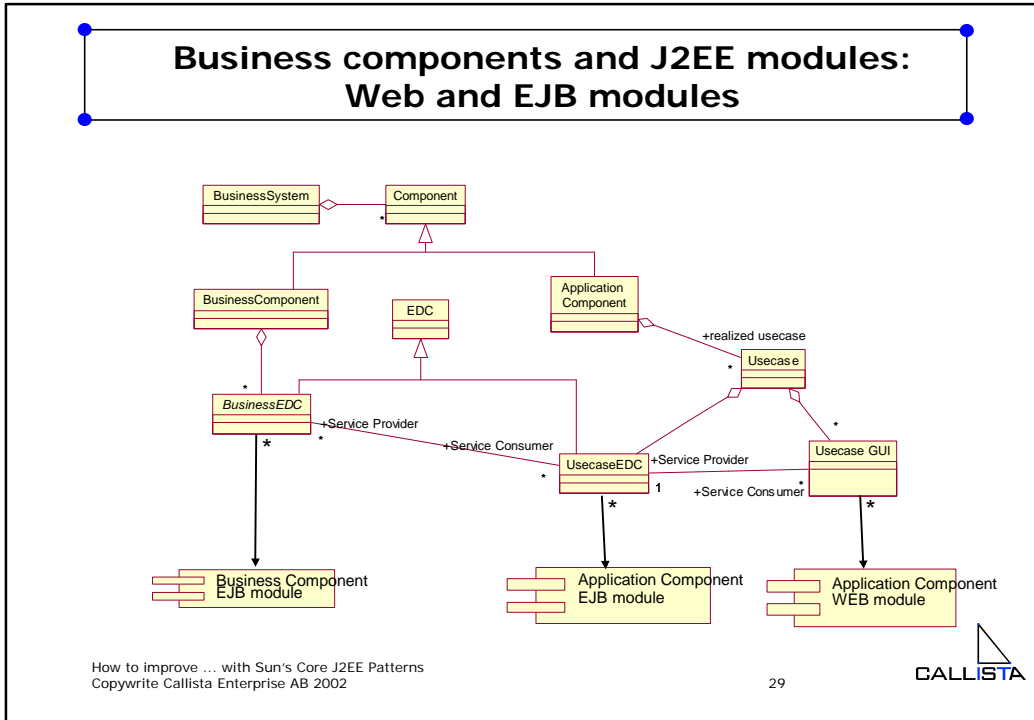
- ✎ Supports large-grained components
- ✎ Autonomous building blocks of an enterprise business system
- ✎ Unit of management
- ✎ Development teams design, develop and test components
- ✎ Units of deployment
- ✎ Defines the logical layers
- ✎ Defines an enterprise namespace
  - ✎ Business systems
  - ✎ Business components
  - ✎ J2EE modules
  - ✎ JNDI names
  - ✎ Unit of Configuration
  - ✎ Directory structures etc

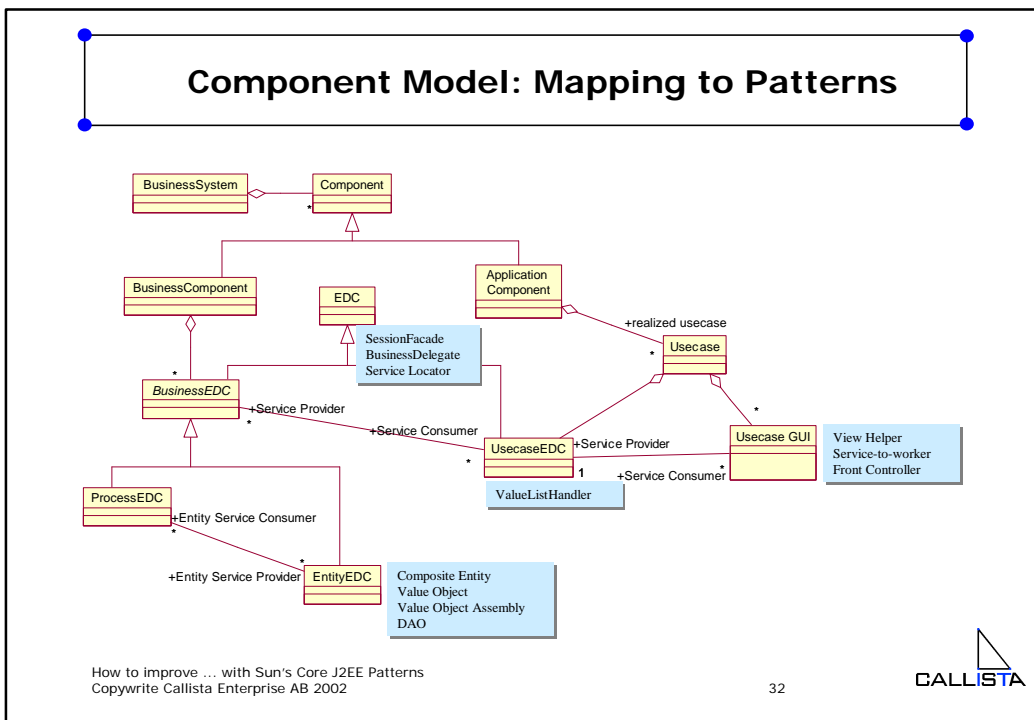
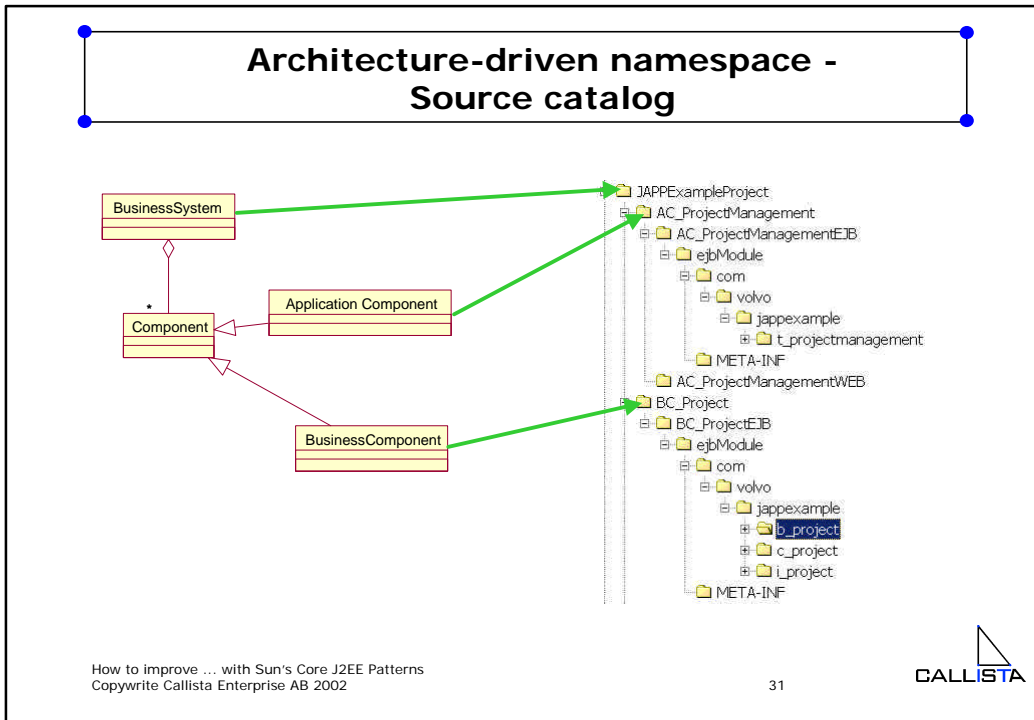
## Reference Architecture: Chosen component model

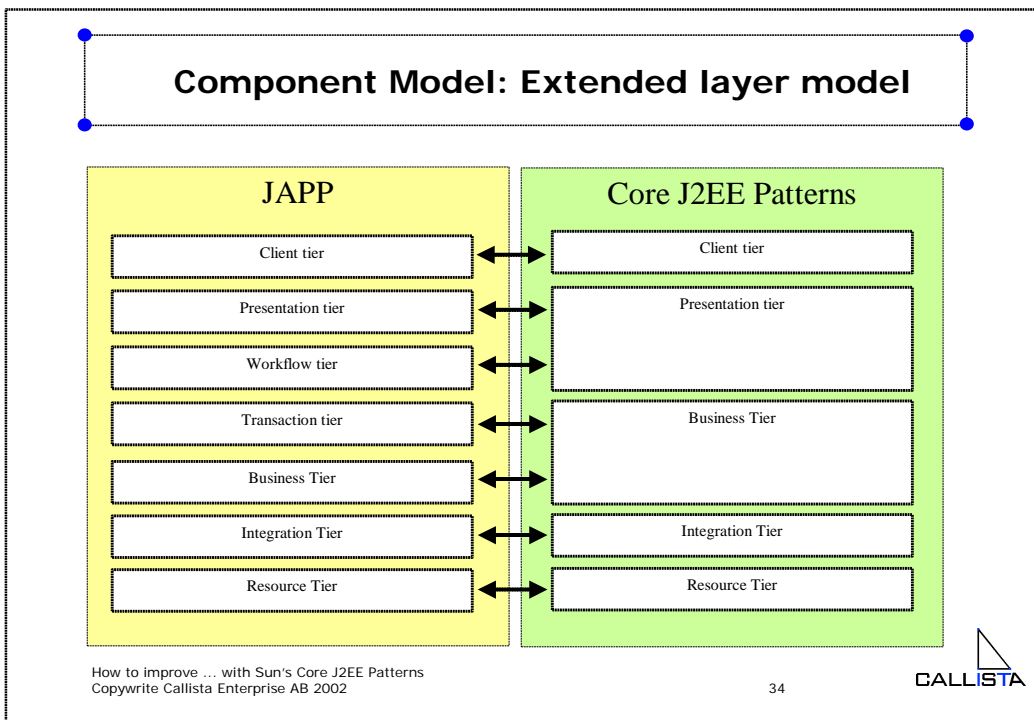
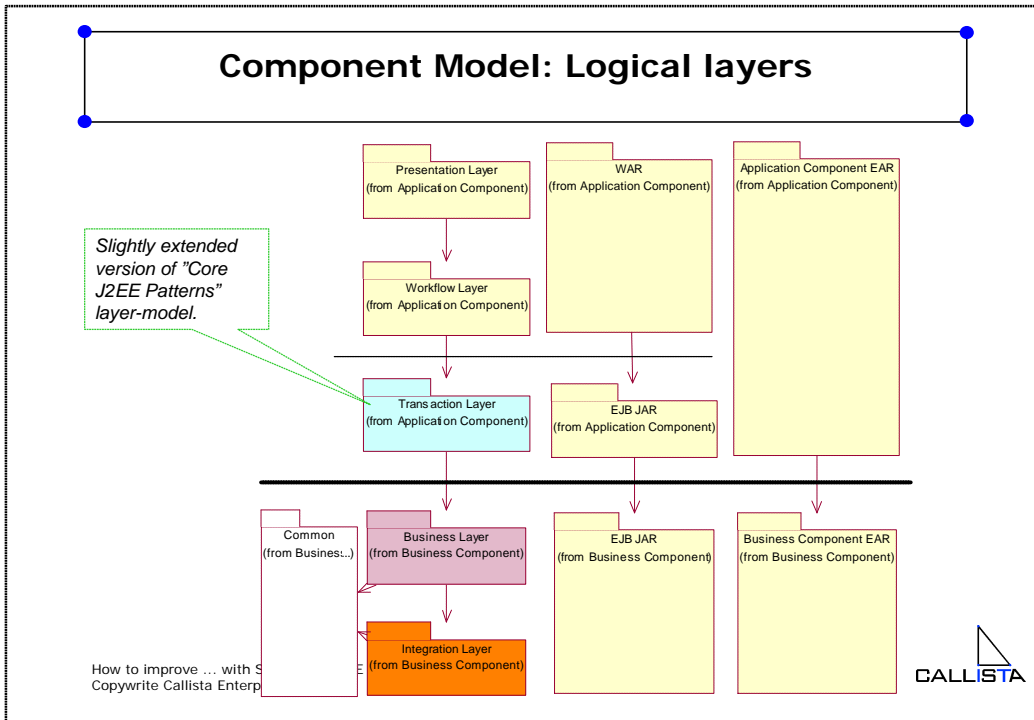












## Some aspects not covered by Sun's patterns

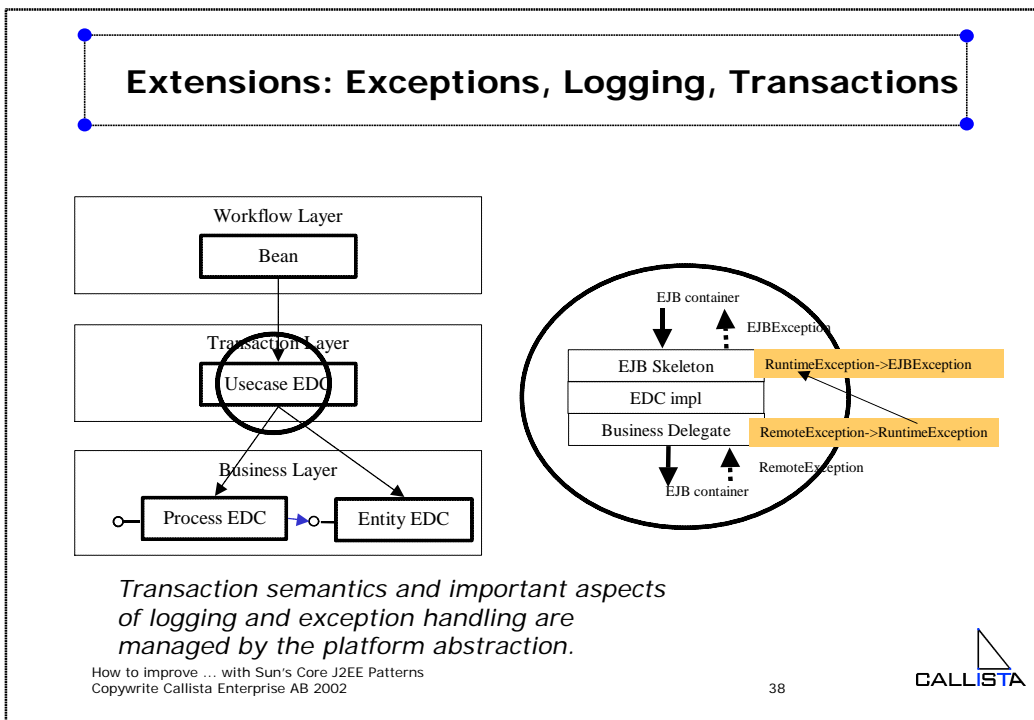
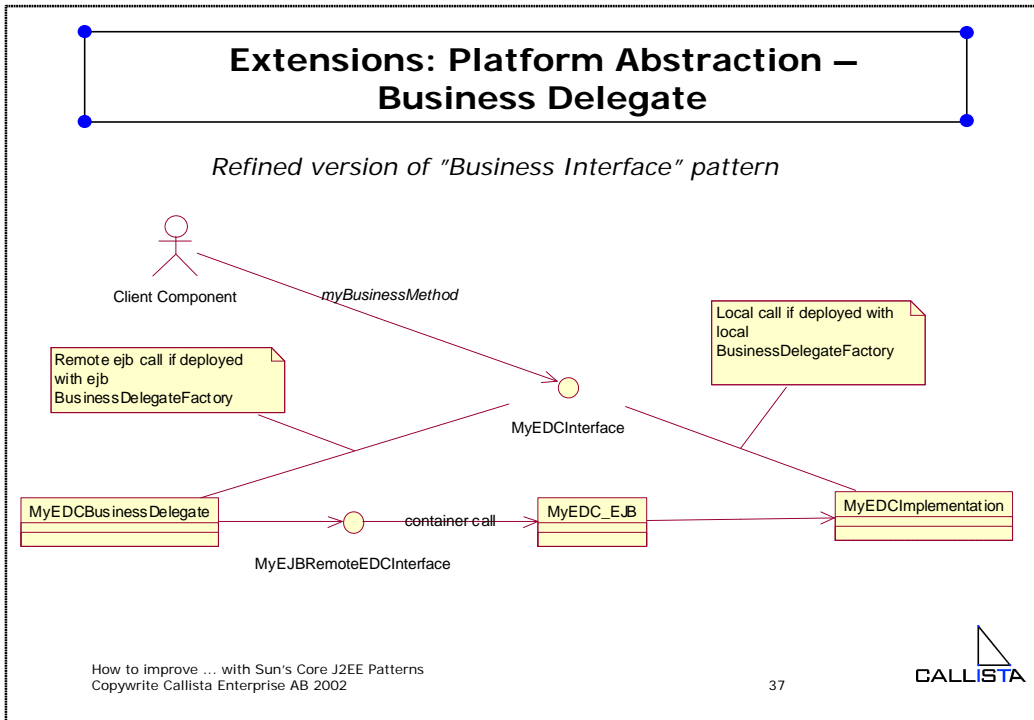
*Most critical areas for JAPP to resolve:*

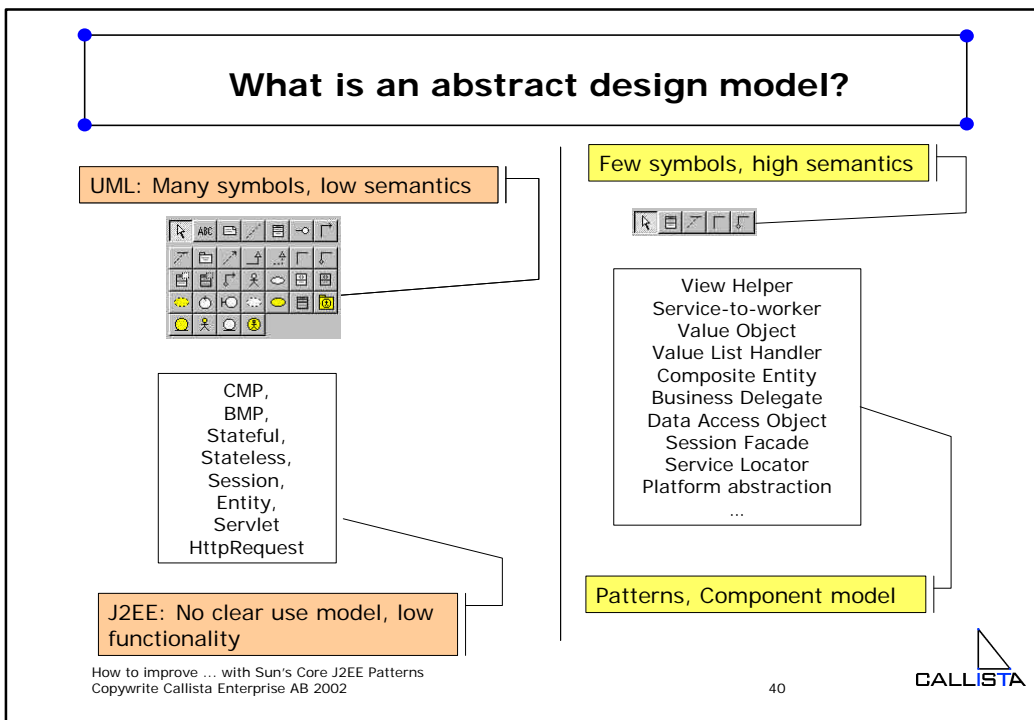
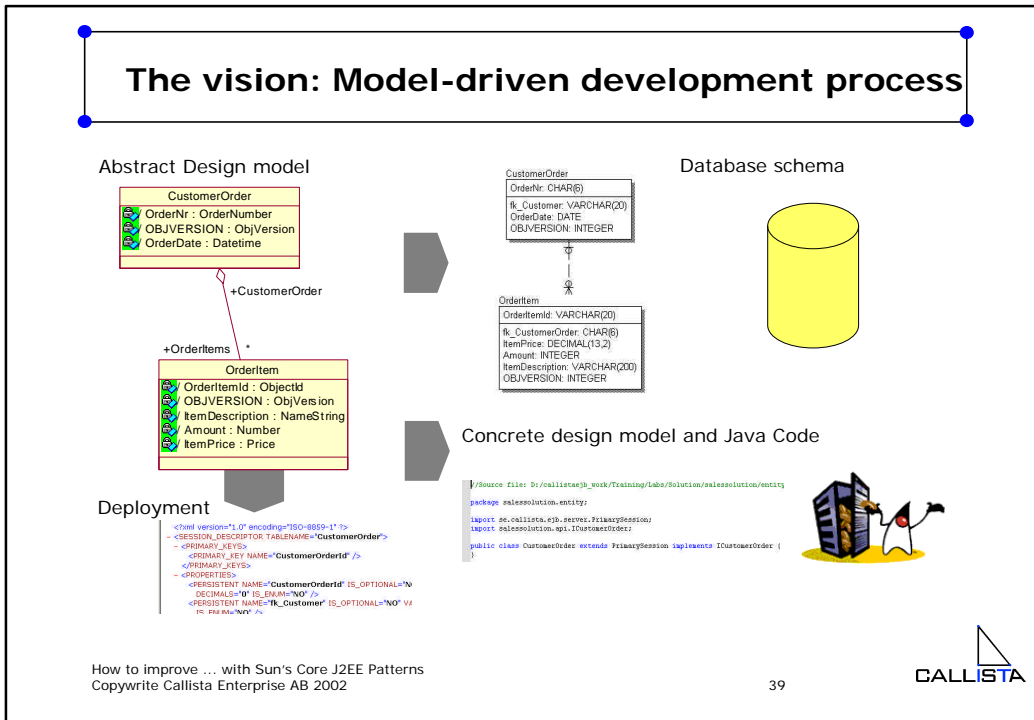
- ✍ Exception handling
- ✍ Flexible deployment
- ✍ Pluggable services (partly covered for DAOs)
- ✍ Logging and Localization
- ✍ Transaction semantics
- ✍ Find a better strategy for ValueListHandler

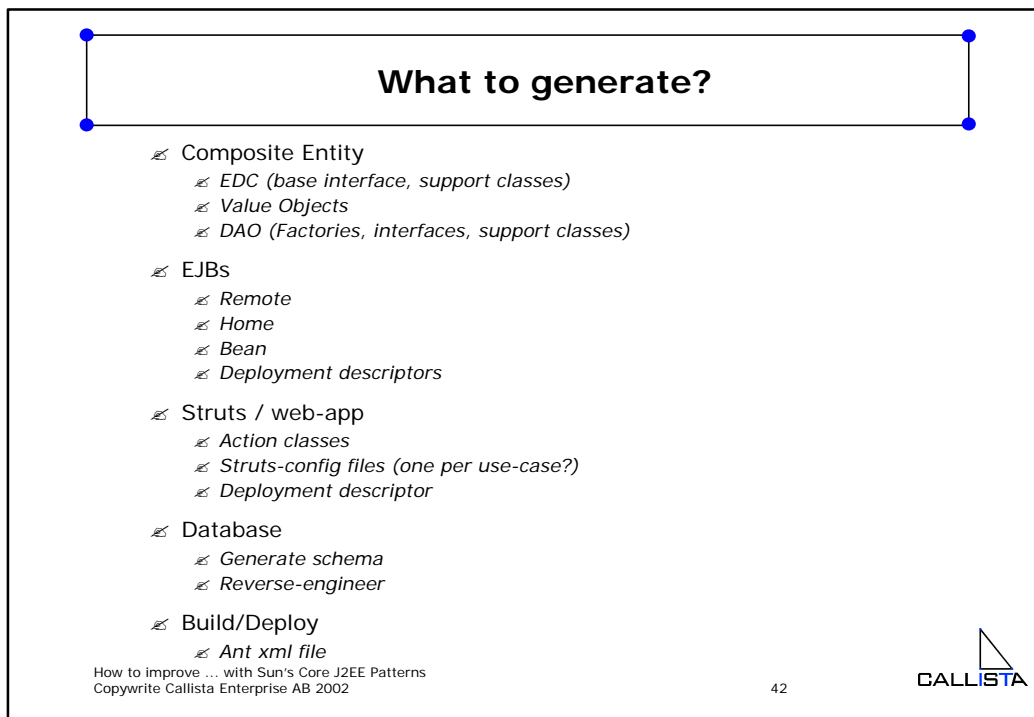
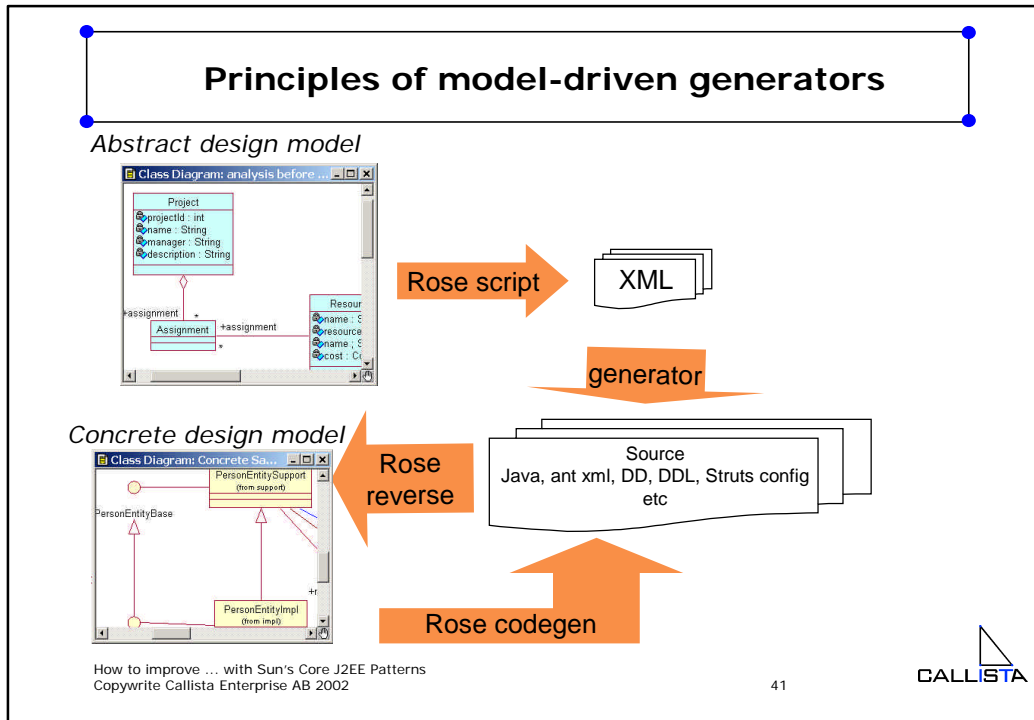
## Extensions: Platform Abstraction

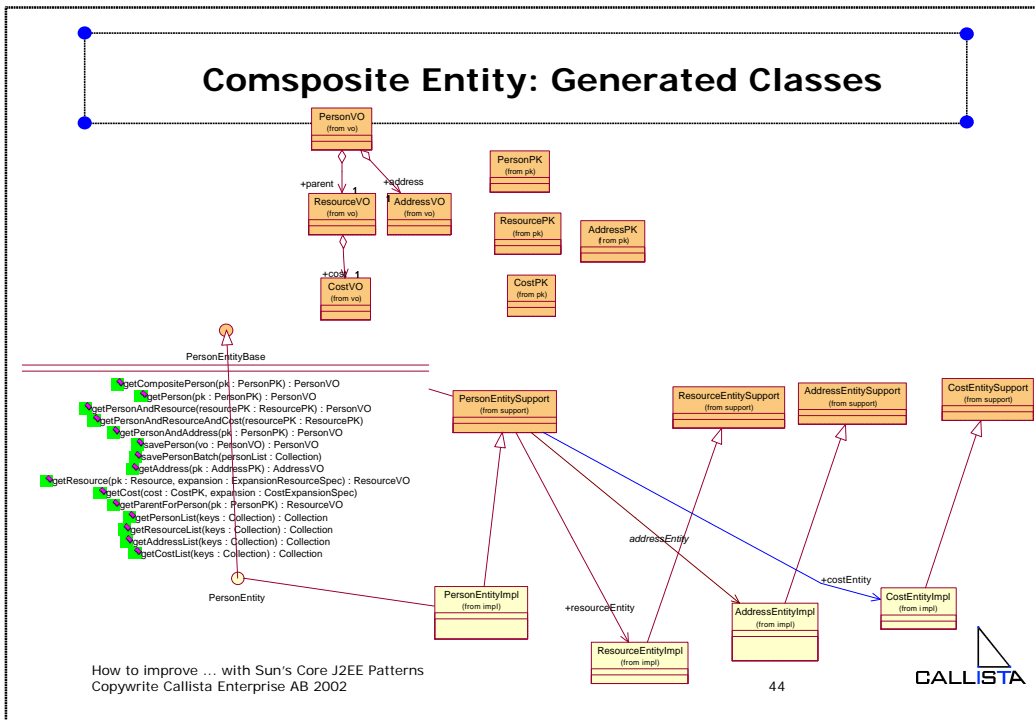
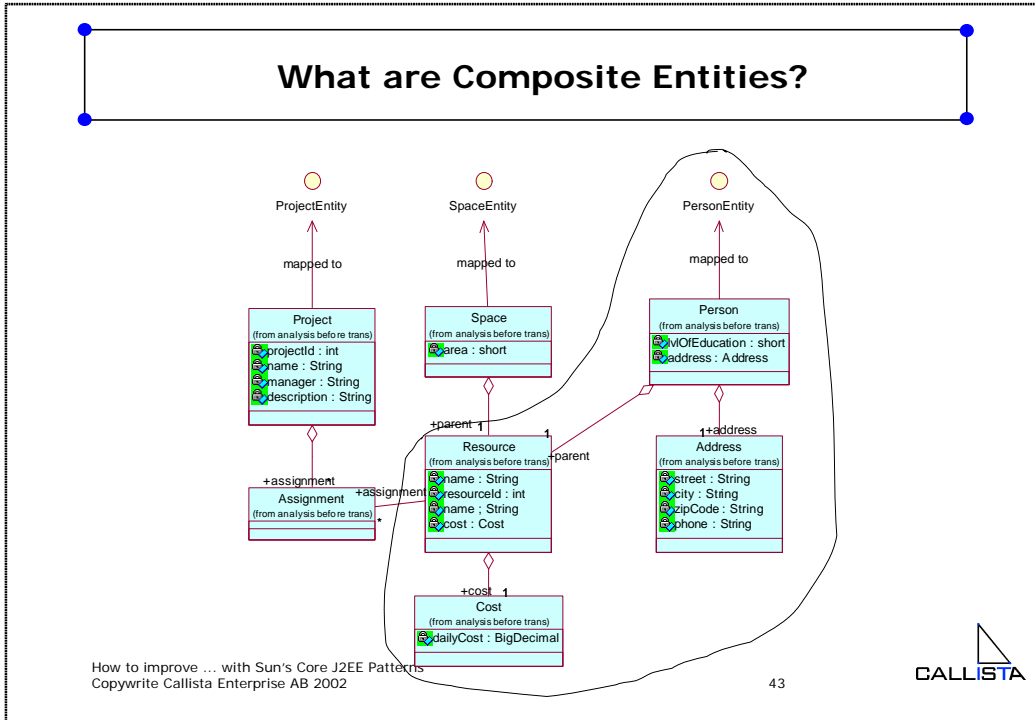
**From "Business Component Factory":**  
**BCVM = Business Component Virtual Machine**

- ✍ Shielding the developer from EJBs resolved many issues!
  - ✍ *Transaction handling (when and how to rollback)*
    - ✍ Tightly coupled with exception handling
  - ✍ *Flexible deployment*
    - ✍ The abstraction code hides the activation model
- ✍ Implementation Strategy
  - ✍ *Pluggable business delegate factory*
  - ✍ *Delivers bean or JDK 1.3 proxy*





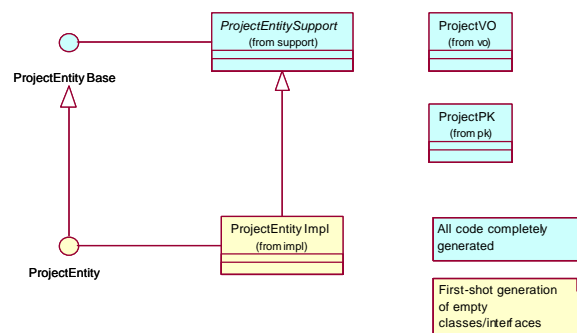




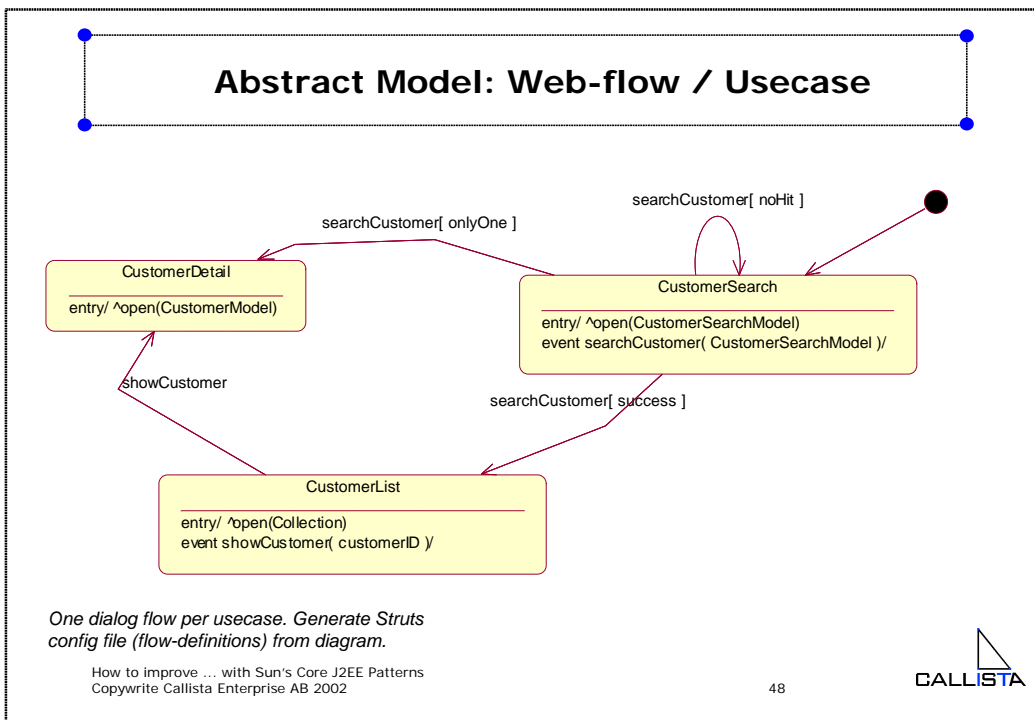
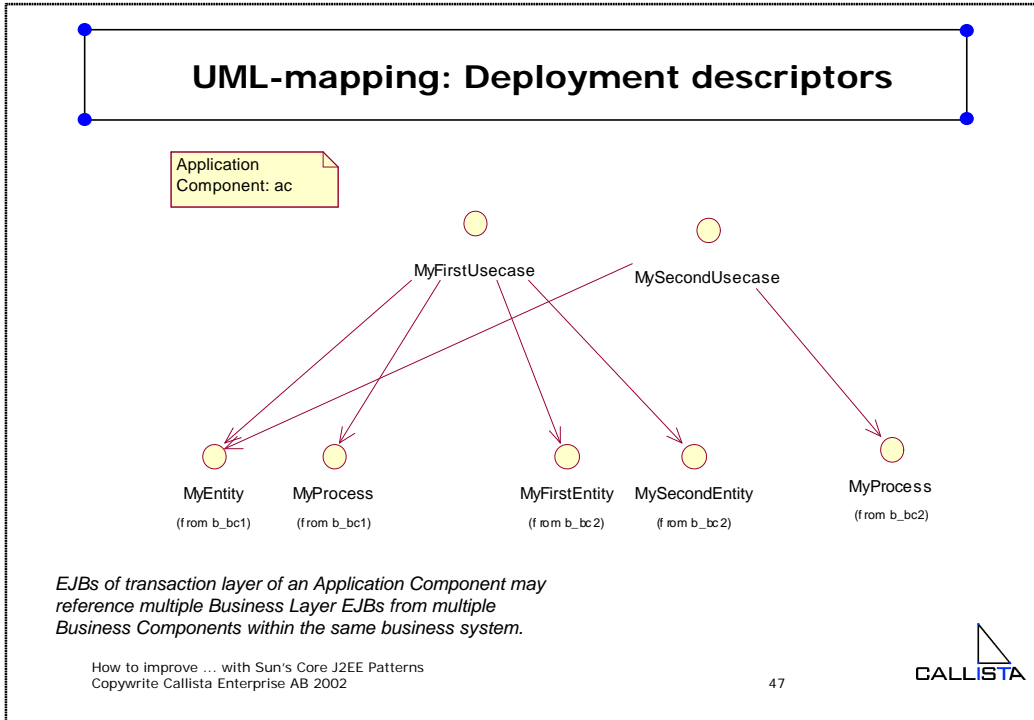
## UML mapping: Entity EDC vs DAO

- ⌘ DAO – Data Access Objects
  - ⌘ Persistence operations for a value object with a related PK object
  - ⌘ Hides type and structure of the underlying resources
  - ⌘ Fine-grained persistence operations: **insert, update, delete**
- ⌘ Entity (Composite)
  - ⌘ Composites, to represent
    - ⌘ "Business documents"
    - ⌘ Complex types
    - ⌘ Inheritance
  - ⌘ Entity-level business rules (template method)
    - ⌘ Validation rules
    - ⌘ "Triggers"
    - ⌘ Resolves DAO constraint exceptions into app-specific exceptions
  - ⌘ Coarse-grained persistence operation: **save...**

## UML mapping: Composite Entities...



Generated artifacts for a flat entity (not composite)



## Summary

- ⌘ Model-driven architecture
  - ⌘ *Enterprise Component Model*
  - ⌘ *Patterns*
  - ⌘ *Supporting tools and frameworks*
  - ⌘ *Tight, automated process*
- ⌘ Applying the patterns
  - ⌘ *Nail them to a reference architecture!*
  - ⌘ *You need most of them!*
  - ⌘ *Good: Names for important problem domains*
  - ⌘ *Not so good: The implementation strategies*
  - ⌘ *Use the strategies for problem understanding – not as solutions*
  - ⌘ *Composite Entity is complex, but valuable*
- ⌘ Core J2EE Pattern conclusions
  - ⌘ *Far from GOF quality*
  - ⌘ *Still alot to cover*
  - ⌘ *Evolution?*
  - ⌘ *Very good foundation for planning and communication!*

## Thank you for your attention!

[www.callista.se/enterprise](http://www.callista.se/enterprise)  
[Johan.eltes@callista.se](mailto:Johan.eltes@callista.se)

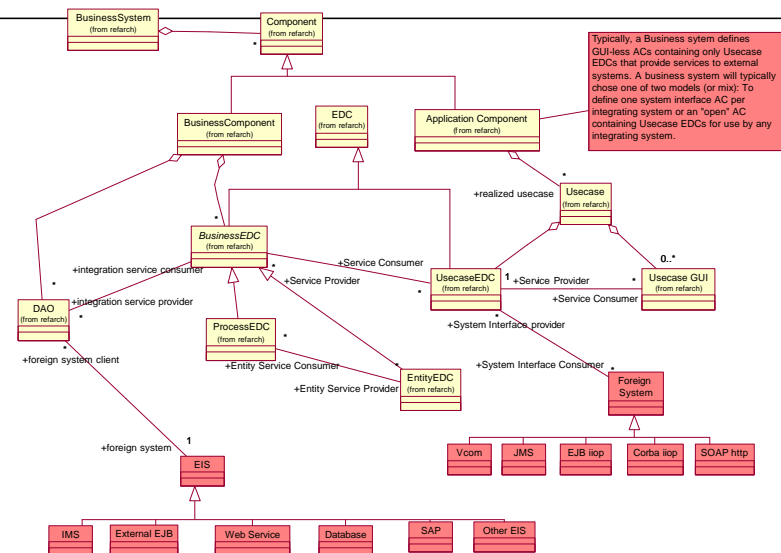
***This presentation:***

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

## Questions

- ✍ Ref.arch: How do business systems interact?
- ✍ Ref.arch: How is transaction demarcation mapped to the ref.arch?
- ✍ BCVM: How to specify EJB vs none-EJB activation?
- ✍ ...

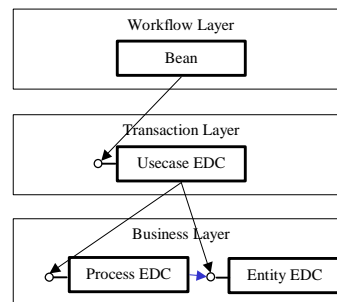
## Questions: How do business systems interact?



## Question: How is transaction demarcation mapped to the ref.arch.?

Trans.methods: RequiresNew  
No Trans methods: NotSupports

All methods: Supports



The BVCM hides setRollbackOnly from app.logic.  
Rollback is mapped to application exception returned by Usecase EDC.  
Implemented by EJB stubs for Usecase EDCs.

## Question: How to specify EJB vs none-EJB activation?

- ✗ The BusinessDelegate implementation is chosen at runtime based on Abstract Factory pattern. Different implementations:
  - ✗ EJB, local, UserTransaction
  - ✗ Future: LocalEJB
- ✗ BusinessDelegateFactory is one of many configurable services. Others are:
  - ✗ LoggingFactory
  - ✗ DataSourceFactory
  - ✗ UniqueKeyProviderFactory
  - ✗ PropertyFactory
- ✗ Each factory is associated with a Java interface
- ✗ Each J2EE module has a configuration file (XML)
  - ✗ One or more configurations for each factory interface
  - ✗ Each configuration has an identifier and a set of parameters

### Question: How to specify EJB vs none-EJB activation...

```
Factories.getProvider(this).  
    getFactory(com.volvo.japp.platform.BusinessDelegateFactory.class);
```

**Or...**

```
Factories.getProvider(this).  
    getFactory(com.volvo.japp.platform.BusinessDelegateFactory.class, "ejb");
```

**Then...**

```
myBusinessDelegateFactory.  
    getUsecaseEDCDelegate(  
        com.volvo.jappexample.t_projectmanagement.ManageProject.class);
```

### Question: How to specify EJB vs none-EJB activation...

✍ The model-driven generator generates a helper class per Component (ref.arch.)

```
ProjectManagementEDCs.getManageProject(this).assignPerson(...)
```

Transaction layer of application component "ProjectManagement"

Usecase EDC: "ManageProject"

The caller is a workflow layer bean of application component "ProjectManagement". Likely a Struts action class.