A Reflective Component-Based Transaction Architecture

Enterprise JavaBeans & Progress Update

Zhixue Wu

9th April 1998

Foot Steps

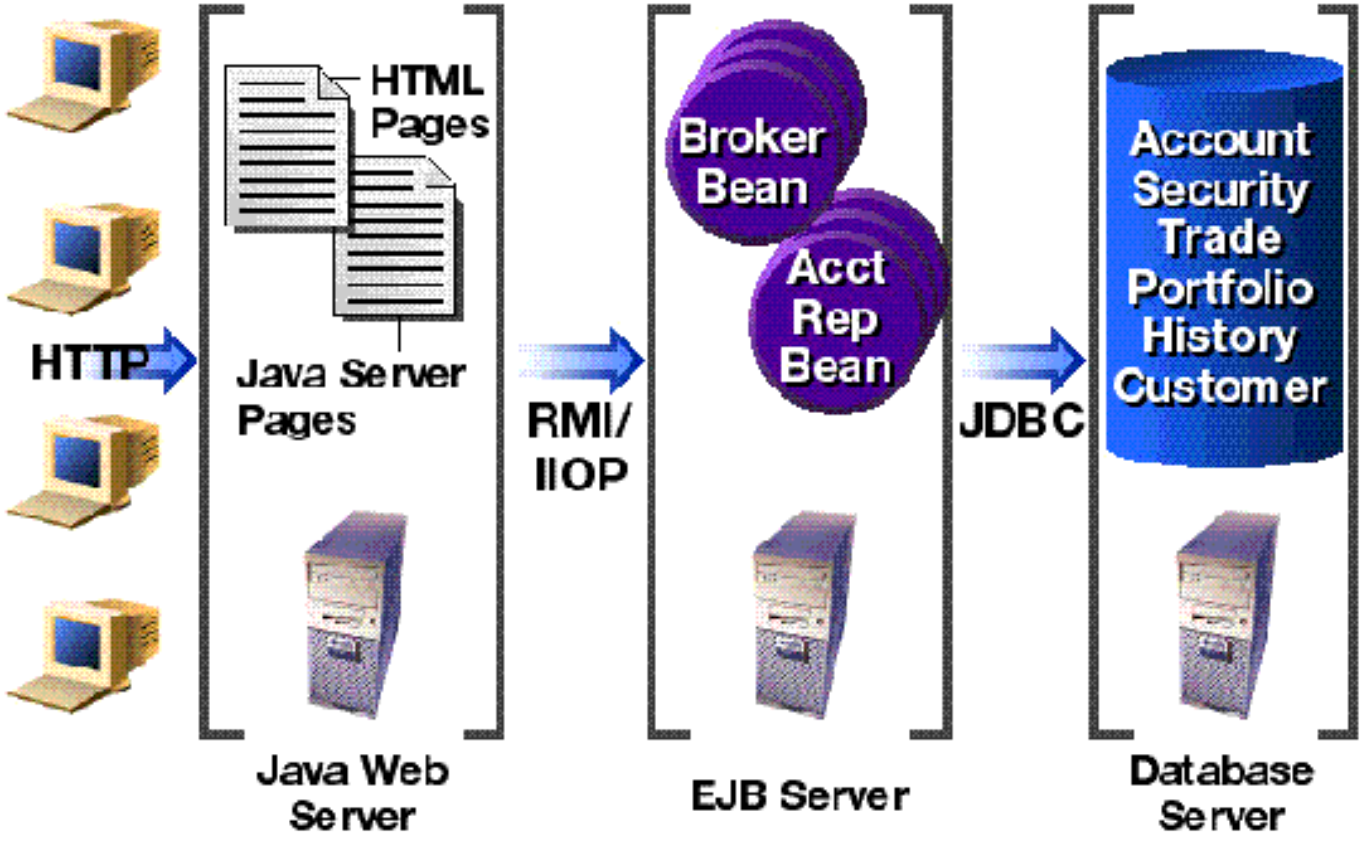
- Oct. 97: we proposed a declarative transaction architecture for Internet applications
  - focus on the middle tier development
  - declarative approach to transaction
  - flexible configuration for system capabilities
  - use application information for improving performance
- Dec. 97: Sun published it's Enterprise JavaBeans specification for comments
  - aims for helping middle tier development
  - supports implicit transaction
- Jan. 98: we aligned our position with EJB
- This talk:
  - summary of EJB
  - EJB vs. our work
  - progress update

Summary
of
Enterprise JavaBeans

Enterprise JavaBeans

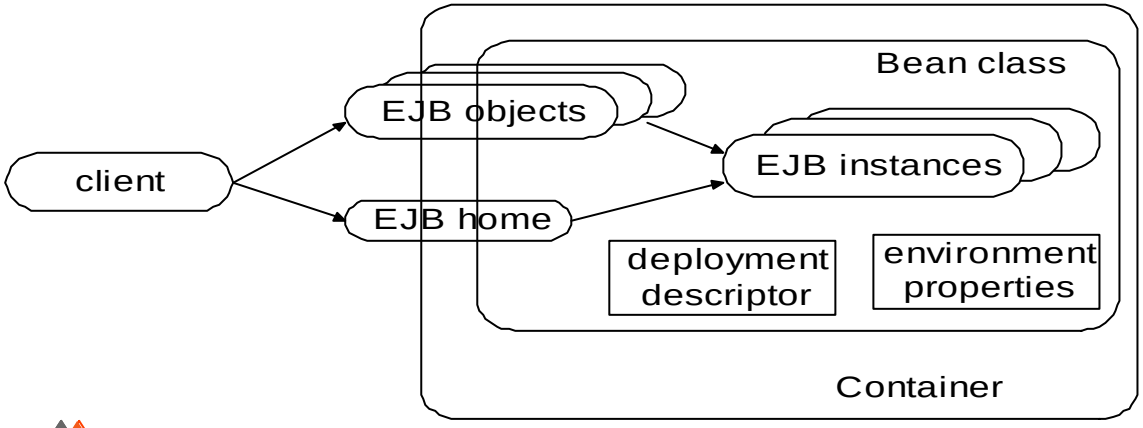
- Defines a component model for the development and deployment of multitier Java applications
- Easy to develop, deploy, and manage enterprise applications, enable Bean developers to ignore
  - transaction programming
  - multi-threaded programming
  - distributed programming
- Platform and system independent applications
  - develop once and deployed anywhere (any container)

Structure of an EJB Application



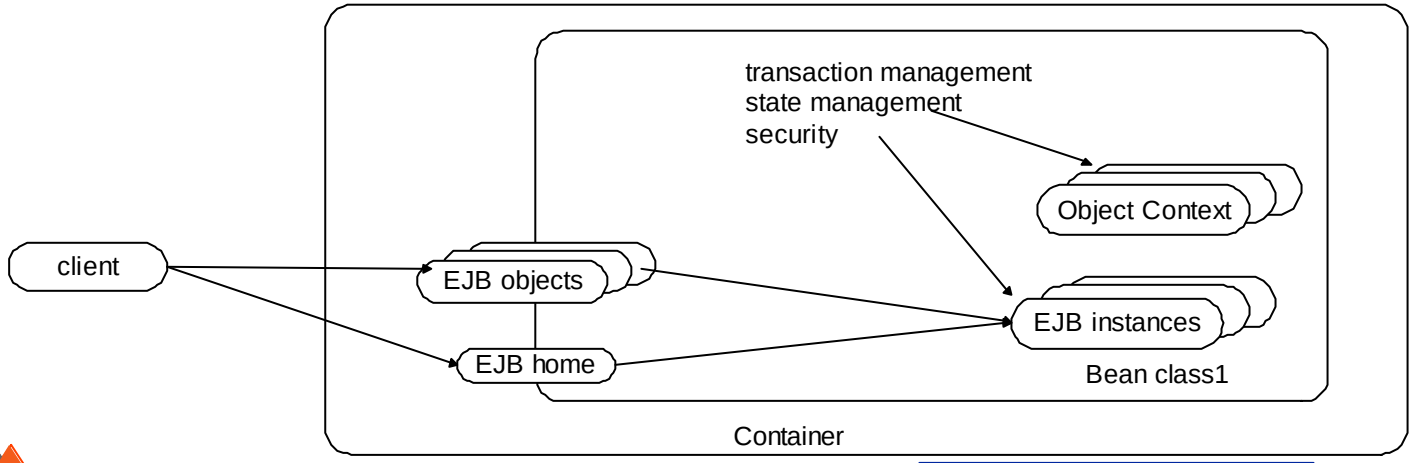
EJB Provider Responsibilities

- EJB Class: implementation of business methods and creation
- Home Interface: create, remove and lookup an EJB object
- Remote Interface: business methods callable by a client
- Deployment Descriptor: declarative attributes instructing the container how to manage EJB objects
- Environment Properties: if dependents on special ones



Container's Responsibilities

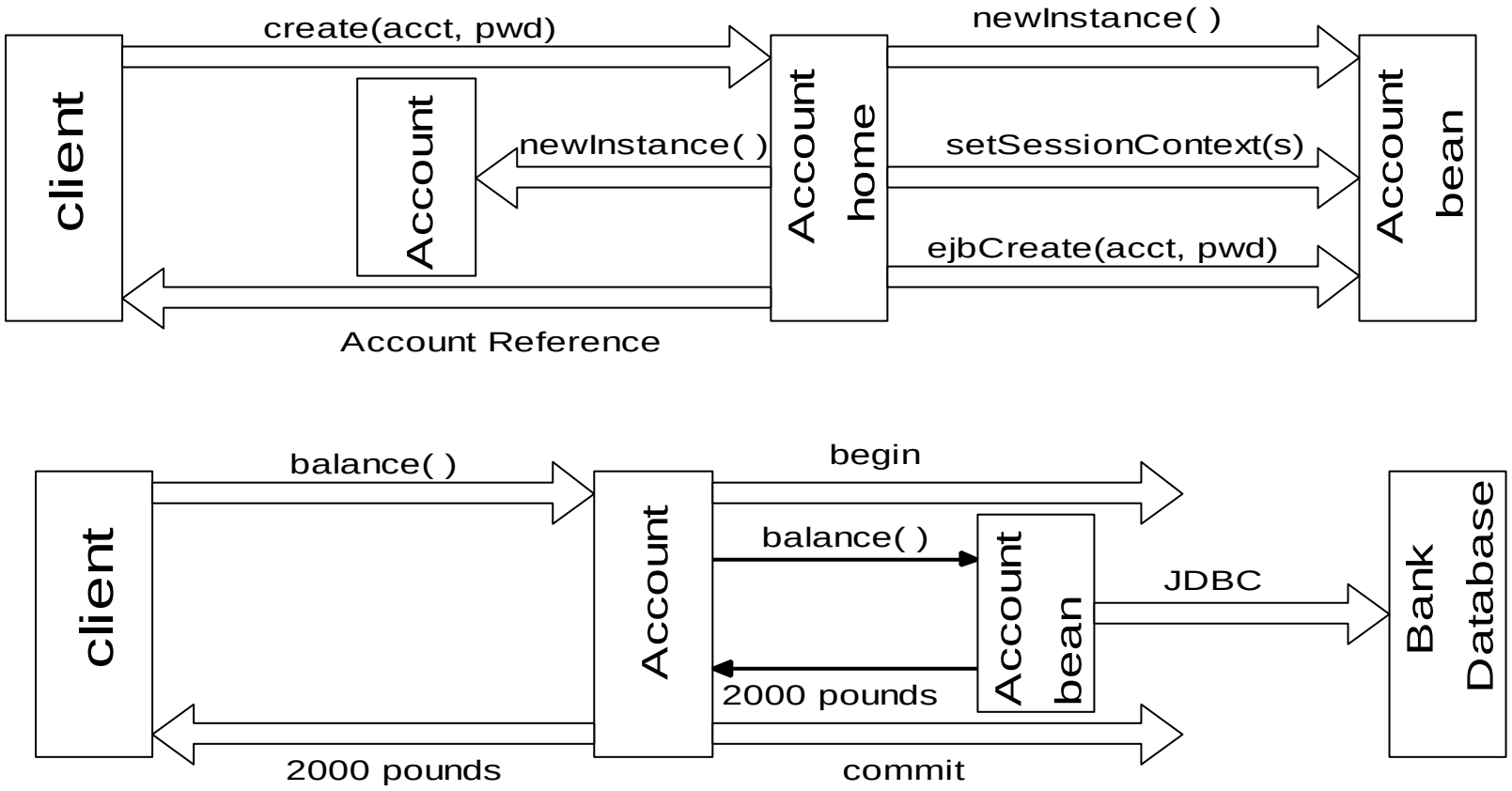
- Implement EJB Home interface
- Implement EJB Remote interface
- Ensure transaction, state and security rules
- Provide SessionContext/EntityContext for a bean to obtain various information and services from its container
- Ensure high availability and performance



Session and Entity Beans

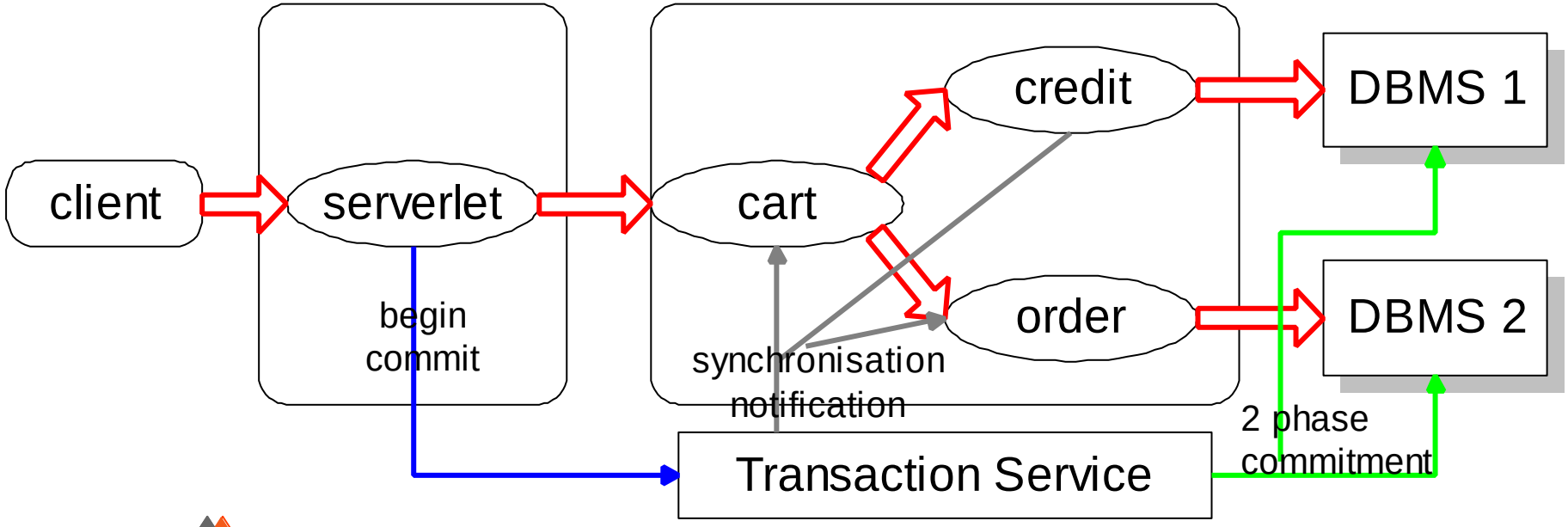
- Session (conversation)
  - not shared
  - persistent object reference
  - can be transaction-aware
  - transient state
  - not survive crashes
  - example: shopping cart
  - mandatory at 1.0
- Entity (persistent object)
  - shared
  - persistent object reference
  - transactional
  - persistent state
  - service crashes
  - example: Account
  - optional at 1.0
  - mandatory at 2.0

Creation and Method Invocation



Transaction Management

- Transaction can be demarcated by: client, container, or beans
- Beans need not do anything to ensure transaction semantics
- EJB server and database systems perform a 2PC protocol
- EJB server notify beans about a transaction completion

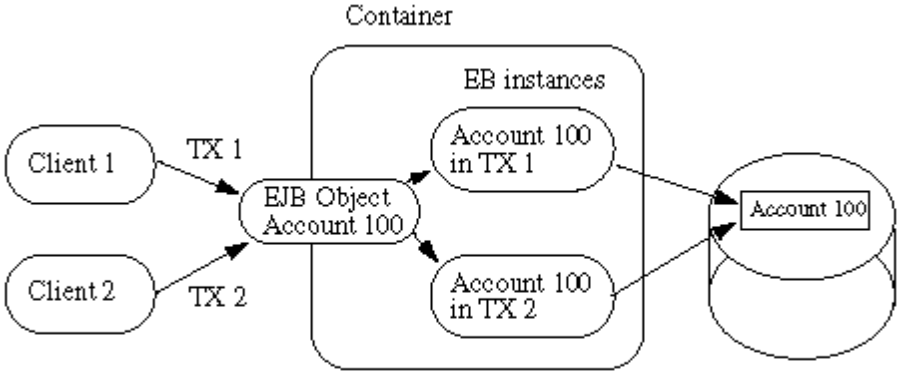


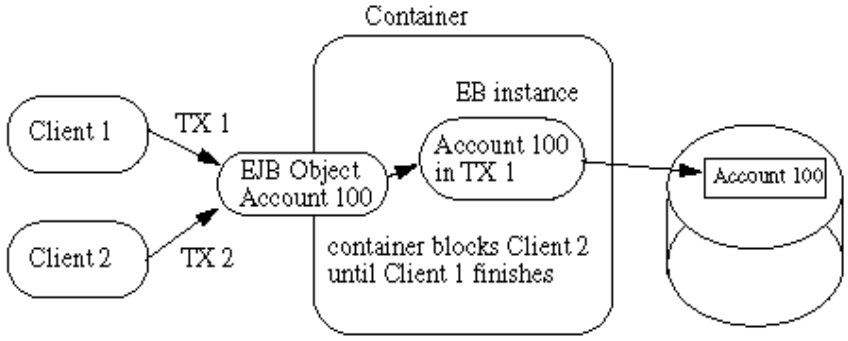
Declarative Transaction Attributes

- Bean can specify its transaction attribute in its deployment descriptor
- A container manages a bean according to the attribute
  - TX_NOT_SUPPORTED
  - TX_BEAN_MANAGED
  - TX_REQUIRED
  - TX_SUPPORTS
  - TX_REQUIRES_NEW
  - TX_MANDATORY

Concurrency Access Control

- Database managed
  - activate an instance for each transaction
  - synchronisation done by underlying database system
- Container managed
  - activate a single instance
  - container serialises access from multiple transactions
- Either is ideal

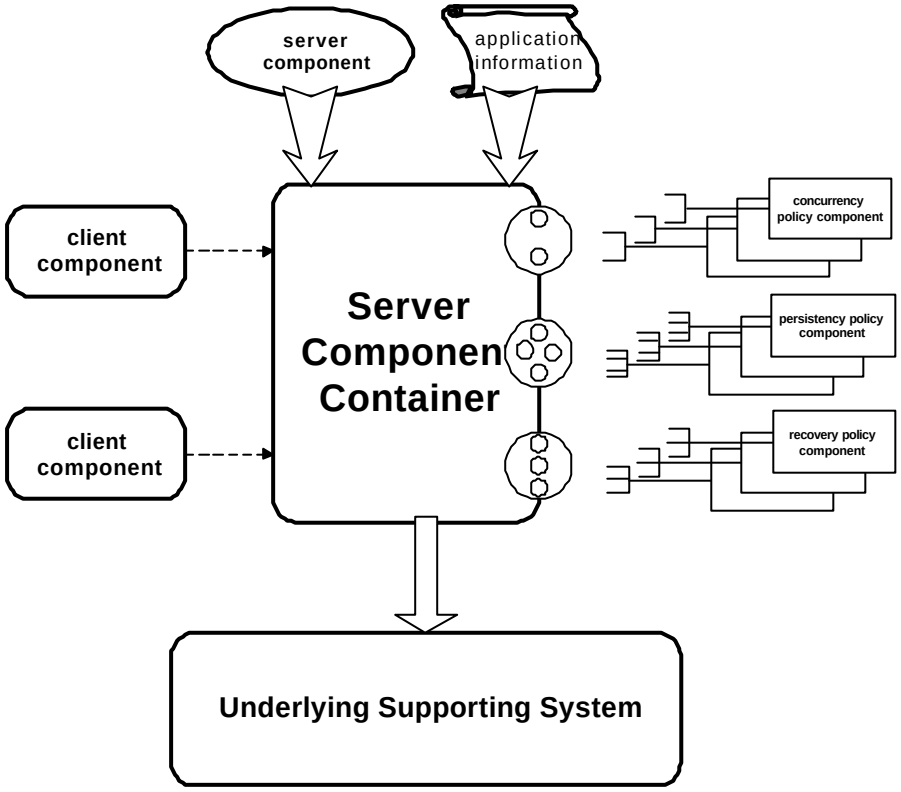




Reflective Transaction Architecture

A specialised EJB container

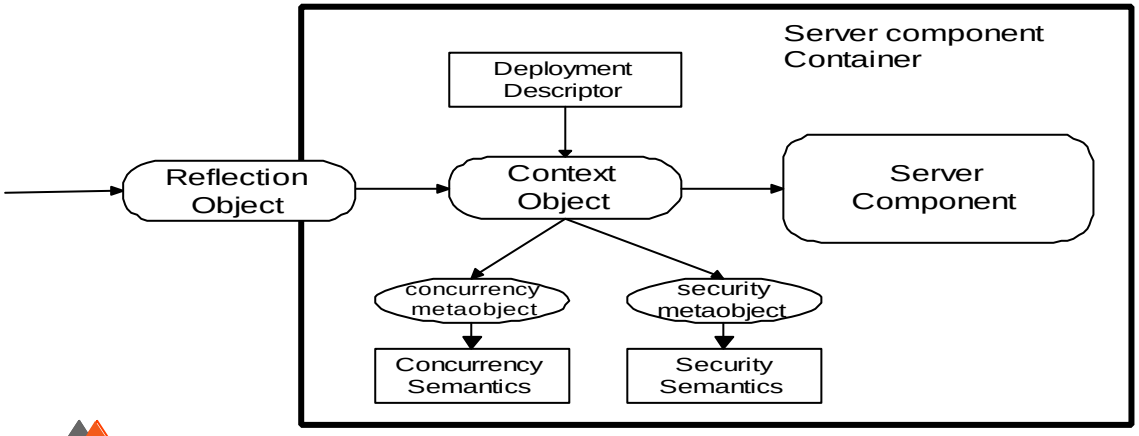
Reflective Component-Based Transaction Architecture



- Container provides system capabilities, e.g. transaction
- Implementation strategies are represented in metaobjects
- Users can choose “off-the-shelf” metaobjects that are best suited to their own applications
- Application-specific information is provided via scripts
- The information can be used for improving configuration and system performance

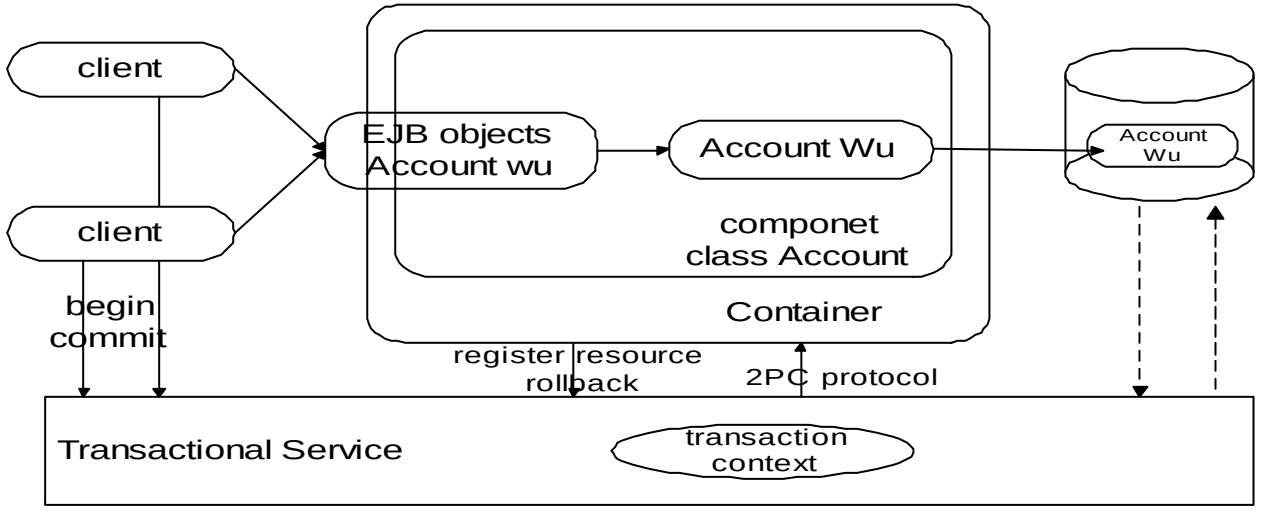
Container's Structure

- A reflection object acts as a component's external interface
- Container intercepts calls to a component via reflection object
- Container enforce transaction & security rules via interacting corresponding metaobjects at appropriate time
- Container insulates component from underlying system
- Context object maintains component's runtime information
- Component semantics are used for improving performance



Transaction Model

- Based on OMG's Object Transaction Service, but
  - not rely on supports from database management systems
  - concurrency control based on individual objects
  - in pure Java environment
- Containers are responsible for transaction management
  - transaction is transparent to components
  - object store may not be involve in transaction



Our Work *vs.* EJB

- A specialised EJB container
- Allow customising containers to cater for application requirements
- Enable using application-specific information to improve system performance and deployment
- Object-based concurrency control
- Not rely on transaction supports from database management system

Deliverables &Current Status

- A vision component build tool (beta)
- A compiler for generating reflection class (beta)
- A system component container (alpha)
- A set of concurrency control metaobjects (TPL)
- An object transaction service (75%)
- A demonstration example
- An architecture report (beta)