Transaction Framework

Progress Report & Demonstration

Zhixue Wu

21 July 1998

Goals of the Work

- High transparency to application developers
  - transaction and state management
  - distributed and multi-threading programming
- High performance
  - enable to use application-specific information
  - enable to choose best-suitable protocol
- Fast application development
  - assembling components via graphical tools
- Flexibility
  - plug & play new functionality

Approach

- Three-tier architectures
  - separate business logic from system issues
  - scalability, reliability and manageability
- Component technology
  - align with Enterprise JavaBeans
  - reusability and fast application development
  - platform and system independent
- Reflection and introspection
  - flexibility
  - transparency

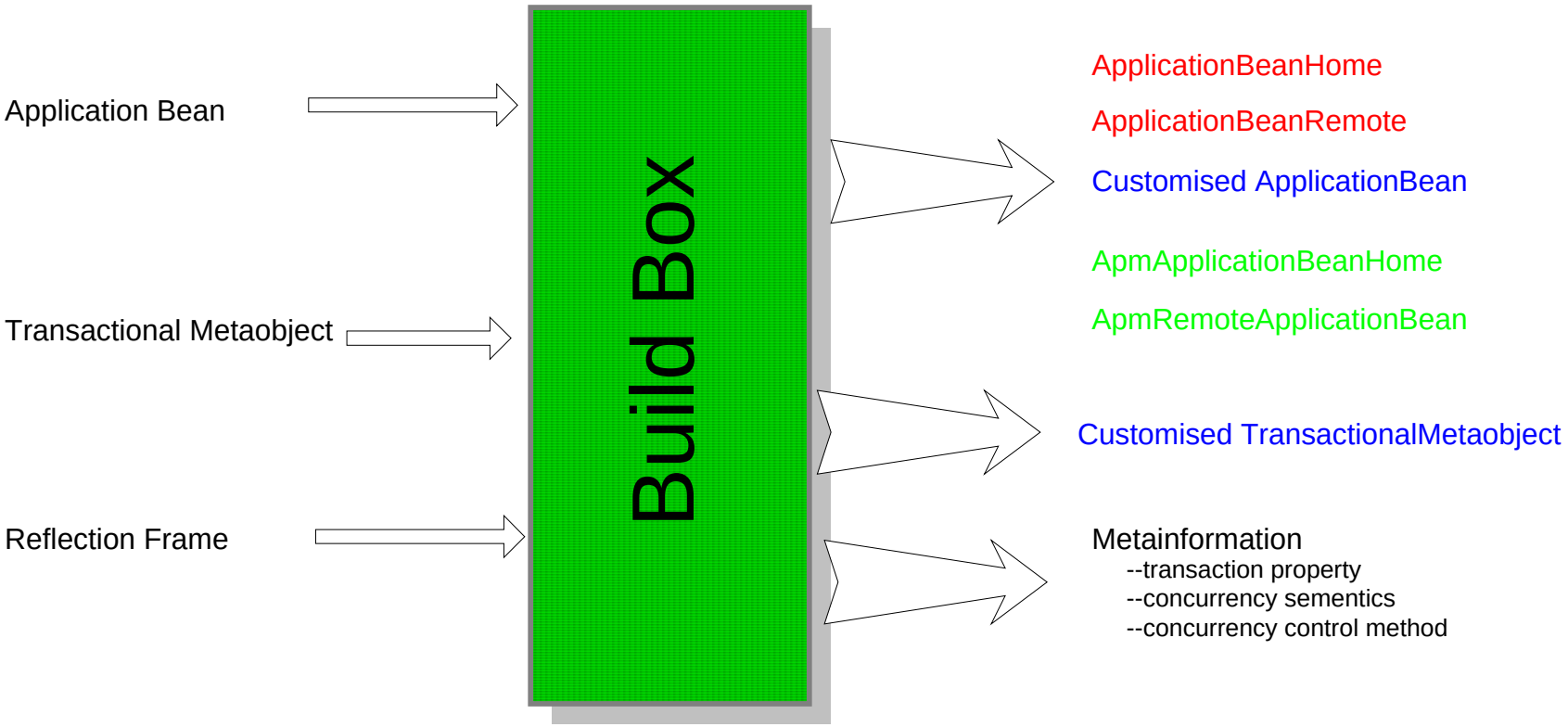
Process of Building EJB Application

- 1. Design and implement Beans
- 2. *Assembly and customisation*
- 3. Package into EJB-Jar
- 4. Deploy EJB-Jar
- 5. Test and debug
- 6. Run application

Assembly and Customisation

- Connect a component to the transaction frame
- Select operations for creating beans
- Select operations for accessing beans
- Select transaction property for external operations
  - requires, supports, not-supported, bean-managed
- Select concurrency control method
- Input concurrency semantics
- Customise component via its properties
- Set up connection with other components

Build Process

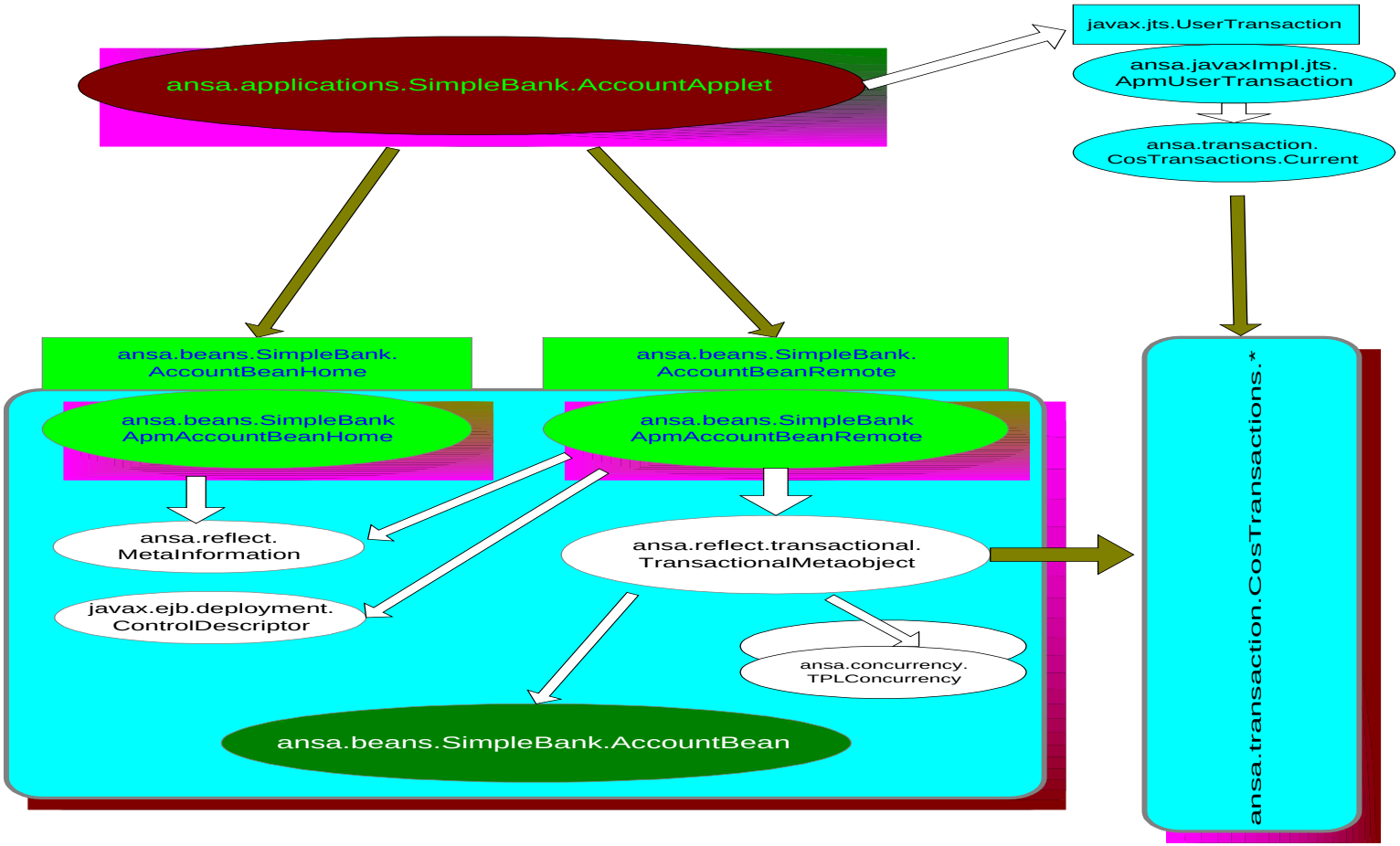


Input Concurrency Semantics

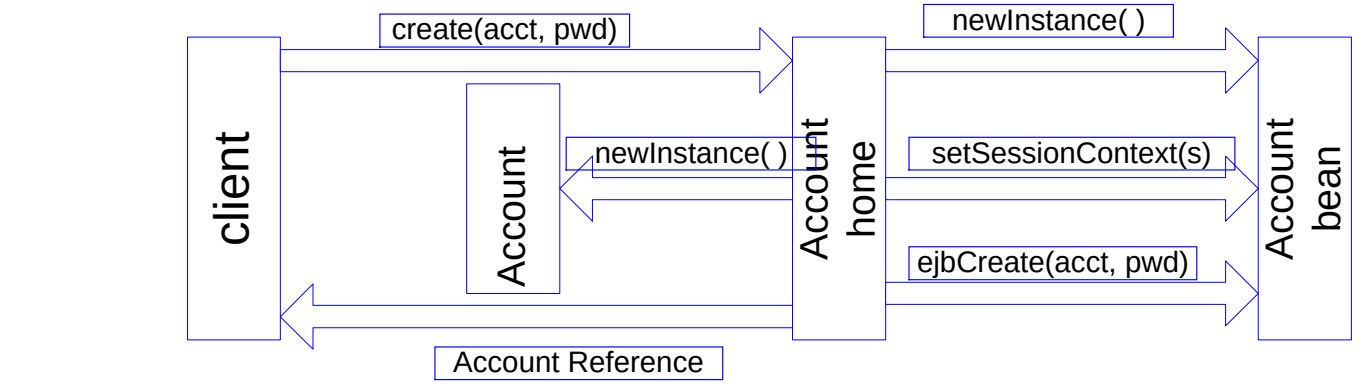
- The representation of concurrency semantics is related to individual concurrency control method
- For 2PL, operations are classified into 2 categories: *read / write*
- Normally, concurrency semantics is represented as relations between operations

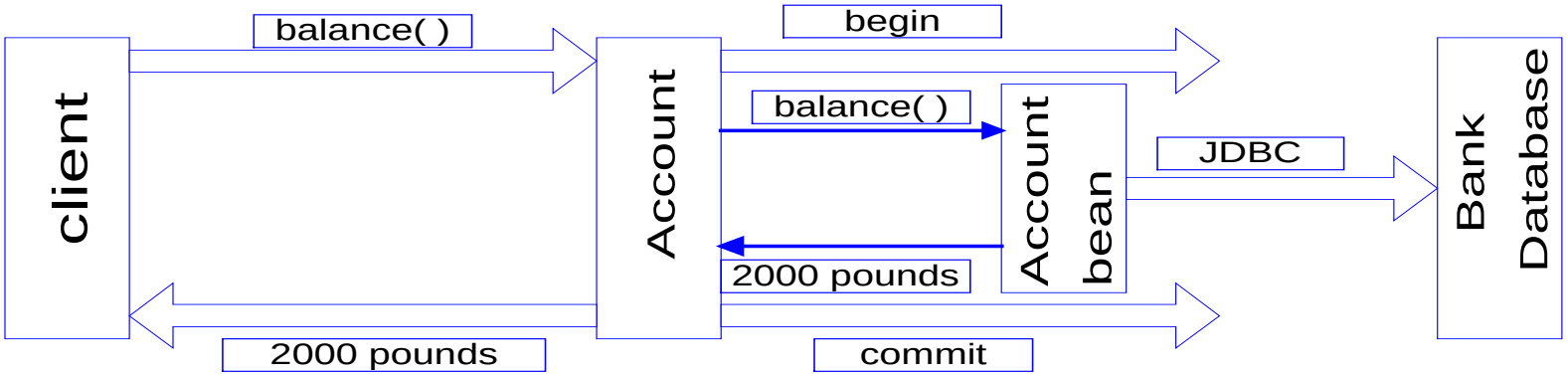
```
class Account {
        private  double  amount;
        public           Account( );
        public   void    credit(Money in);
        public   void    debit(Money out);
        public   double  balance( );
}
2PL:    read operations: balance
        write operations: credit, debit
commute operations:
        (balance, balance)
        (credit,  credit)
invalid-by relations:
(credit,  balance)        (credit,  debit)
(debit,   balance)        (debit,   debit)
```

Runtime Architecture



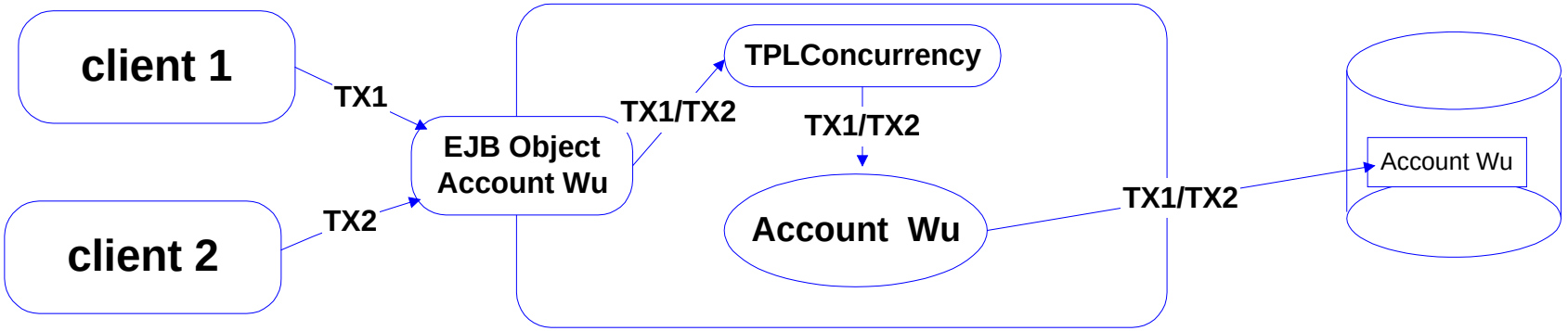
Create and Access Beans





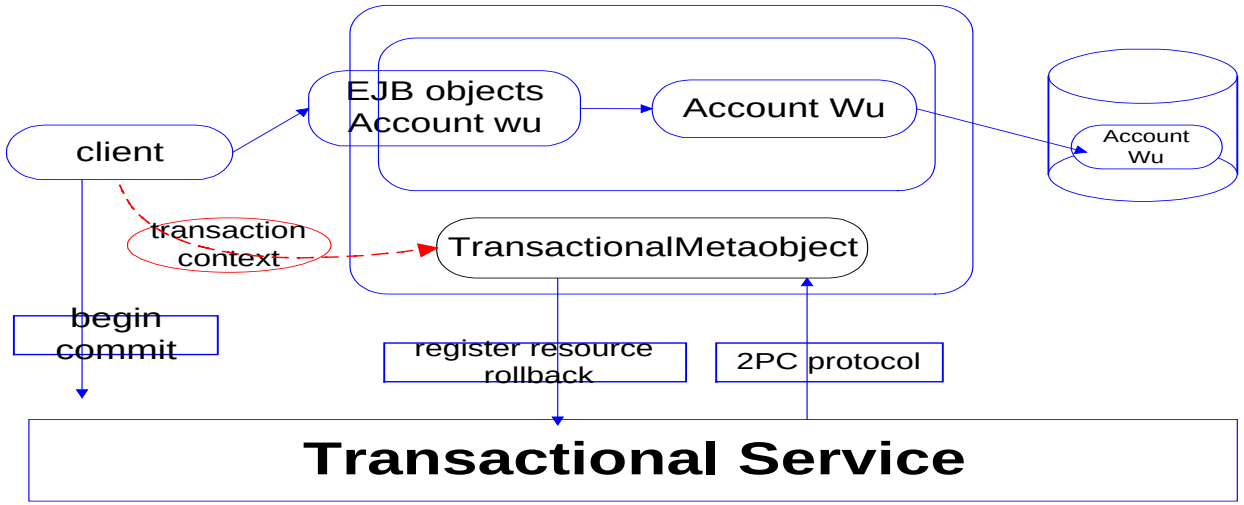
Concurrency Control

- Object-based concurrency control
- An object can be accessed concurrently by multiple transactions
- Object semantics can be used to increase concurrency

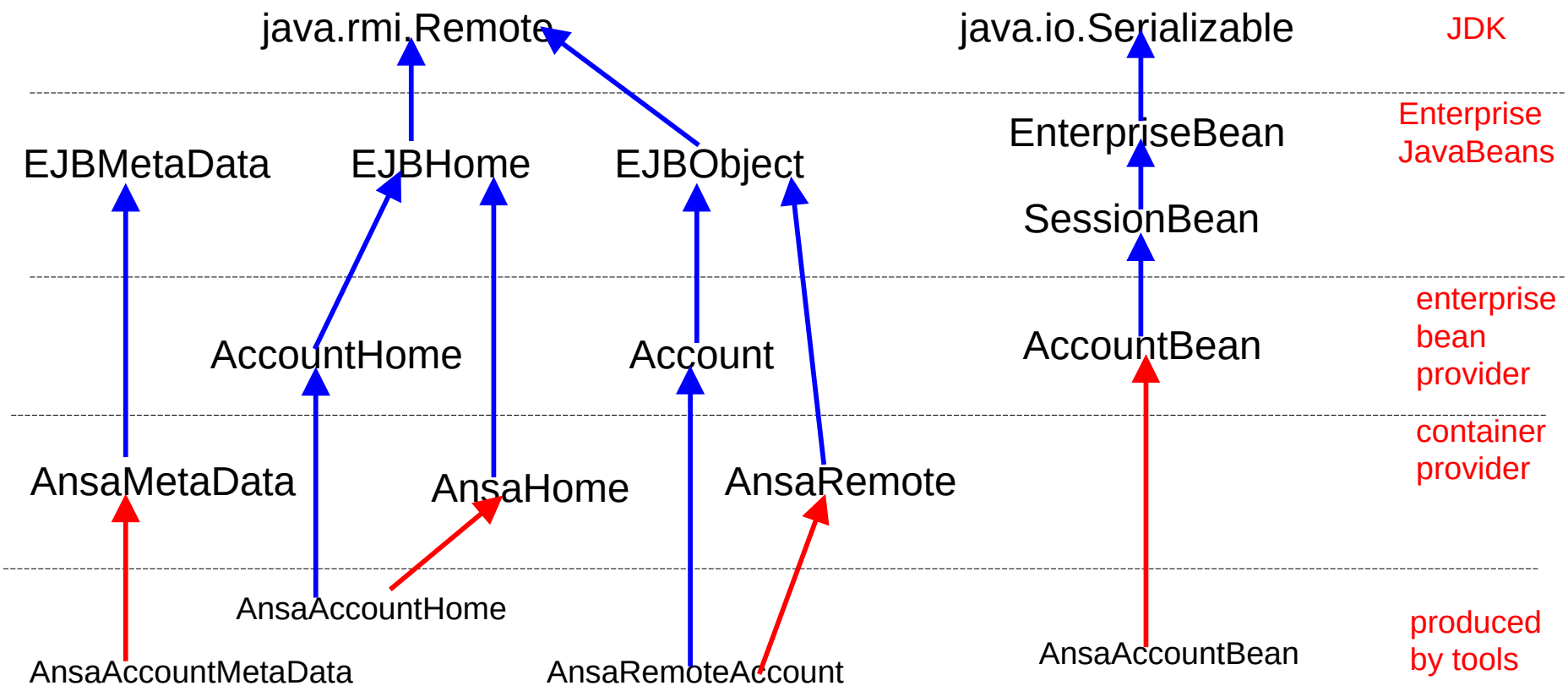


Transaction Management

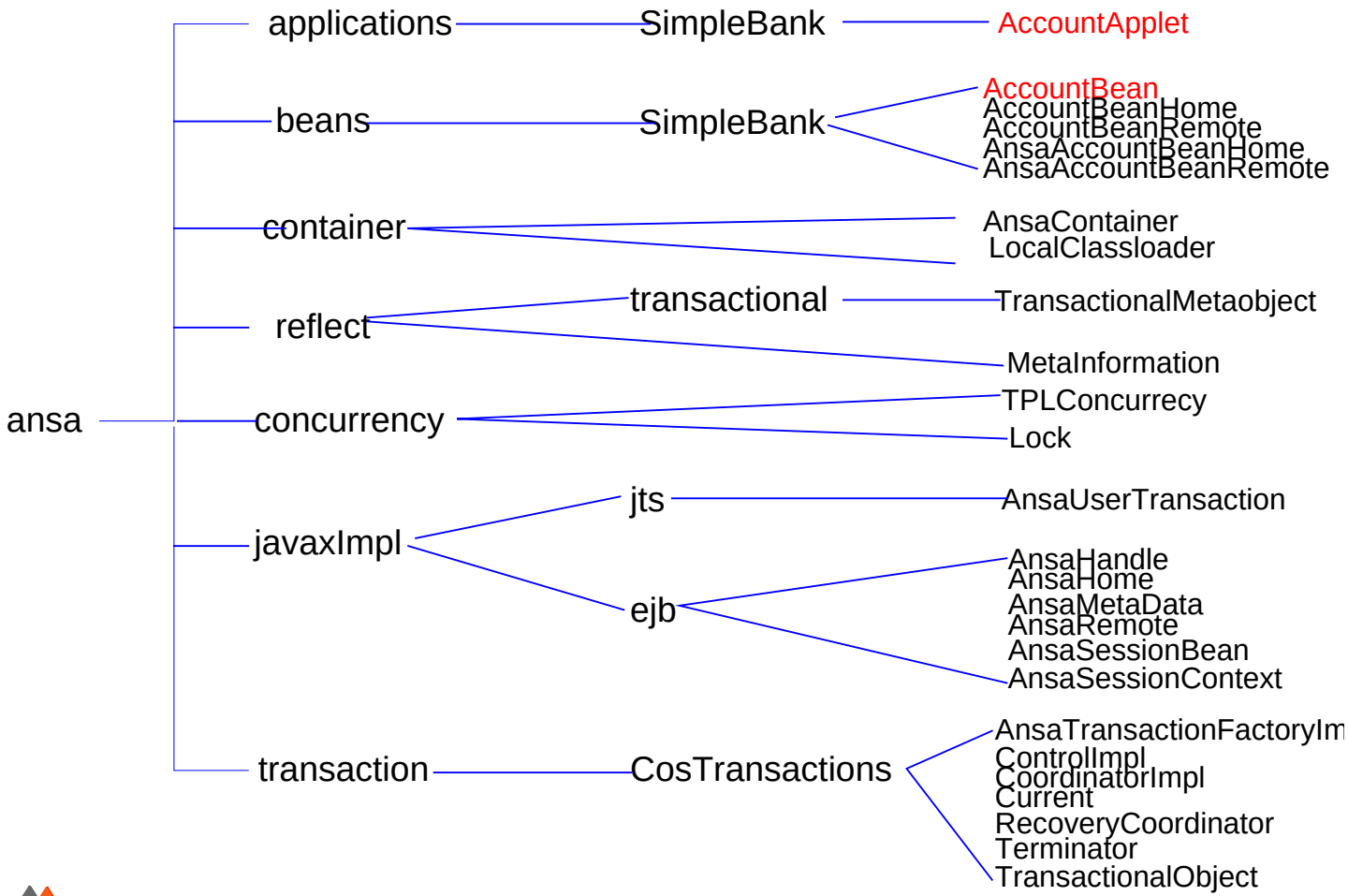
- Based on OMG's Object Transaction Service, but
  - object-based concurrency control
  - allow an object to participate in multiple transactions concurrently
  - not rely on database system's transaction functionality



Inheritance Relationship



Class Architecture



Deliverables & Current Status

- A visual component builder tool (beta)---> (1.0)
- A compiler for generating reflection class (beta)--->(1.0)
- A system component container (alpha)--->(beta)
- A set of concurrency control metaobjects (TPL)
- An object transaction service (75%)--->(beta)
- A demonstration example( )--->(alpha)
- An architecture report (beta)
- Integration with FlexiNet
- Packaging to EJB Jar
- Programming guide

DEMONSTRATION

A simple bank example