JOTM: Overview and Perspectives

Marek Prochazka
Marek.Prochazka@inrialpes.fr

Presentation Outline

JOTM overview

- What is it (interfaces, services, functionality)
- Relation to standards
- Communication protocols: RMI/(JRMP, Jeremie), RMI/IIOP(David)
- Target platforms (JOnAS, …)
- Research work status

JOTM perspectives

- Engineering
  - refactoring code base, separating single-JVM and multiple-JVM transaction manager, add-ons, recovery, etc.
- Research
  - CNT/ONT, transactions in component world, adaptable transaction manager, …)

JOTM Overview

- **JOTM = Java Open Transaction Manager**
  - JTA-compatible transaction manager
  - Functional and mature implementation (used in JOnAS)

- **JOTM functionality**
  - Transaction demarcation
  - Registration of transaction participants
  - Local and distributed transactions
    - Multiple transaction coordinators and resource managers
    - Transaction context propagation
  - Two-phase commit protocol
  - ACID properties
    - Atomicity
    - Consistency

Use by clients

- Getting the UserTransaction interface via JNDI

```
interface UserTransaction {
   void begin( );
   void commit( );
   void rollback( );
   ...
}
```

- Explicit transaction demarcation

- Involving operations upon objects (EJB) to transactions
  - Implicit transaction propagation

Use by application servers

Middleware layer

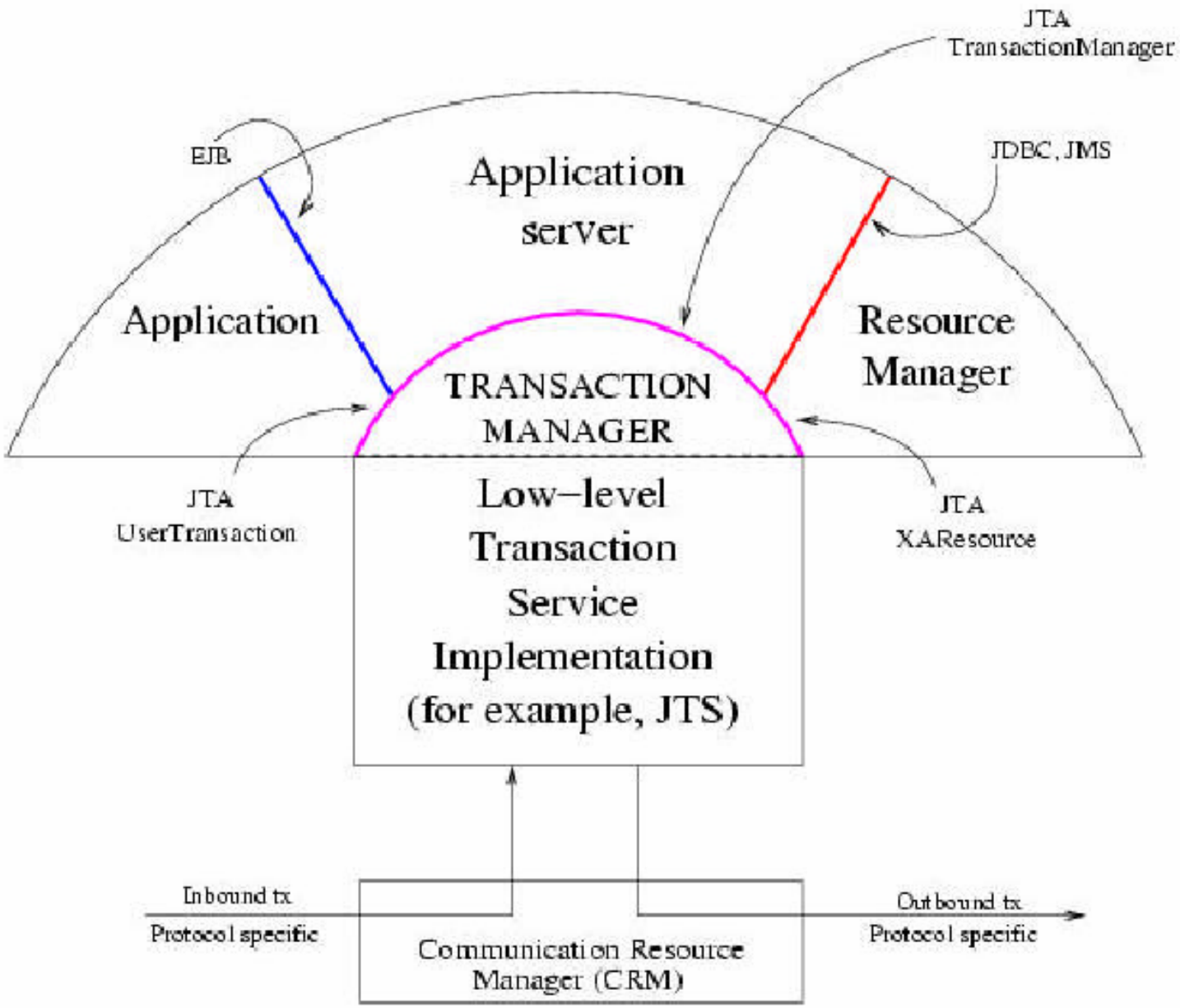
- Getting the TransactionManager interface via JNDI

```
Transaction TransactionManager.getTransaction( );

interface Transaction {
   void commit( );
   void rollback( );
   boolean enlistResource(XAResource, ...);
   void registerSynchronization(Synchronization);
   ...
}
```

- Explicit transaction demarcation
- Declarative transactions (EJB transaction attributes)
- Registration of synchronization objects and XA resources

Java Transaction API (JTA)



Current JOTM Implementation

What is and what is not implemented

- **Implemented**
  - JTA
    - All interfaces
    - Local and distributed transactions (JDBC 2.0 standard extension)
    - Timer management
  - JDBC 1.0 wrapper (JDBC 2.0 pseudo driver)
  - Communication protocols supported
    - RMI (Sun's JRMP, Jeremie)
    - RMI/IIOP (David)
- **Not implemented**
  - Recovery
    - In case of a crash, ACIDity is not assured
  - Communication
    - Preprocessing needed
    - Full CORBA RMI/IIOP support missing
  - OTS-compatibility

What we have planned and accomplished

Grow a shared expertise

- More members: Bull, FranceTelecom, LIFL, Uni. of Valenciennes, INRIA, Experlog, IMAG, Christophe Ney, Atomicos (potential partner)
- Knowledge base at http://www.objectweb.org/jotm/doc/index.html

Leverage the existing code base

- BTP proof of the concept implementation (Experlog)
- Reengineering proposal for the OTS support (LIFL)
- Advanced transactions prototypes (Valenciennes, Charles Uni.)
- Recovery support analysis (INRIA)
- Integration of CAROL (unified code base for support of RMI and RMI/IIOP)

What we have planned and accomplished (cont.)

- **Developing a more generic architecture**
  - 2 architecture meetings
  - Architecture proposals from several parties
  - Short-time and long-time strategies

Implementation of CNT/ONT

by University of Valenciennes

- CNT = (Close) Nested Transaction Model
- ONT = Open Nested Transaction Model

- JOTM code modification
  - New JOTM branch

- Status: CNT almost implemented

Bourgogne Transactions

by Charles University

- **Support for advanced transaction models**
  - Dependencies, sharing, delegation
  - Support for user-defined transaction models
  - Prototype implemented using JOTM

- **Extending component interface by transaction context propagation specification**
  - Transaction propagation policy specification using NT&CT attributes

JOTM Workplan

Engineering

- Short-time steps
- Improving functionality (communication)
- New functionality (OTS compatibility, recovery)

Research

- Work with long-time perspective
- Rebuilding JOTM
- Advanced transaction models
- JOTM for Fractal

Refactoring the code

Short-time steps

- **Simplifying the code**
- **Consolidate the javadoc documentation**
- **Separation of the single-JVM and multiple-JVM transaction manager parts**
  - Sometimes called local/distributed transactions
  - Single-JVM transactions do not deal with transaction context propagation
  - Multiple-JVM transactions are dependent on the communication protocol used
- **CAROL: Short-time solution for multi-RPC integration**
  - Pre-processing not needed
  - CosNaming resource binding
  - RMI/JRMP context propagation

Support for OTS

- **No equivalent for some OTS methods in JOTM**
  - Coordinator interface

```
register_subtran_aware(SubtransactionAwareResource)
Control create_subtransaction()
PropagationContext get_txcontext()
```

  - Current interface

```
String get_transaction_name()
Control get_control()
```

- **Differences between JTA and OTS**
  - JTA: `javax.transaction.Transaction suspend()`
  - OTS:

```
Control Current.suspend()
void resume(Control which)
```

  - Transaction and Control interfaces are different
- **OTS supports nested transactions**
  - JTA supports flat transactions only

Support for Recovery

Recovery: Crucial feature

Enabling JOTM by recovery

- Log manager
  - Logging transactional significant events
  - Logging two-phase commit states
  - Logging all data related to all transaction participants
- Recovery manager: Employing the log after a JOTM crash, during recovery
  - Reconnection to all participating resources/sub-coordinators
  - Reconnection to all XA resources (JDBC, JCA, JMS connections, …)
  - Committing or rolling back transactions depending on their state
- Enabling the JOTM code by logging in appropriate events

Recovery Challenges

Problems

- Lack of specification (JTA vs XA resources)
- Lack of implementation (JDBC 2.0/3.0 drivers)
- Lack of documentation
- No OTS support in JOTM

Divide et impera

- Local recovery
  - Single JVM
  - XA resources only
    - Persistent XA resources
    - Persistent Resource Manager
- Distributed recovery
  - OTS implementation (persistency needed!)

Engineering: Others Tasks

- **Employing Monolog**
- **Fractalization**
  - Refactoring the code before fractalization
  - Related to research workplan

Refactoring JOTM: two branches

- **Engineering – short time steps**
  - JTA implementation
  - BT prototype
  - CNT/ONT prototype
  - BTP prototype
  - OTS implementation
  - Activity Service implementation
- **Research – long time work**
  - Identification of JOTM building blocks
    - Transaction Manager
    - Lock Manager
    - Log Manager
    - Resource Manager
    - Dependency Manager
    - Communication Manager
  - Defining their interfaces
  - Implementation / refactoring JOTM

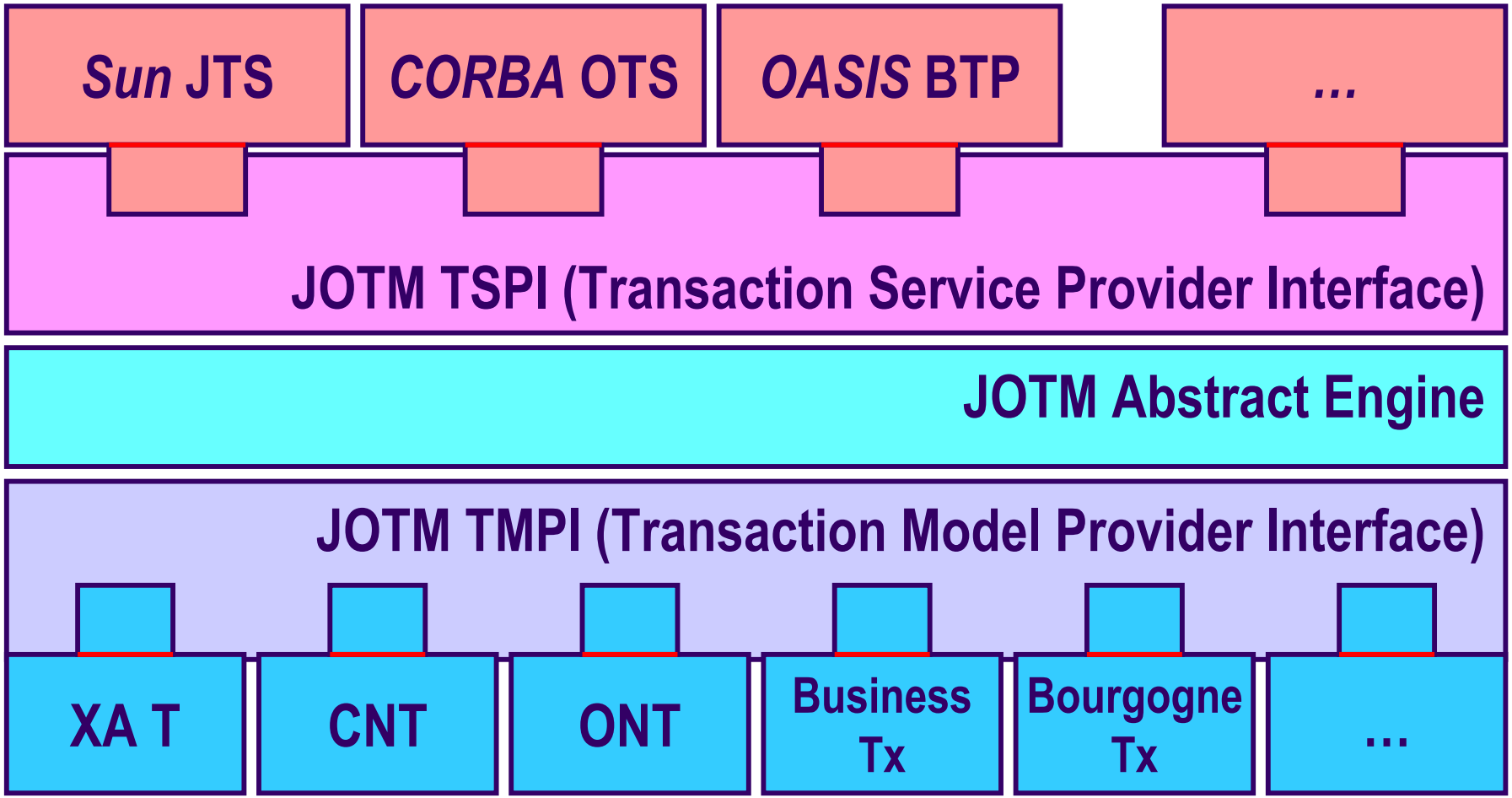
Refactoring JOTM

- **Requirement of common abstractions**
  - The notion of Transaction
  - The notion of Resource or Participant
    - Recoverable, compensable, ...
  - Transaction identification
    - Id implying inter-transaction dependencies
  - Transaction demarcation
    - Significant events, commit protocols, …
  - The notion of Coordinator or Terminator
  - Delegation
  - Dependencies

- **Transaction Manager: Engine based on abstractions**

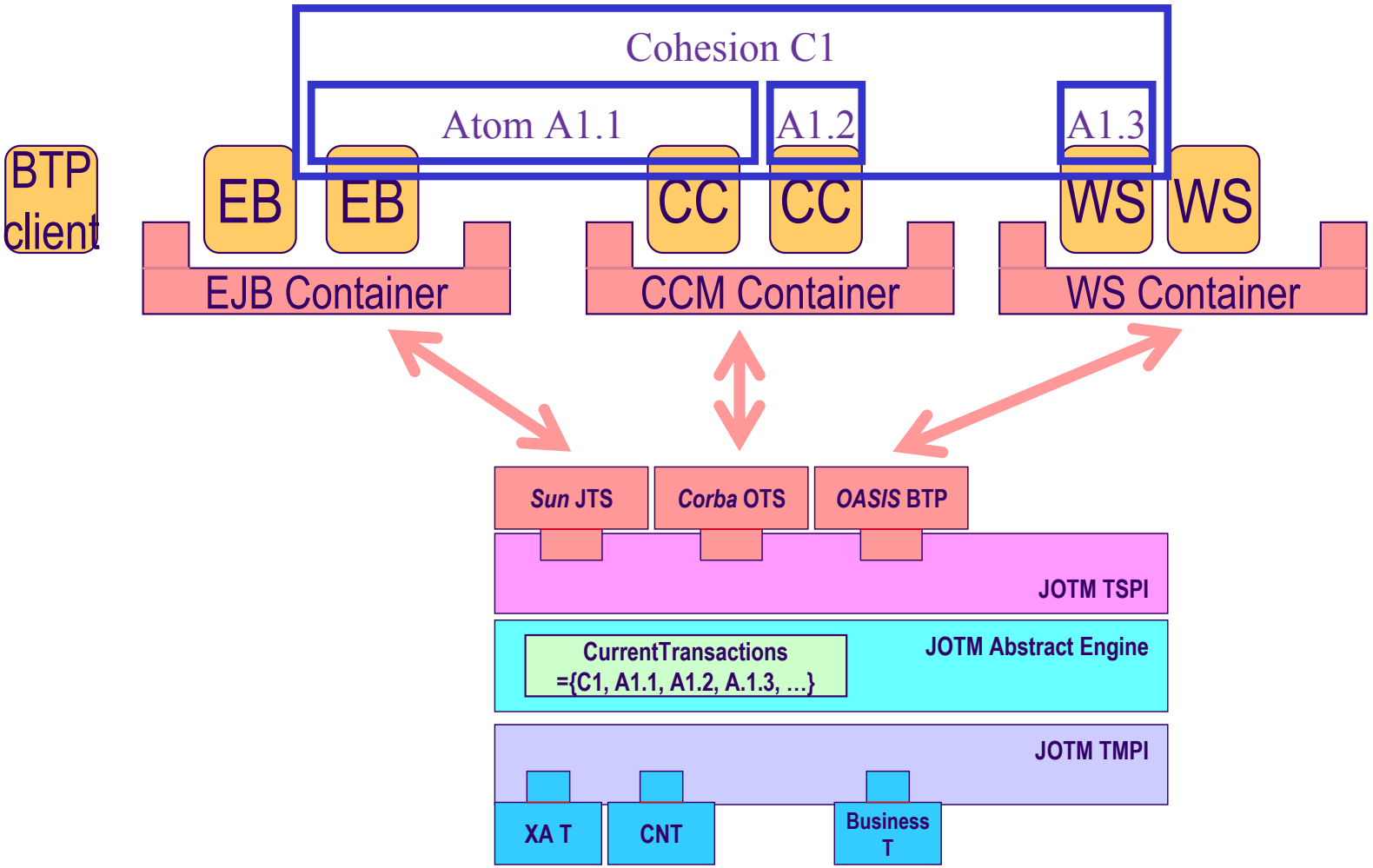
Selecting JOTM Model

Proposal by University of Valenciennes



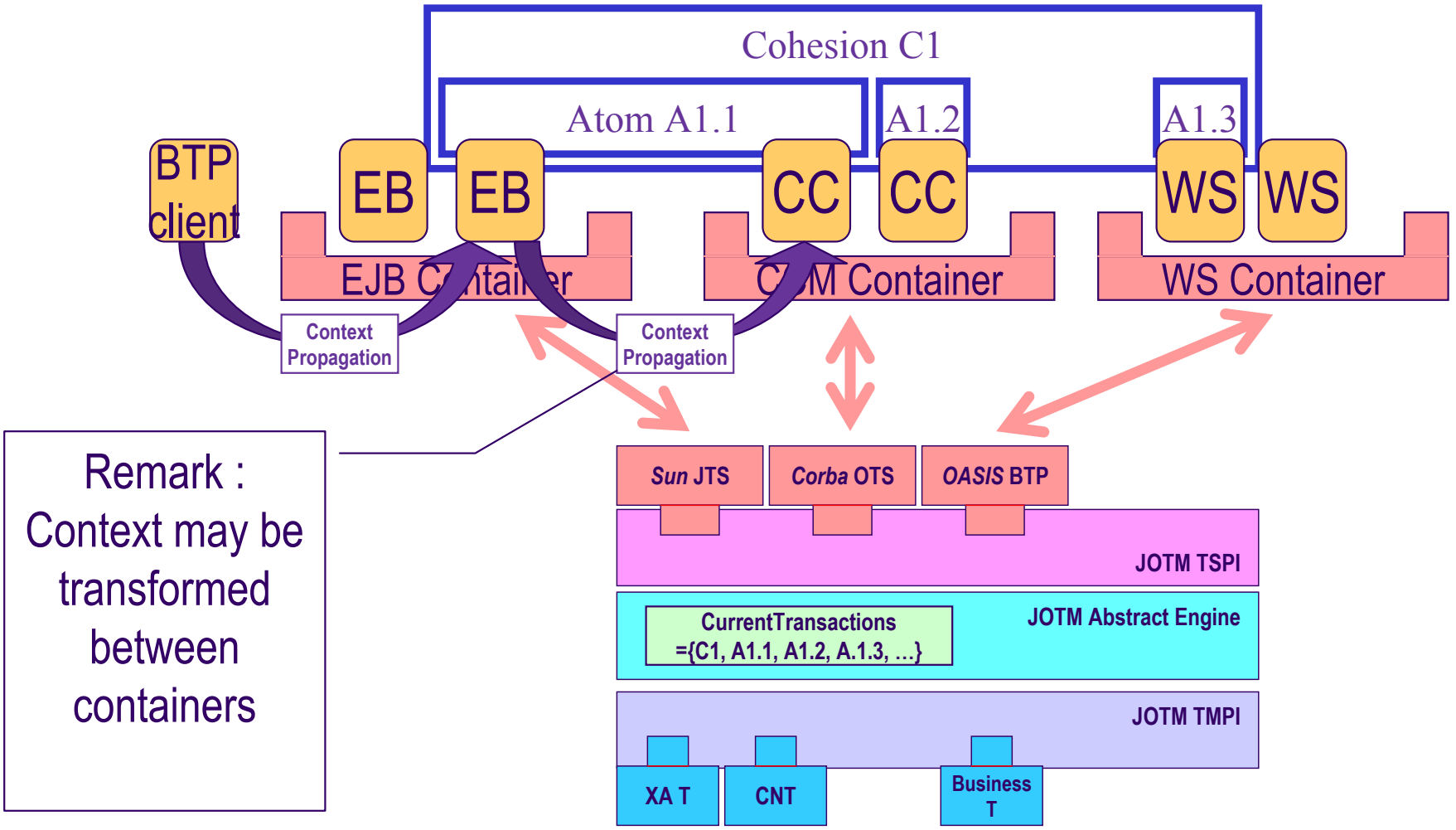
JOTM and Containers

Scope of transactions



JOTM and Containers

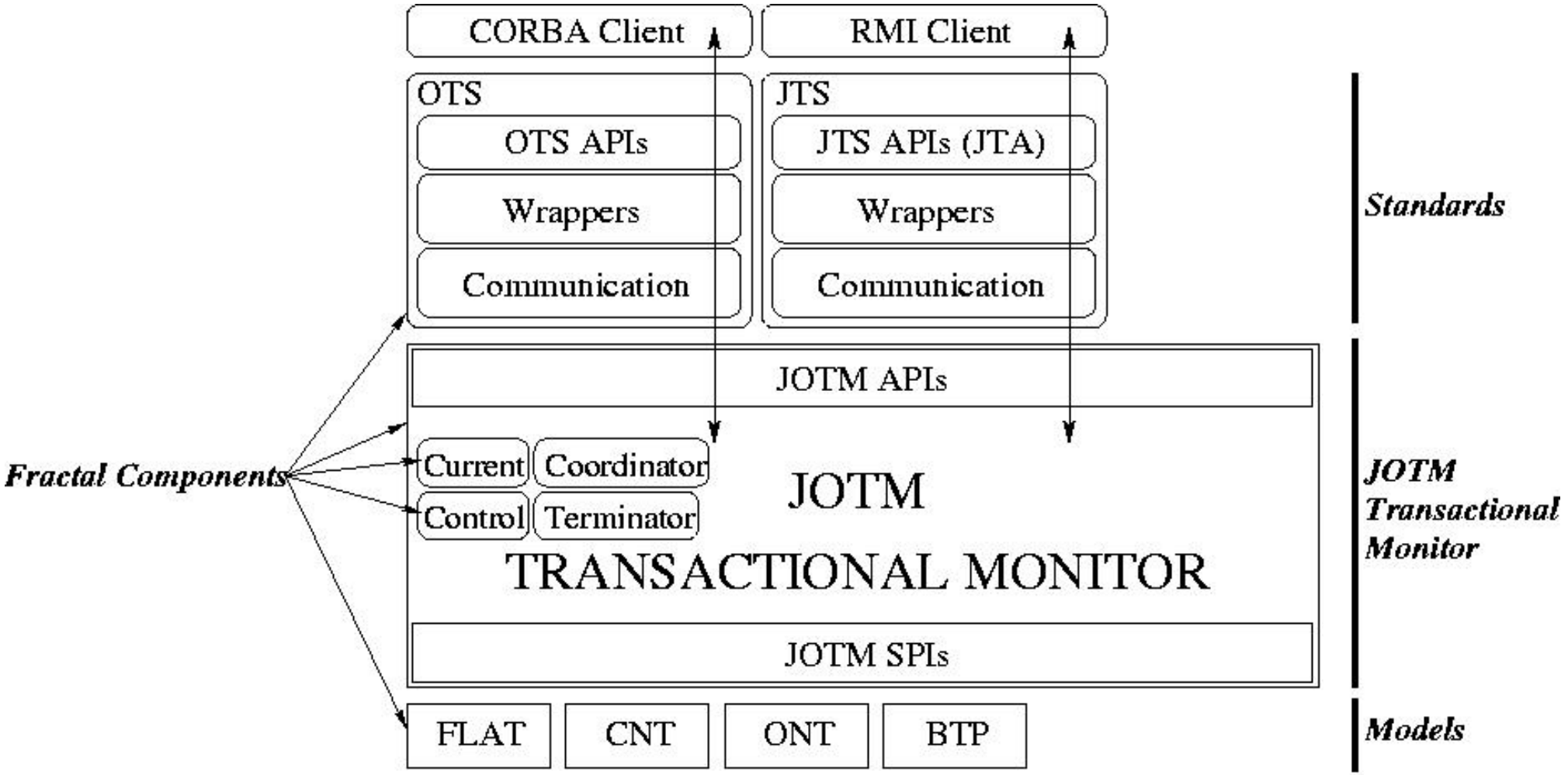
Context propagation through several containers



Remark : Context may be transformed between containers

Selecting JOTM Model

Proposal by LIFL



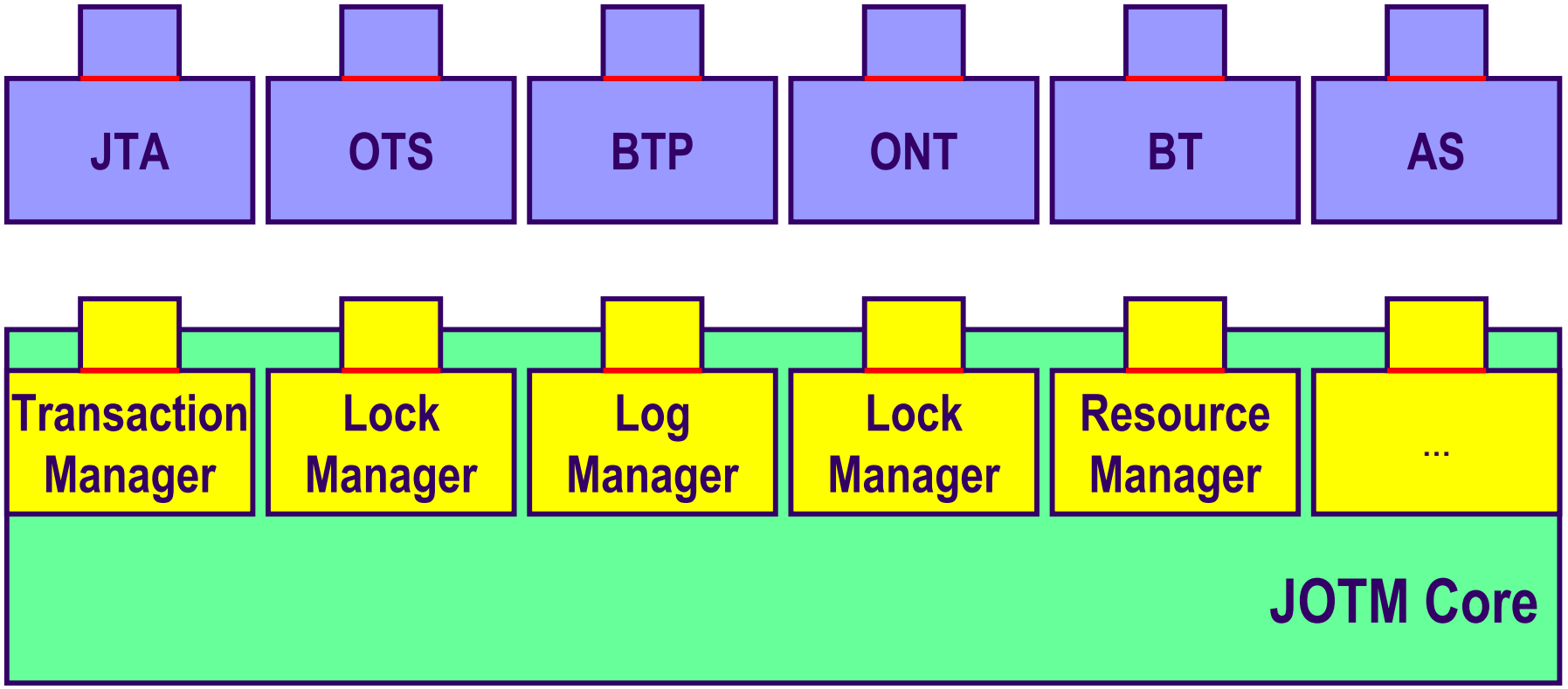
Selecting JOTM Model

Proposal by INRIA

- **Experiencing the work done by engineering branch**
  - Implementations of specifications
    - JTA, OTS, BTP, …
  - Advanced transaction models prototypes
    - Bourgogne Transactions, CNT/ONT, …
  - Communication
    - CAROL, …
- **Analyzing requirements/limitations**
- **Proposal of key building blocks and their interfaces**
- **Developing the JOTM core**
- **Developing add-ons and personalities**

JOTM Building Blocks

Proposal by INRIA



JOTM Building Blocks

Examples

- **Transaction Manager**
  - begin, commit, abort
  - JTA, OTS: two-phase commit protocol
  - BTP: different commit protocol
- **Lock Manager**
  - Lock, unlock, define_conflict_table, relax_conflict
  - None of JTA, OTS, BTP, … takes care of locking
- **Resource Manager**
  - Register_resource, delegate_resource, …
- **Log Manager / Recovery Manager**
  - Read_record, write_record
  - What is “log record” (open-nested transactions: storing compensating actions)

Adaptable Transactional Services

- **Basic interfaces of building blocks**
  - Basic functionality (flat transactions)
  - Implementation-independent (e.g., registering generic “resources”)
- **Meta-level interface allowing modify basic interface semantics**
  - Modifying semantics of operations (e.g., commit protocol for ONT, BTP)
- **Emphasis on recovery aspects**
  - Usually missing in related work
  - Log Manager / Recovery Manager interface
  - Support for advanced transaction models

Transactions in Components

- **Continuing the work on Bourgogne Transactions**
  - Transaction context propagation as a part of the component interface
  - Multiple interfaces: Multiple units of concurrency control

- **How transactions should be treated in Fractal-like component world?**
  - What is transaction context
  - How it should be propagated
  - What policies for propagation of a transaction to a component
  - How and where to specify such policies (EJB attributes)
  - Concurrency control policies specification
    (EJB: every component is always locked)