hochschule für angewandte wissenschaften
FACHBEREICH ELEKTROTECHNIK UND INFORMATIK hamburg
university of applied sciences

Long Running Transactions in Service-Oriented Environments

infm3 :: SR

Martin Gerlach

martin.gerlach@informatik.haw-hamburg.de

16.12.2005

Need for consistency of business critical data

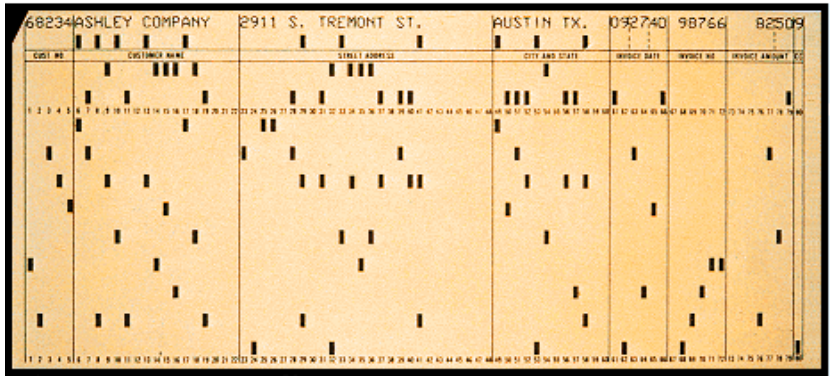
HAW HAMBURG

- … as always, across evolving technologies
- Combination of coordination/composition techniques (workflow) [since early 90ies] with loosely coupled services [this decade]
- Distributed applications [since "sometime long ago"]
- Mobile applications [more and more, recently]
- EAI + B2B
  - Services/SOA/<insert more buzzwords here> fit well
  - Nothing groundbreakingly new, but successful application in numerous projects
  - Many lessons learnt

Noteworthy historical facts (1)

- 6000 years ago, Sumer: Royal inventory of taxes, land, grain, cattle, etc. on clay tablets
- Records kept for every transaction
- … papyrus, …, paper …
- 1890 use of punch card system to report US census by Herman Hollerith
- From ≈ 1950
  - Batch (offline) transactions
  - Followed by online transactions (OLTP)

From Computer Desktop Encyclopedia
© 2000 The Computer Language Co. Inc.



Noteworthy historical facts (2)

Jim Gray, 1981:
The Transaction Concept – Virtues and Limitations

- Purpose: There are no "perfect systems", so we need to make "almost perfect" systems safe(r), i.e. fault tolerant
- ACID Transactions: Activities composed of actions of different criticality
- Realization: Update-in-place, Time Domain Addressing, Logging and Locking, UNDO/REDO, etc.
  - First undo/redo log: Hänsel and Gretel ☺ … also first log failure
- Limitations
  - Nested Transactions
  - ***Long Lived Transactions***
  - Transparent integration into programming languages
- Peter Principle: "Every idea is generalized to its level of inapplicability"

Thesis outline :: Where are we today?

HAW HAMBURG

Thesis

- Working title **"Long Running Transactions in Service-Oriented Environments"**
- Outline
  - SOA: From objects to components to services: Why services? — SR
  - Distributed transactions in component frameworks
  - Why long running transactions? — AW1
  - Existing specs and ongoing research work
  - Conceptual design of a framework that supports coordination and long running transactions — SR/PJ
  - Breaking down the (somewhat overloaded) Web Services related specs and the design to the essential concepts
    - Aim at performance
    - Aim at limited resources (mobile devices)
  - Prototyping (Web Services) — PJ

SOA: Why services?

HAW HAMBURG

- What is needed?
  - Coarse granulation
  - Loose coupling
  - Asynchronous (and reliable) messaging
  - Abstraction from underlying transport and implementation
  - Standards
- Can previously existing architectures provide this?
  - Function libraries and packages
  - Classes / Objects
  - Components

SOA: Why services? :: Classes and Components

HAW HAMBURG

Classes	**Components**
Fine granulated	Coarse granulated
Described by interface	Described by interface, deployment descriptor, contract
Object oriented	Need not be OO
Stateful (instance vars)	"No" persistent state
Tight coupling	Loose coupling
Stand alone	***C³*** - "Component – Contract – Container" runtime environment
Problem domain specific *libraries*, few well known	*Frameworks*, separate from business logic, some well known
MFC, STL, …	CORBA, EJB, JavaBeans, .NET, COM

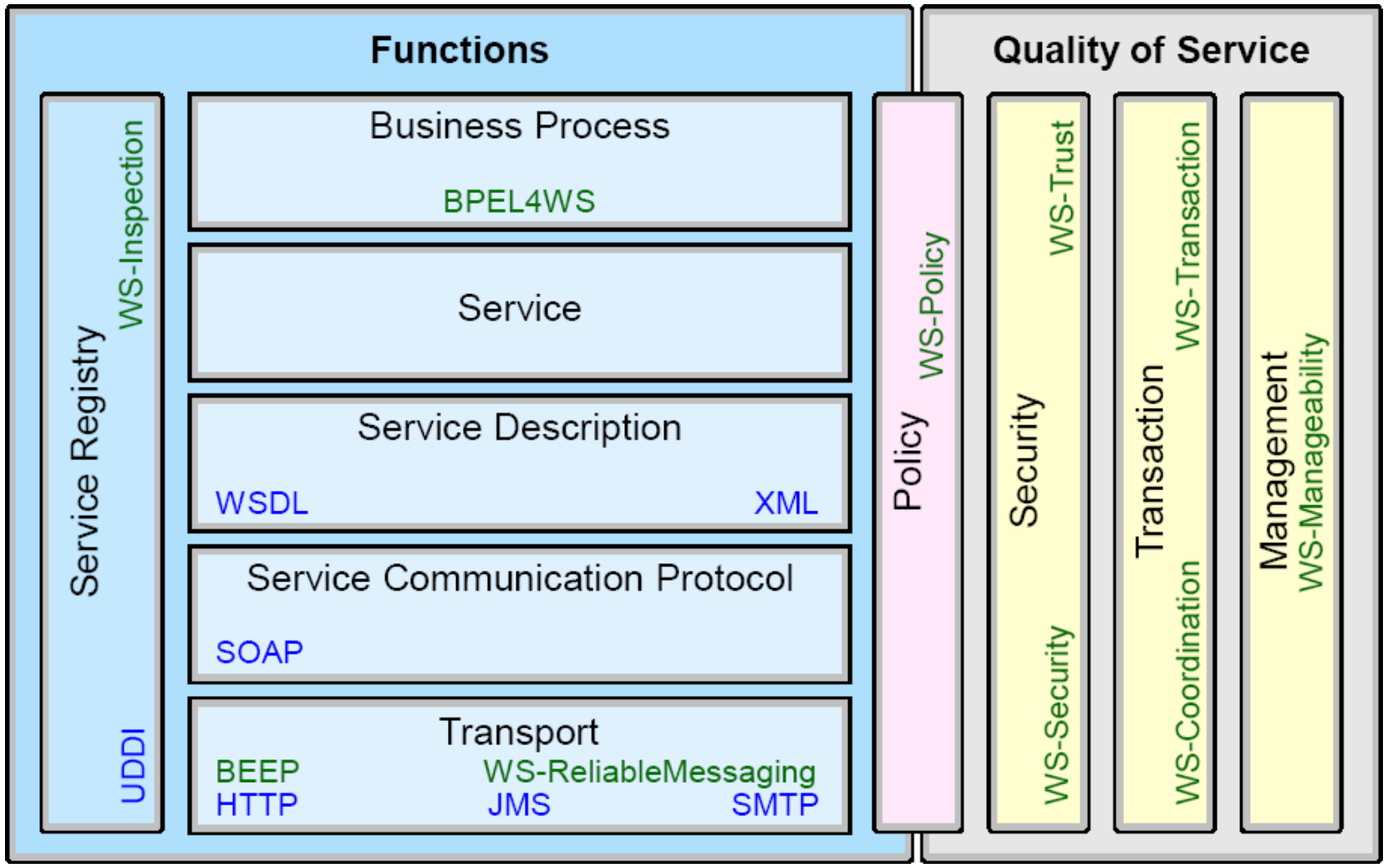
SOA: Why services? :: Conclusions

HAW HAMBURG

- Mapping of Services to Components
  - Interface
  - QoS requirements
  - Semantics
  - "Live" in managed environment (≈Container)
- Services additionally provide
  - Abstraction from implementation
  - Abstraction from transport and encoding
- Interoperability through standards
  - At least a good idea

➔ Web Services architecture specifications provide a framework for realization of SOAs

SOA: Web Services Framework



Thesis outline :: Where are we today?

HAW HAMBURG

Thesis

- Working title **"Long Running Transactions in Service-Oriented Environments"**
- Outline
  - SOA: From objects to components to services: Why services?
  - Distributed transactions in component frameworks

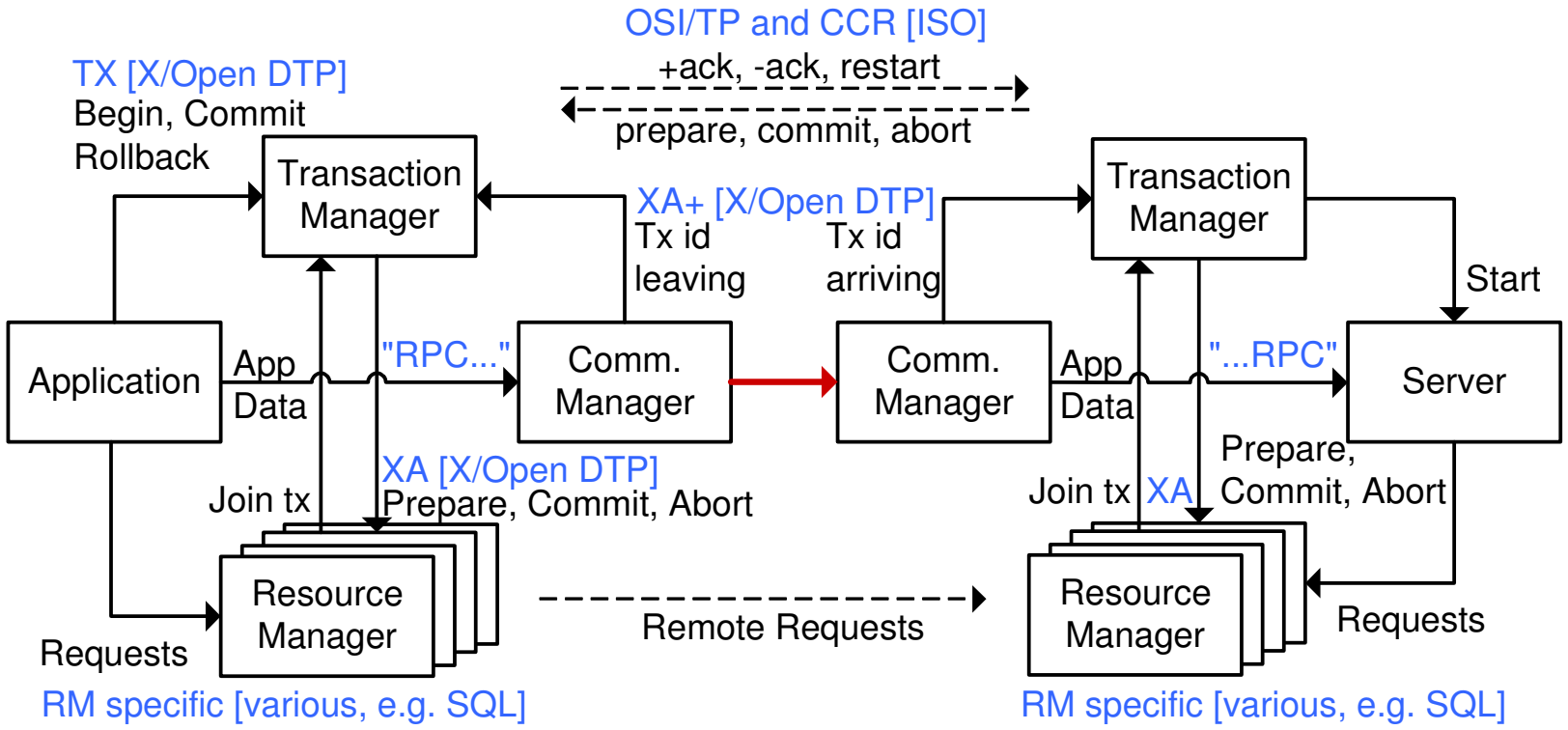
  - Why long running transactions? — AW1
  - Existing specs and ongoing research work
  - Conceptual design of a framework that supports coordination and long running transactions — **SR**/PJ
  - Breaking down the (somewhat overloaded) Web Services related specs and the design to the essential concepts
    - Aim at performance
    - Aim at limited resources (mobile devices)
  - Prototyping (Web Services) — PJ

TX management fundamentals

HAW HAMBURG

- **Transaction Processing System**
  - A whole application
    - Distributed (logically, physically, geographically)
    - Heterogeneous
    - With stringent QoS requirements
- **Transaction (TX)**
  - A collection of operations on the physical and abstract application state
  - Specifies failure semantics for computation through ACID properties
- **Transaction Processing Monitor (TP Monitor)**
  - A collection of core services that manage and coordinate transactions
  - TX manager, "RPC" manager, [Logging,] [Locking,] …
- **Resource Manager**
  - Data, code, processes providing access to shared data
  - Provides ACID operations

TX management fundamentals :: X/Open DTP



TX management fundamentals

HAW HAMBURG

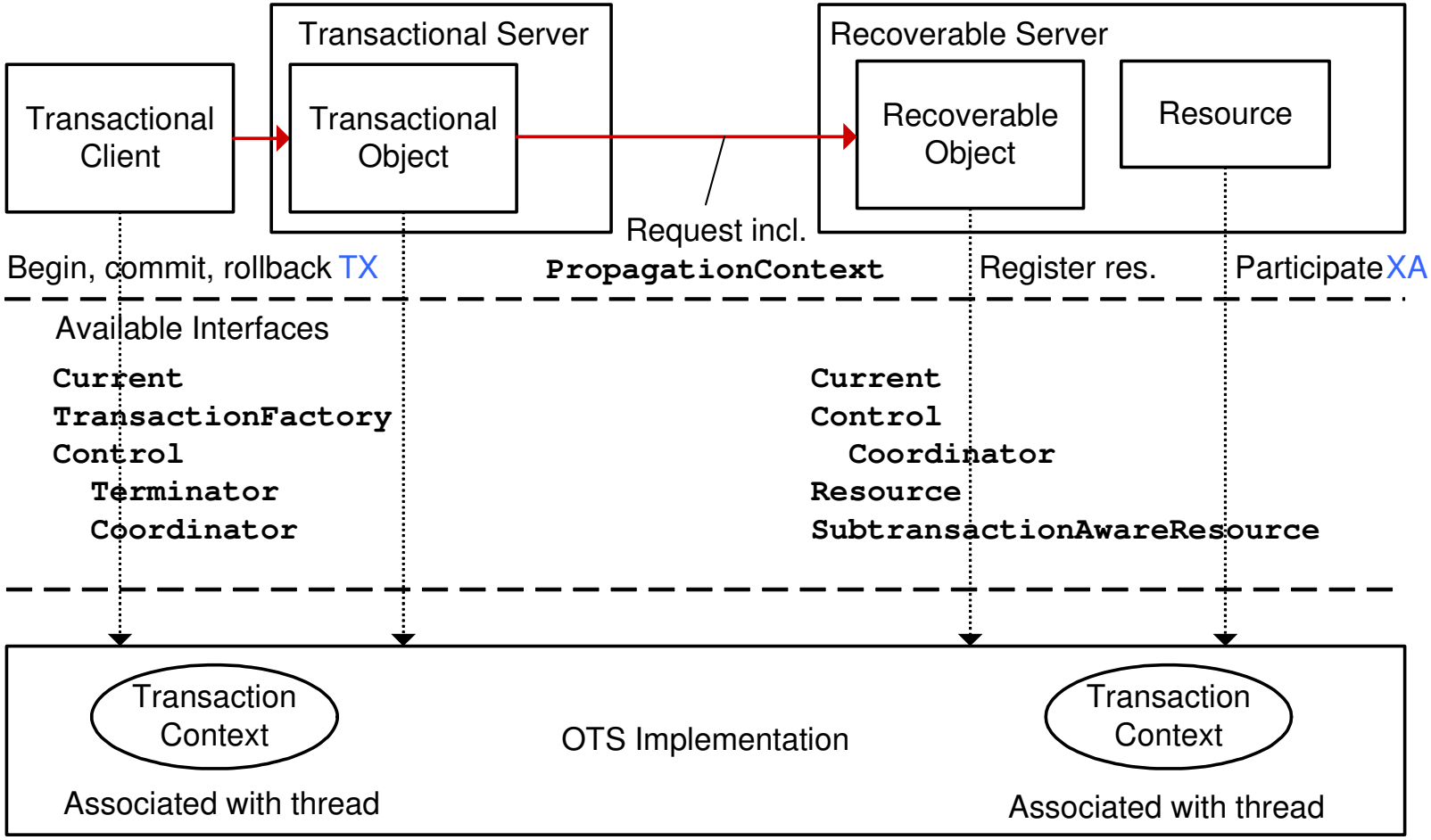
- TX models (AW1 recap)
  - Flat
  - Flat, distributed (splits into TX on each participating node)
  - Flat with savepoints (partial rollback)
  - Chained
  - Nested
  - Multilevel
  - Open nested
  - Long lived with compensation and context management
- TX processing models
  - Direct, synchronous
  - Queued, asynchronous
  - Compensation-based, both, using extra middleware

CORBA Object Transaction Service (OTS)

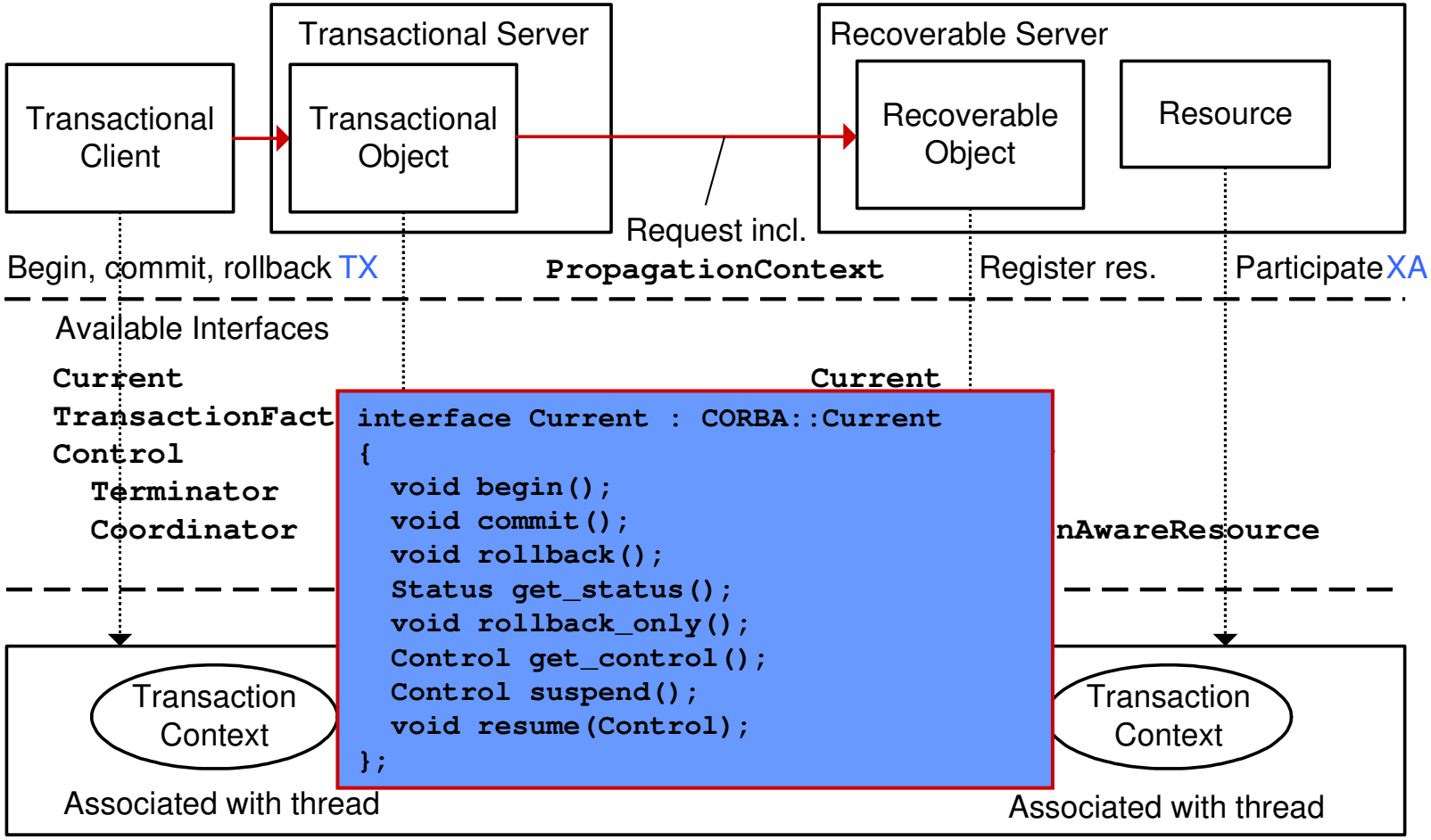
- Object Management Architecture (OMA) specifies syntax (IDL) and semantics (English text) for
  - ORB
  - CORBAservices
    - Naming, Event, **Transaction**, Security, …
  - CORBAfacilities
    - Vertical (industries)
    - Horizontal (utilities, internationalization, time, …)

- OTS
  - Developed by group including every major TP vendor
  - Extends transactional semantics to OO applications
  - Integration of object and non-object tx systems
  - > 12 implementations in 2000
  - Flat (mandatory) and nested (optional) tx models
  - Interoperable with X/Open DTP model (TX, XA, OSI/TP)

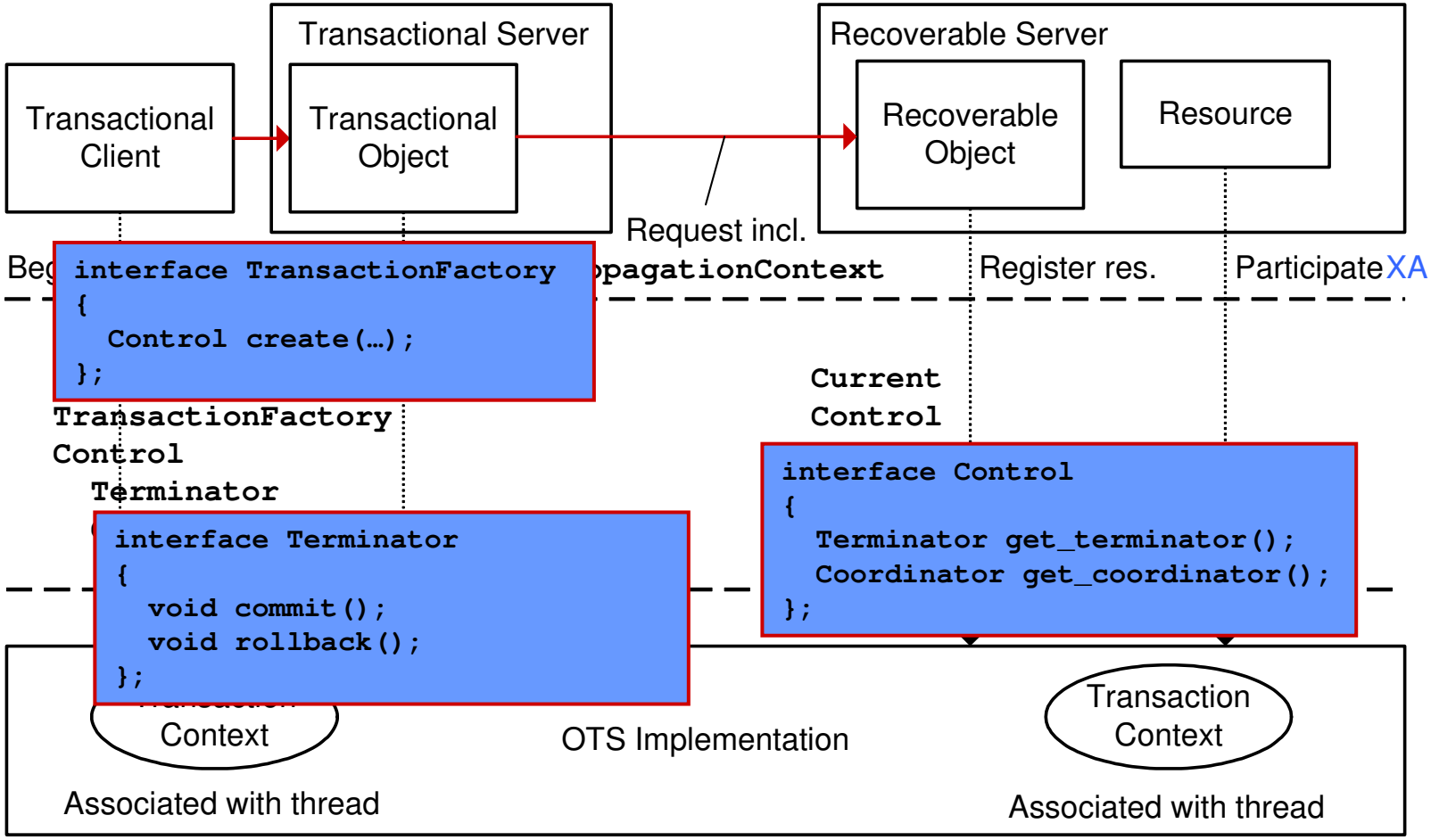
CORBA OTS :: Architecture and API



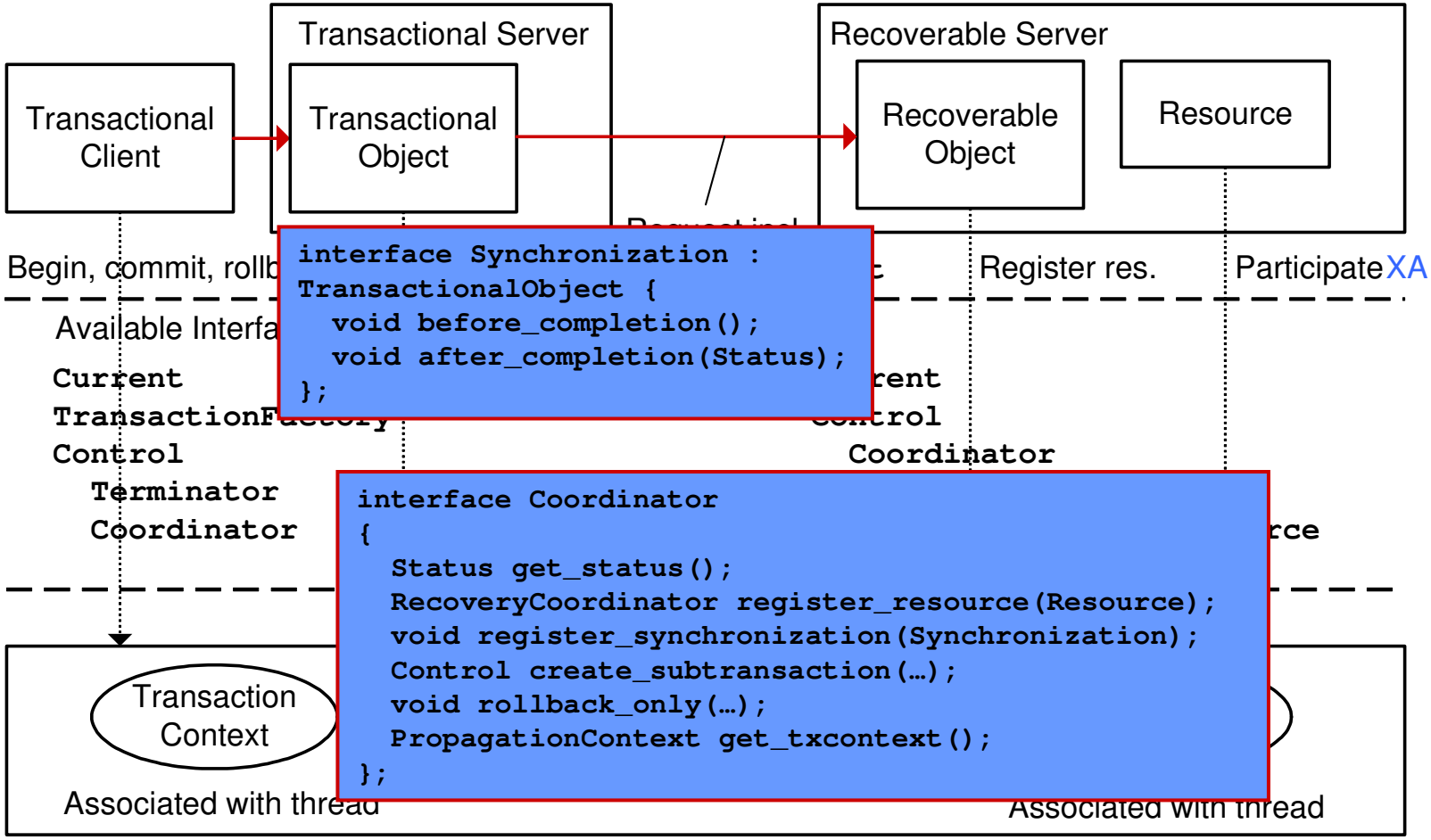
CORBA OTS :: Architecture and API



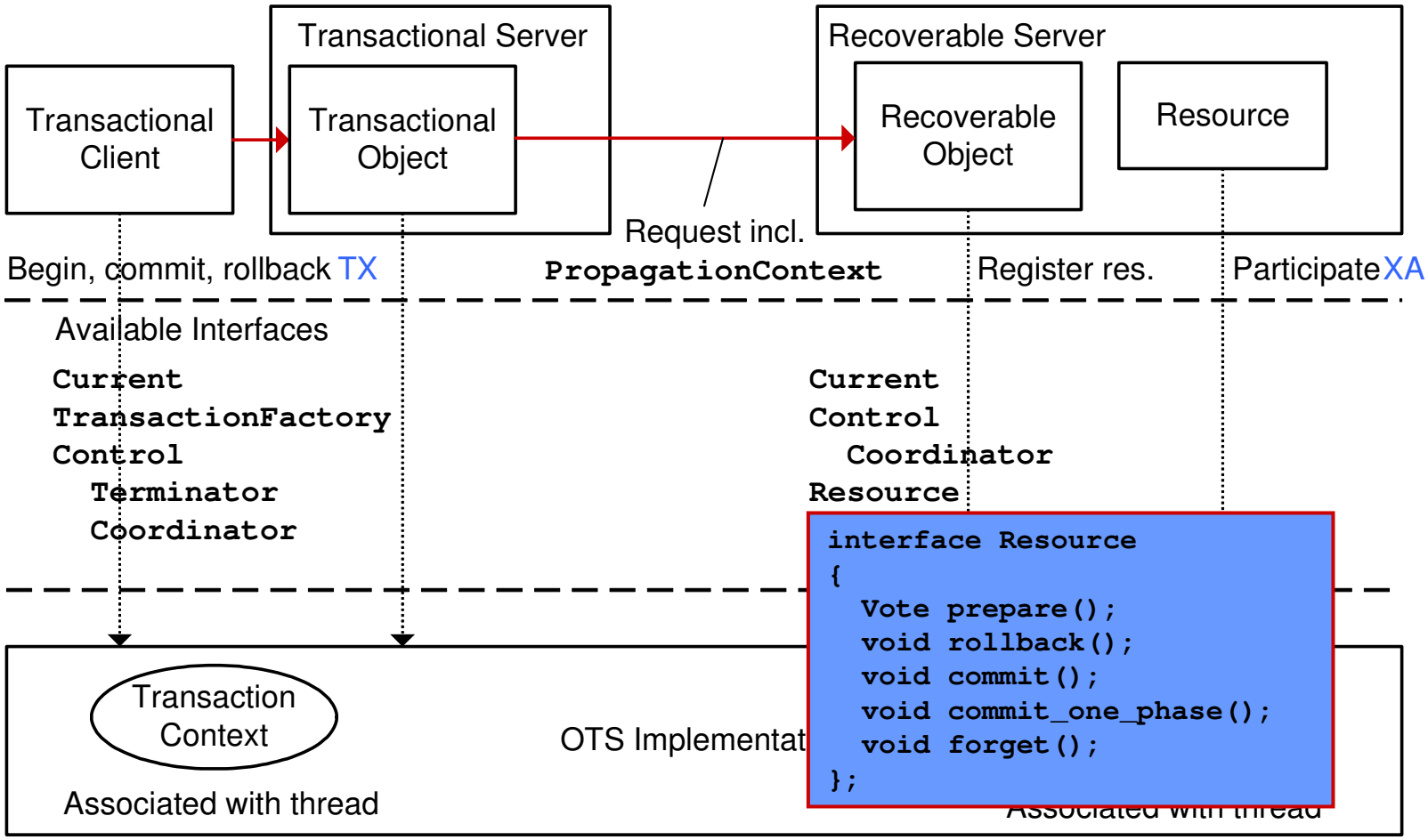
CORBA OTS :: Architecture and API



CORBA OTS :: Architecture and API



CORBA OTS :: Architecture and API



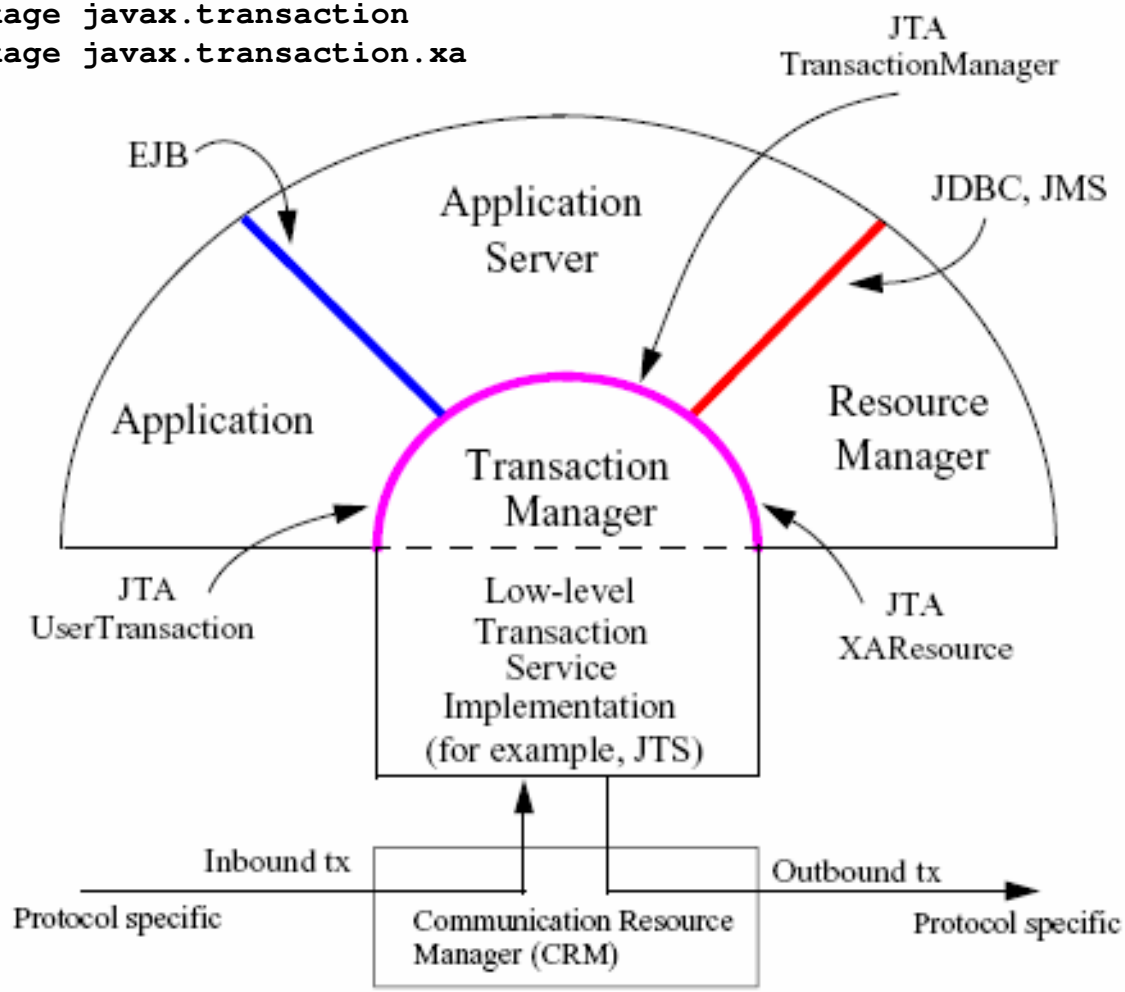
J2EE Java Transaction Service (JTS)

HAW HAMBURG

- Based on OTS 1.1
  - TransactionalObject interface used to declare transactionality, not POA policies
- Uses IIOP for RMI
- Part of most J2EE Application Servers

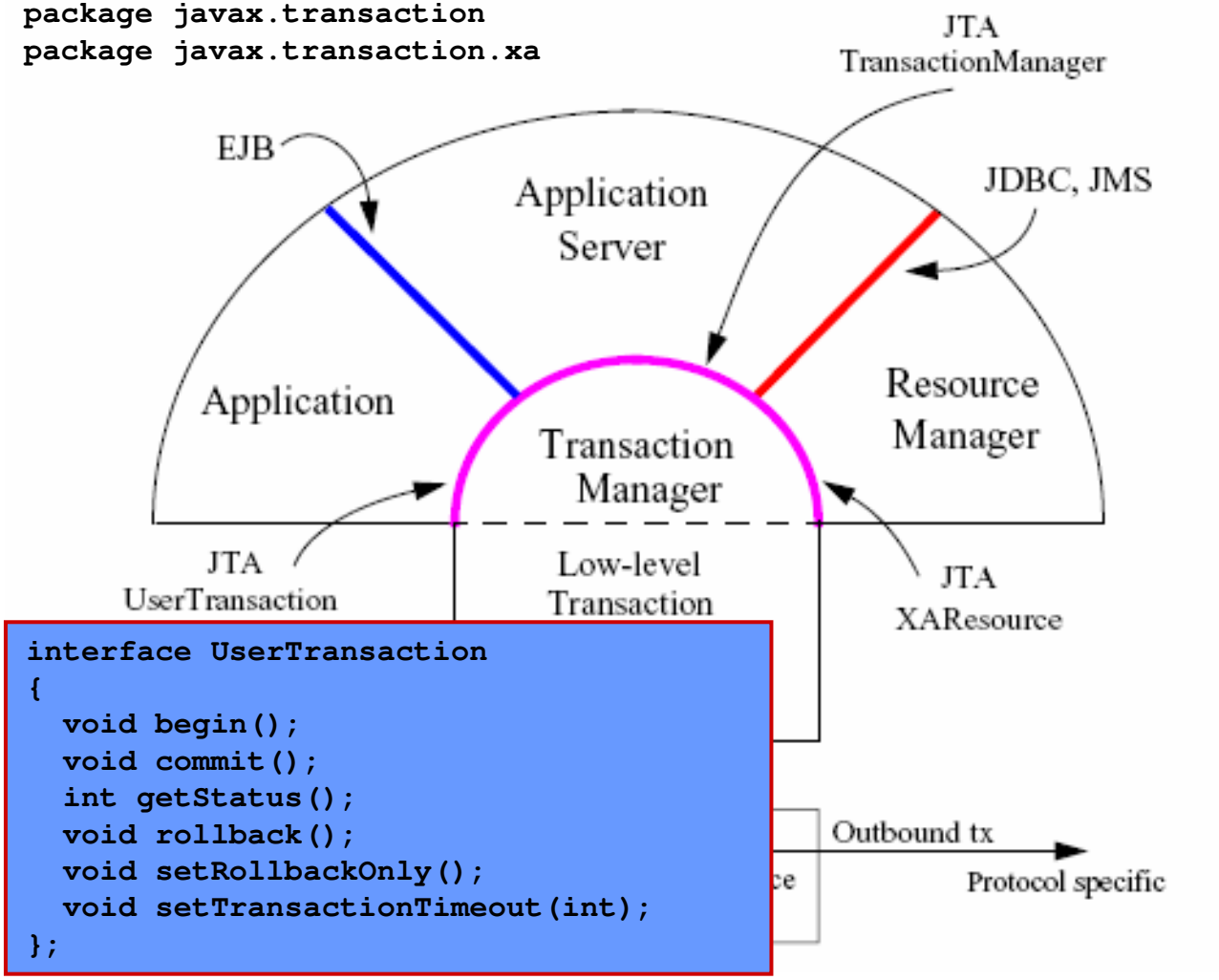
J2EE JTS :: Java Transaction API (JTA)

```
package javax.transaction
package javax.transaction.xa
```



J2EE JTS :: Java Transaction API (JTA)

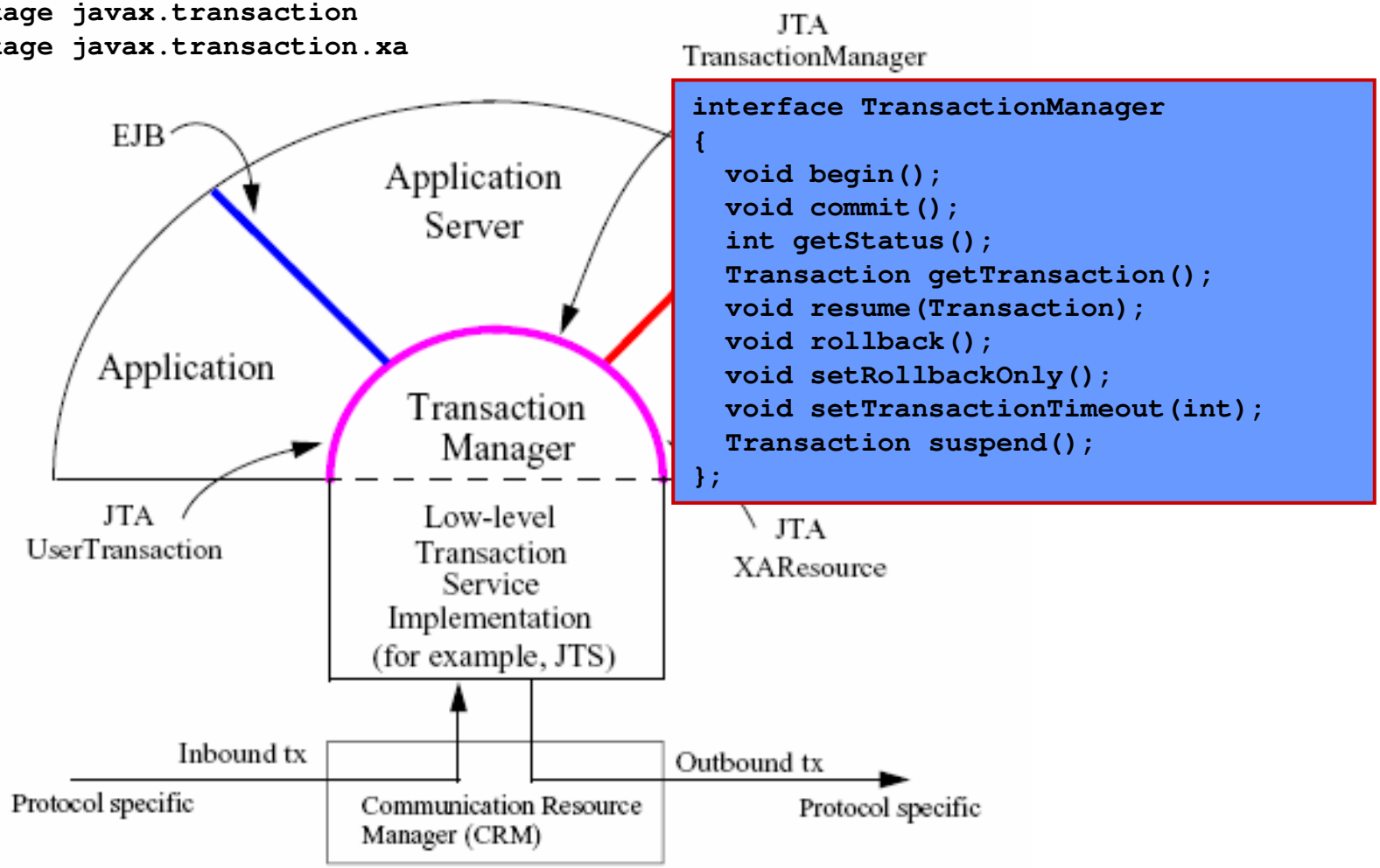
```
package javax.transaction
package javax.transaction.xa
```




```
interface UserTransaction
{
  void begin();
  void commit();
  int getStatus();
  void rollback();
  void setRollbackOnly();
  void setTransactionTimeout(int);
};
```

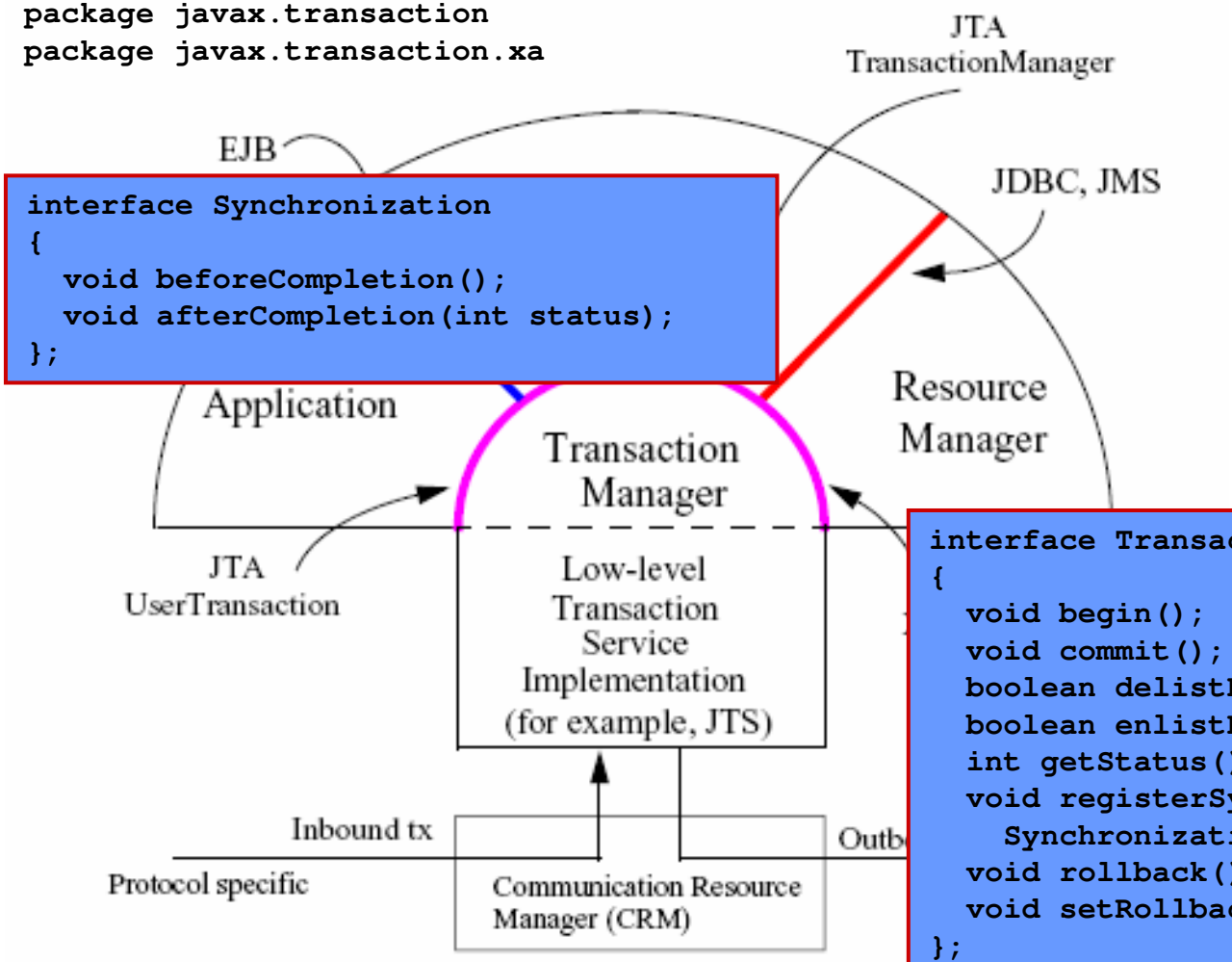
J2EE JTS :: Java Transaction API (JTA)

```
package javax.transaction
package javax.transaction.xa
```



J2EE JTS :: Java Transaction API (JTA)

```
package javax.transaction
package javax.transaction.xa
```

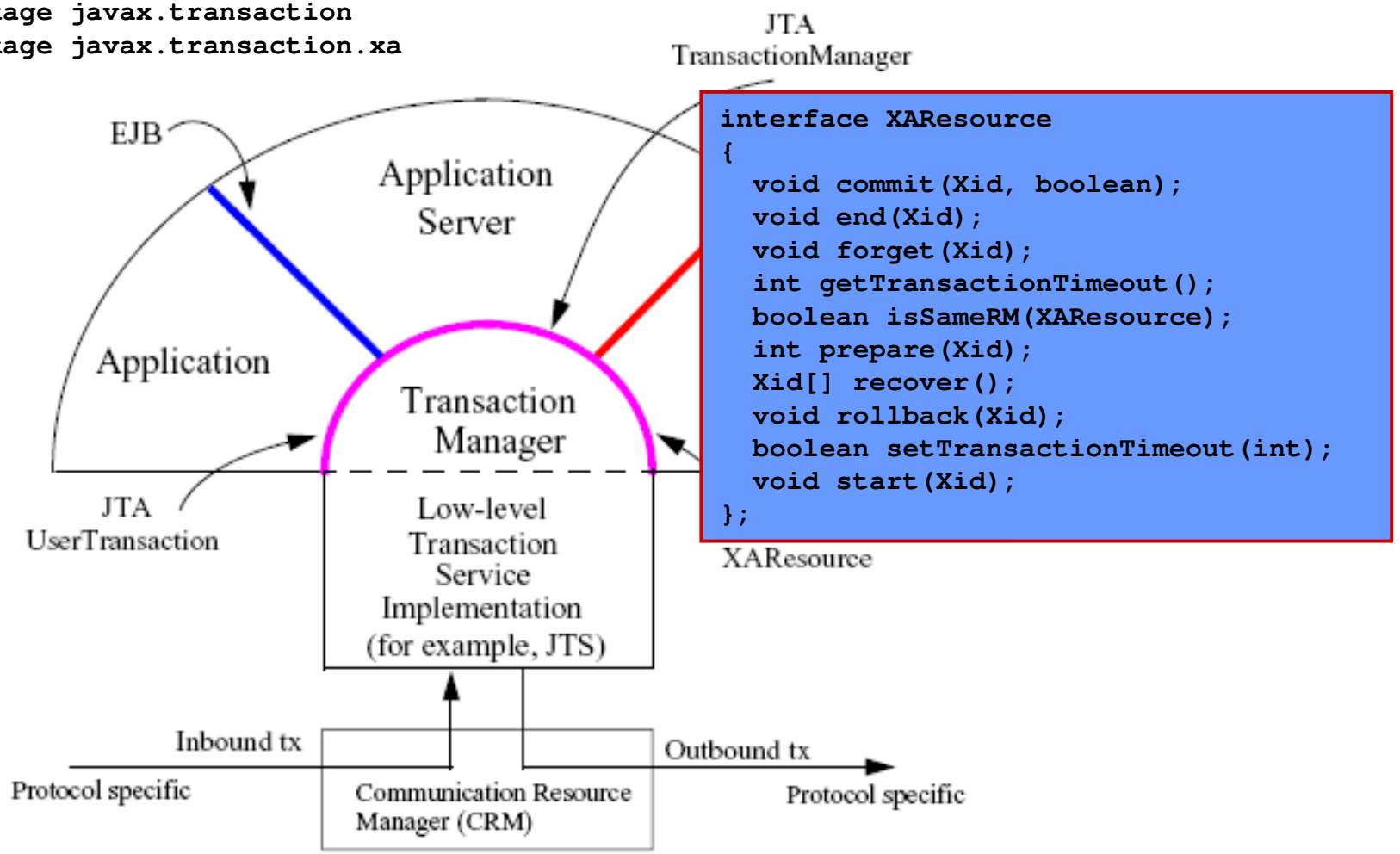



```
interface Synchronization
{
  void beforeCompletion();
  void afterCompletion(int status);
};
```

```
interface Transaction
{
  void begin();
  void commit();
  boolean delistResource(XAResource);
  boolean enlistResource(XAResource);
  int getStatus();
  void registerSynchronization(
    Synchronization);
  void rollback();
  void setRollbackOnly();
};
```

J2EE JTS :: Java Transaction API (JTA)

```
package javax.transaction
package javax.transaction.xa
```




```
interface XAResource
{
  void commit(Xid, boolean);
  void end(Xid);
  void forget(Xid);
  int getTransactionTimeout();
  boolean isSameRM(XAResource);
  int prepare(Xid);
  Xid[] recover();
  void rollback(Xid);
  boolean setTransactionTimeout(int);
  void start(Xid);
};
```

J2EE JTS :: Transactional J2EE components

HAW HAMBURG

- EJBs
  - Similar to basic level CORBA 3 components
  - Methods on the EJB interface can be transactional
    - Bean managed: Through JTA **`UserTransaction`**
    - Container managed: Through deployment descriptor at method level
      - **`NotSupported`** **`[none -> none][T1 -> none]`**
      - **`Required`** **`[none -> T2 ][T1 -> T1 ]`**
      - **`Supports`** **`[none -> none][T1 -> T1 ]`**
      - **`RequiresNew`** **`[none -> T2 ][T1 -> T2 ]`**
      - **`Mandatory`** **`[none -> ERR ][T1 -> T1 ]`**
      - **`Never`** **`[none -> none][T1 -> ERR ]`**
  - EJB Container does not need to use a JTS implementation, but must provide JTA
- JDBC + JMS providers
  - Act as resource managers
  - Must implement JTA XA interface

Thesis outline :: Where are we today?

HAW HAMBURG

Thesis

- Working title **"Long Running Transactions in Service-Oriented Environments"**
- Outline
  - …
  - Conceptual design of a framework that supports coordination and long running transactions
    - Why long running transactions? (AW1 recap)
    - Requirements
    - Specs: WS-C, WS-BA
    - Related work
    - Initial ideas
    - Project experiences
    - Outlook
  - …

Why long running transactions? (AW1 recap)

HAW HAMBURG

- Some activities in the TX can take long
- No satisfying QoS guarantees can be made
- Participants are unable to perform 2PC
- Participants must not hold locks for too long

- ACID is too restrictive
- Participating resource managers should commit as soon as possible

- **BUT** … the rental car should not stay booked if the restaurant is sold out …

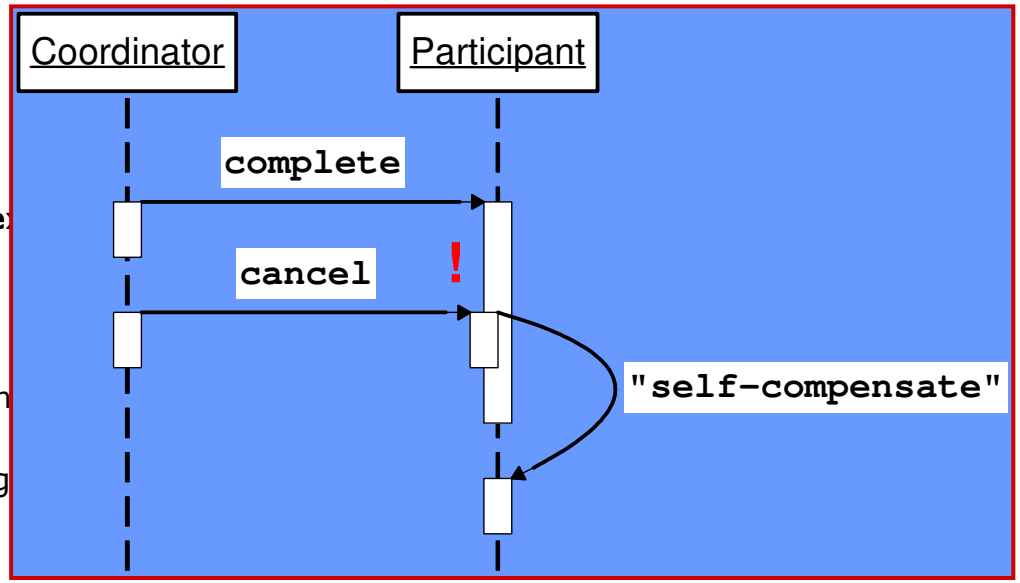
- Coordination and compensation handling required

Requirements

- **Loose coupling**
  - **Long running**
  - **No response time guarantees (at least none that permit sync.)**
  - **Asynchronous**
  - Heterogeneous
- Compensation based TX processing (CTP)
  - Resource manager logging
  - But "no" locking
  - Commutative actions
- **Coordination**
  - **Context management**
    - **Persistent Storage**
    - **Association with threads of execution**
    - **Direct/Indirect?**
  - **Synchronization**
    - **Correlation**
    - **Queuing**
    - Idempotent actions in case of unreliable messaging
- **Recovery**
  - … from context in persistent storage
- **Transparency**
  - **Middleware API**
  - **Code generation at development time**
  - AOP (injection, code generation at runtime)
- **Static** vs Dynamic
  - Policies and implemented interfaces

Requirements

- **Loose coupling**
  - **Long running**
  - **No response time guarantees (at least none that permit sync.)**
  - **Asynchronous**
  - Heterogeneous
- Compensation based TX processing (CTP)
  - Resource manager logging
  - But "no" locking
  - Commutative actions
- **Coordination**
  - **Context management**
    - **Persistent Storage**
    - **Association with threads of e**
    - **Direct/Indirect?**
  - **Synchronization**
    - **Correlation**
    - **Queuing**
    - Idempotent actions in case of un
- **Recovery**
  - … from context in persistent storag
- **Transparency**
  - **Middleware API**
  - **Code generation at development time**
  - AOP (injection, code generation at runtime)
- **Static** vs Dynamic
  - Policies and implemented interfaces



Specifications

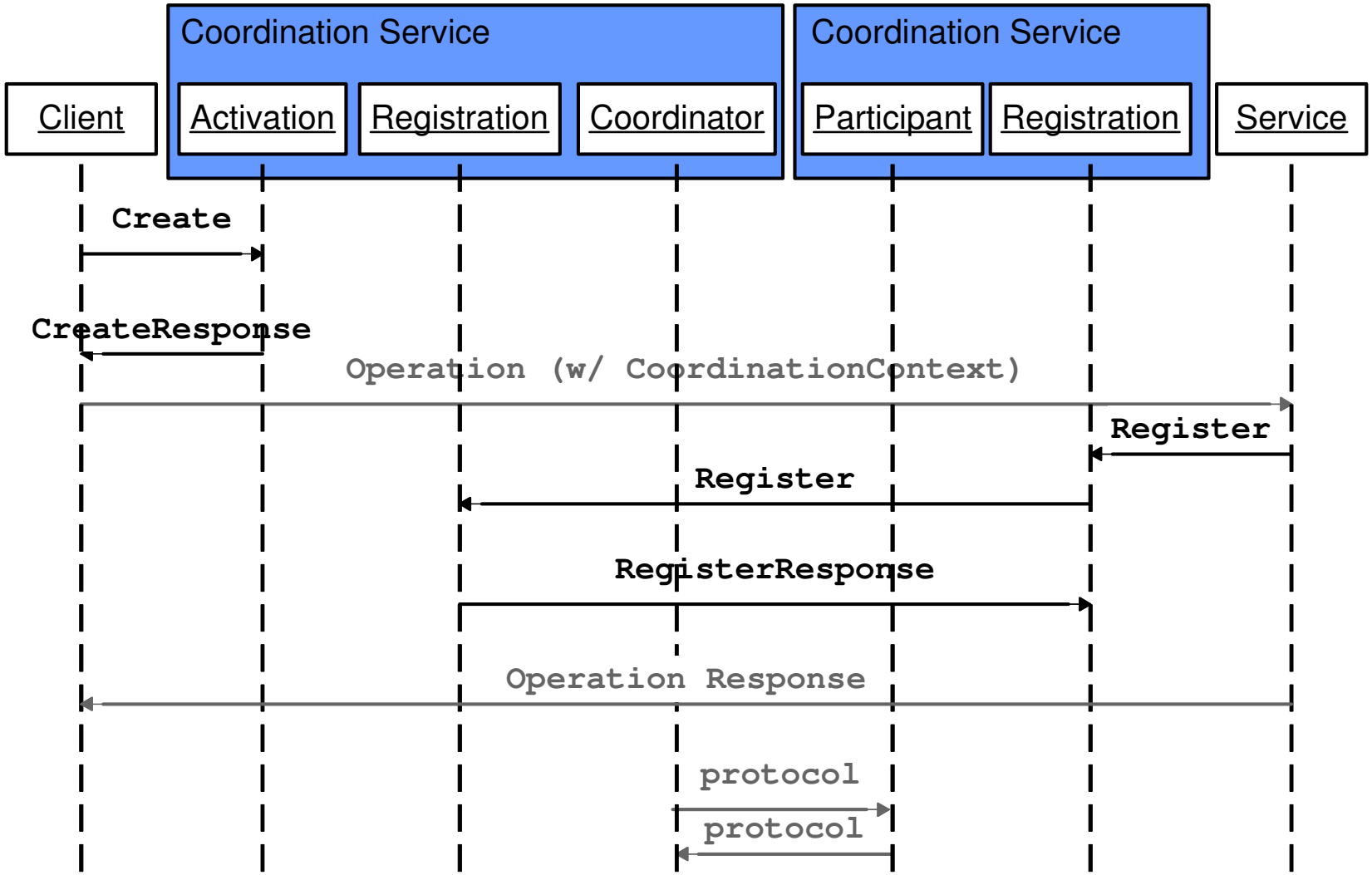
	BTP	WS-C/WS-T	WS-CAF
Since	2001	2002	2003
Companies	HP, Oracle, BEA	IBM, Microsoft	Arjuna, Fujitsu, IONA, Oracle, Sun
Specs(Pages)	BTP(163)	WS-C(23), WS-AT(21), WS-BA(23)	WS-CTX(59), WS-CF(63), WS-TXM(111)
Coordination Roles	Superior, Inferior	Coordinator, Participant	(see next slide)
Atomic	Atom (Open top 2 phase compl.)	AtomicTransaction (3 protocols, incl. 2PC)	ACID Transaction (2PC)
Long Running	**Cohesion**	**BusinessActivity (2 protocols)**	**Long R. Action Business Proc. TX**
Pros	Complete, well formed	Short, easy, generic, interoper., separation of C/T	Generic, interop., sep. C/T, complete, nested scopes
Cons	No legacy TX integr. "Bus./TX-Logic-Mix"	Incomplete, flat	Pretty long ;-)
Status	OASIS since 03/2001, but almost abandoned	OASIS since 10/2005 (!)	OASIS since 10/2003

Specifications

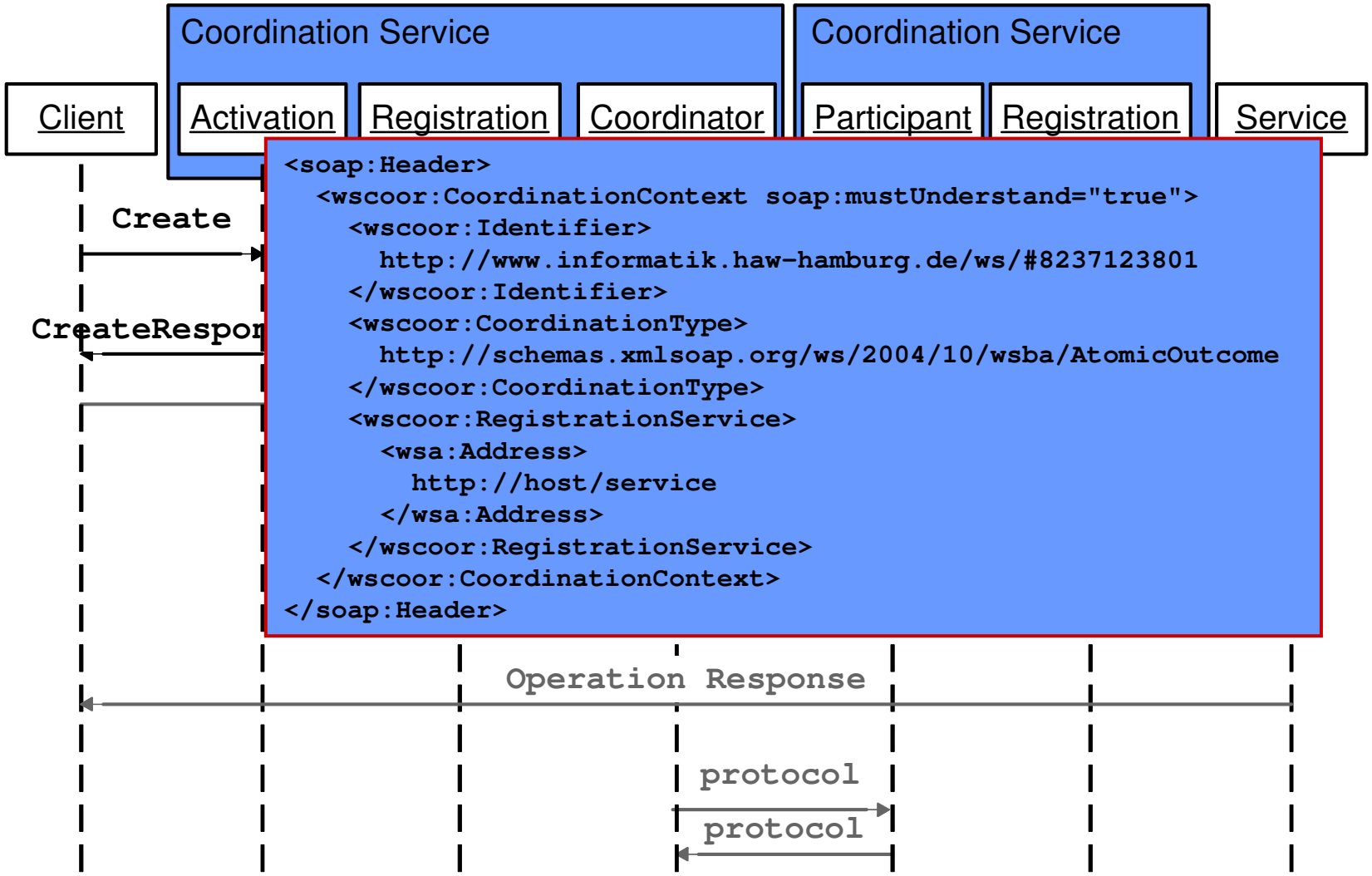
	BTP	WS-C/WS-T	WS-CAF
Since	2001	2002	2003
Companies	HP, Oracle, BEA	IBM, Microsoft	Arjuna, Fujitsu, IONA, Oracle, Sun
Specs(Pages)	BTP(163)	WS-C(23), WS-AT(21), WS-BA(23)	
Coordination Roles	Superior, Inferior	Coordinator, Participant	
Atomic	Atom (Open top 2 phase compl.)	AtomicTransaction (3 protocols, incl. 2PC)	
Long Running	**Cohesion**	**BusinessActivity** **(2 protocols)**	
Pros	Complete, well formed	Short, easy, generic, interoper., separation of C/T	Generic, interop., sep. C/T, complete, nested scopes
Cons	No legacy TX integr. "Bus./TX-Logic-Mix"	Incomplete, flat	Pretty long ;-)
Status	OASIS since 03/2001, but almost abandoned	OASIS since 10/2005 (!)	OASIS since 10/2003

Participant
CoordinatorParticipant
ParticipantCoordinator
ParticipantRespondant
ServiceCoordinator
ServiceRespondant
ClientCoordinator
ClientRespondant
RecoveryCoordinator

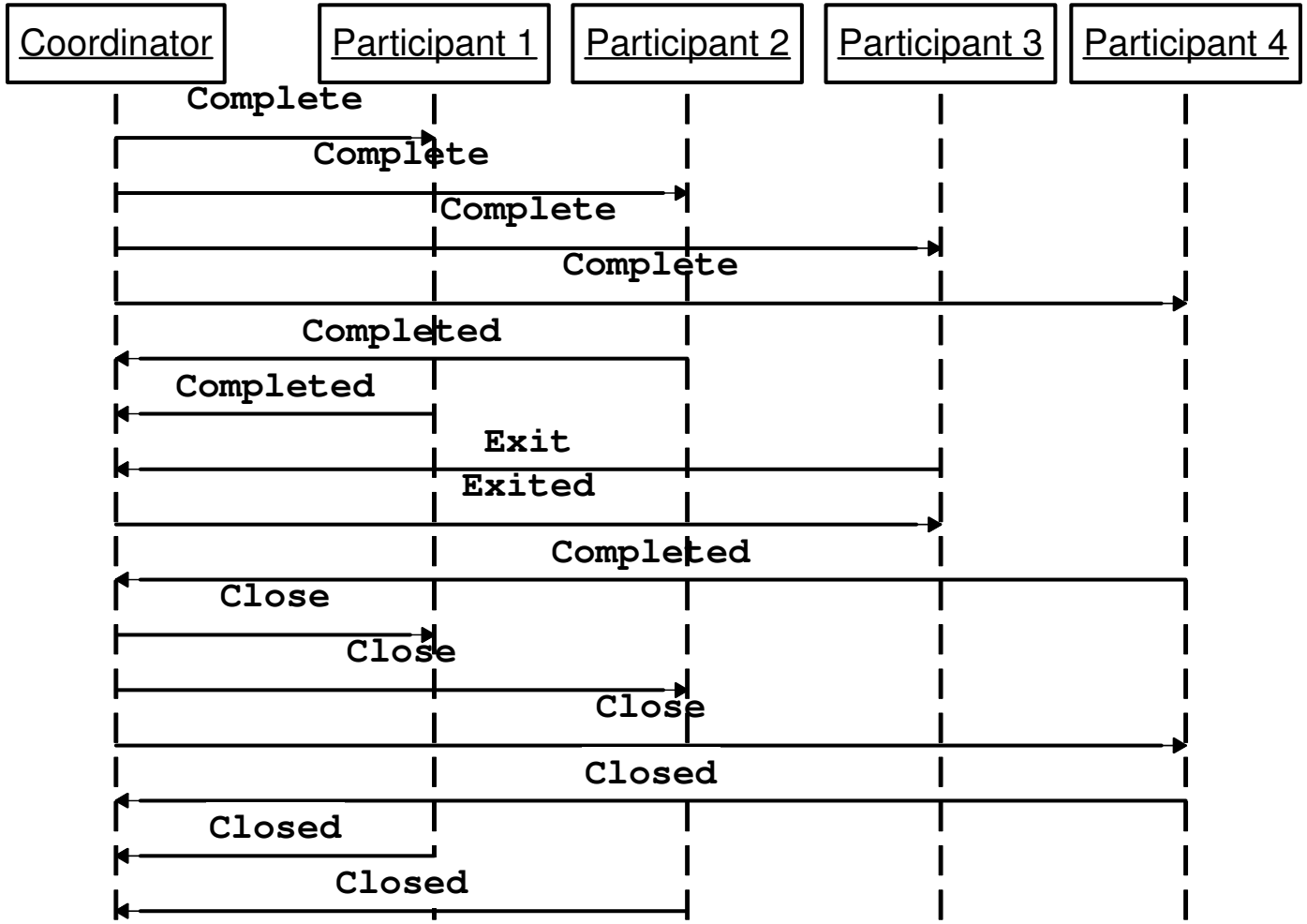
WS-Coordination



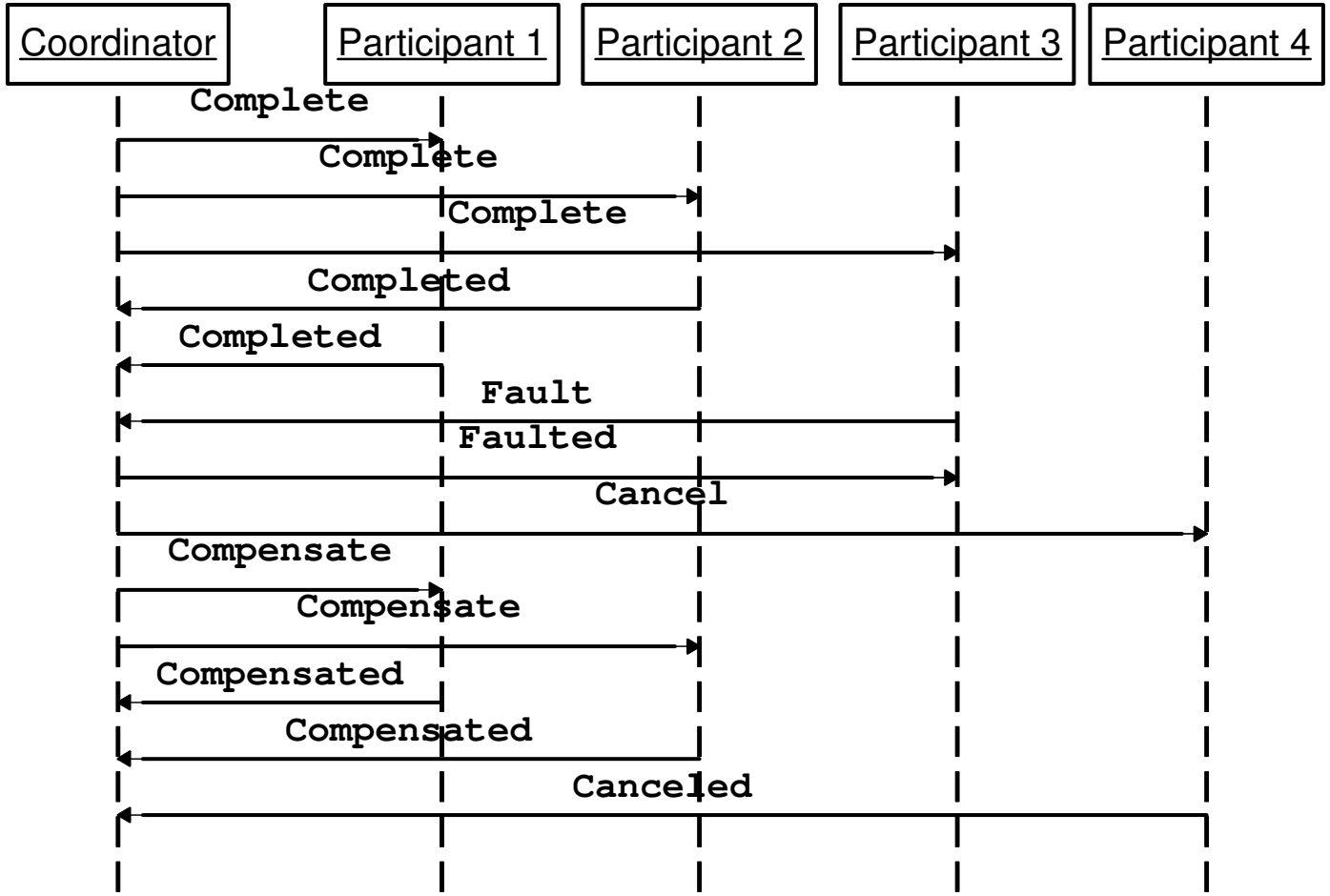
WS-Coordination



WS-BusinessActivity



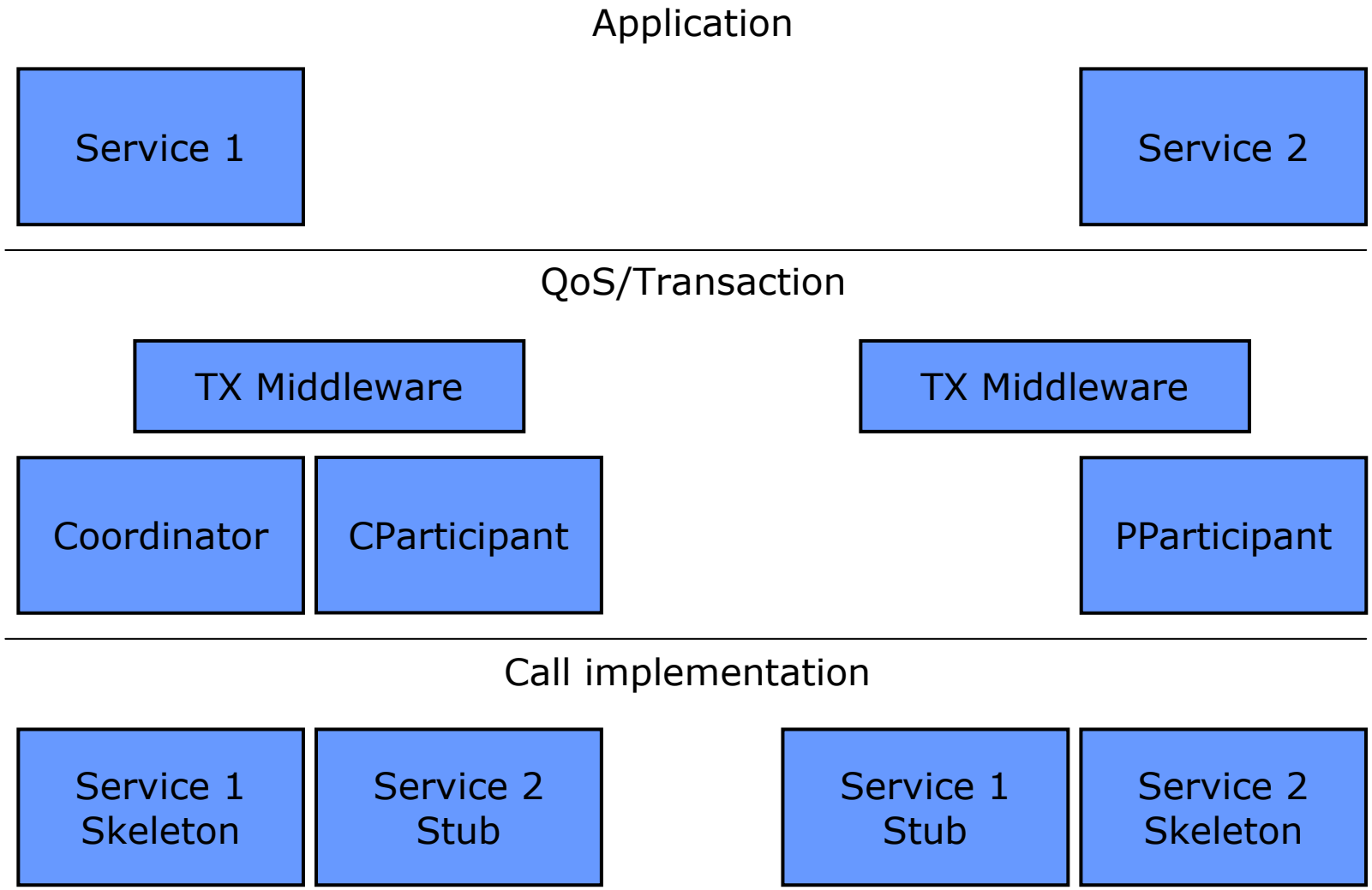
WS-BusinessActivity with compensation



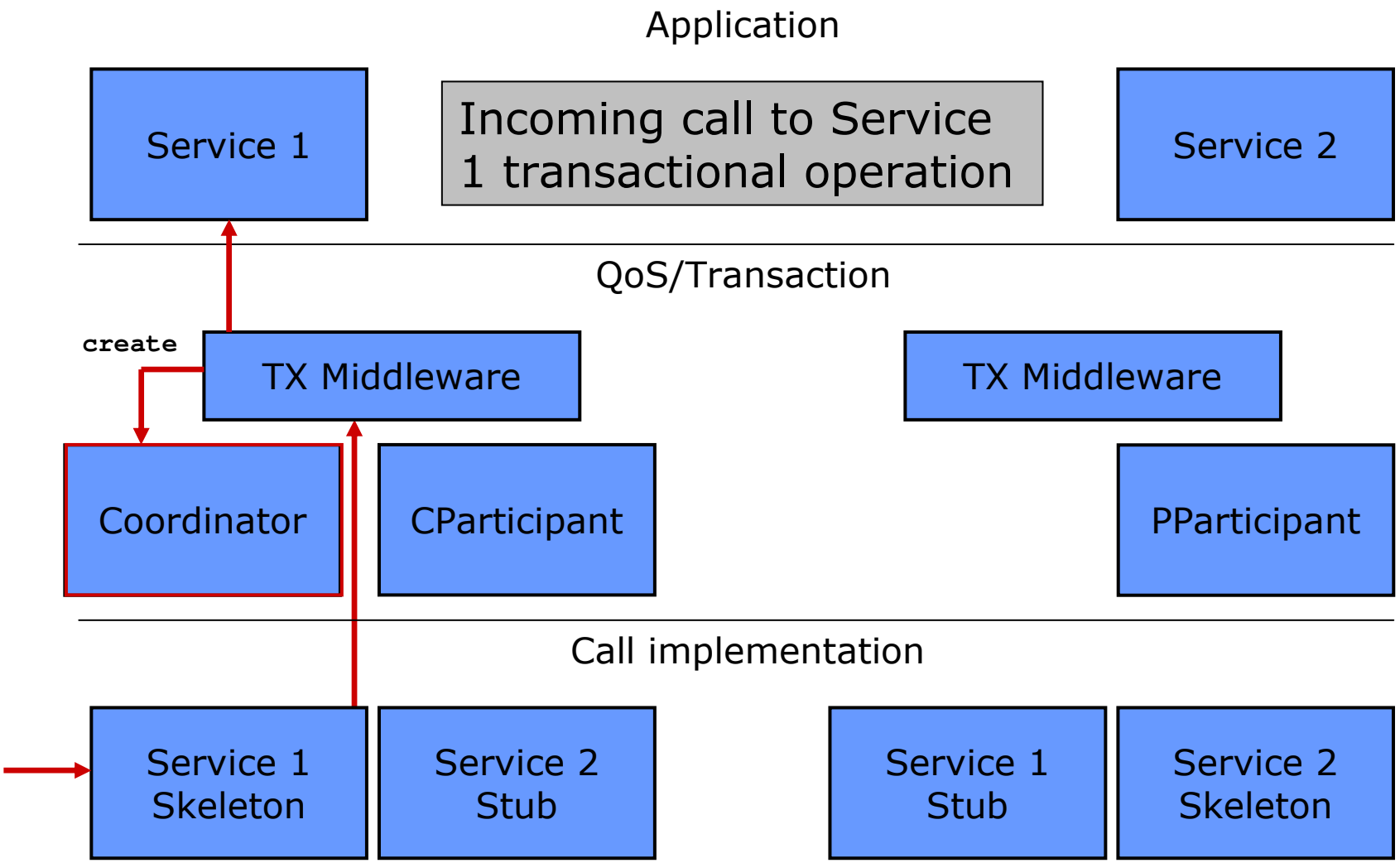
Related work

- JSR 95: J2EE Activity Service for Extended Transactions
  - "JTA for long running transactions" based on OMG Activity Service
  - Activities = units of work, potentially distributed, need not be transactional
- JSR 156: Java API for XML Transactions (JAXTX)
  - Support for JTA, J2EE Activity Service, BTP, WS-T, WS-TXM
  - Purpose: Provide common interface for different tx implemenations
- Apache Kandula
  - Based on Apache Axis 2
  - Subversion Branch Kandula_1 (Early 2005)
    - WS-AT only
    - Mapping of CoordinationContext identifier to JTA UserTransaction in Axis Handler
    - Tested with JBoss and JOTM tx managers
  - Subversion trunk
    - Significantly changed and changing
    - No JTA code so far
    - Seems to be a refactoring effort on Kandula_1
    - Still no WS-BA code
- EJB 2.1 Web Service Endpoint EJBs
  - Stateless Session Beans
  - TX support for local JTA transactions only (**`Mandatory`** not allowed!)

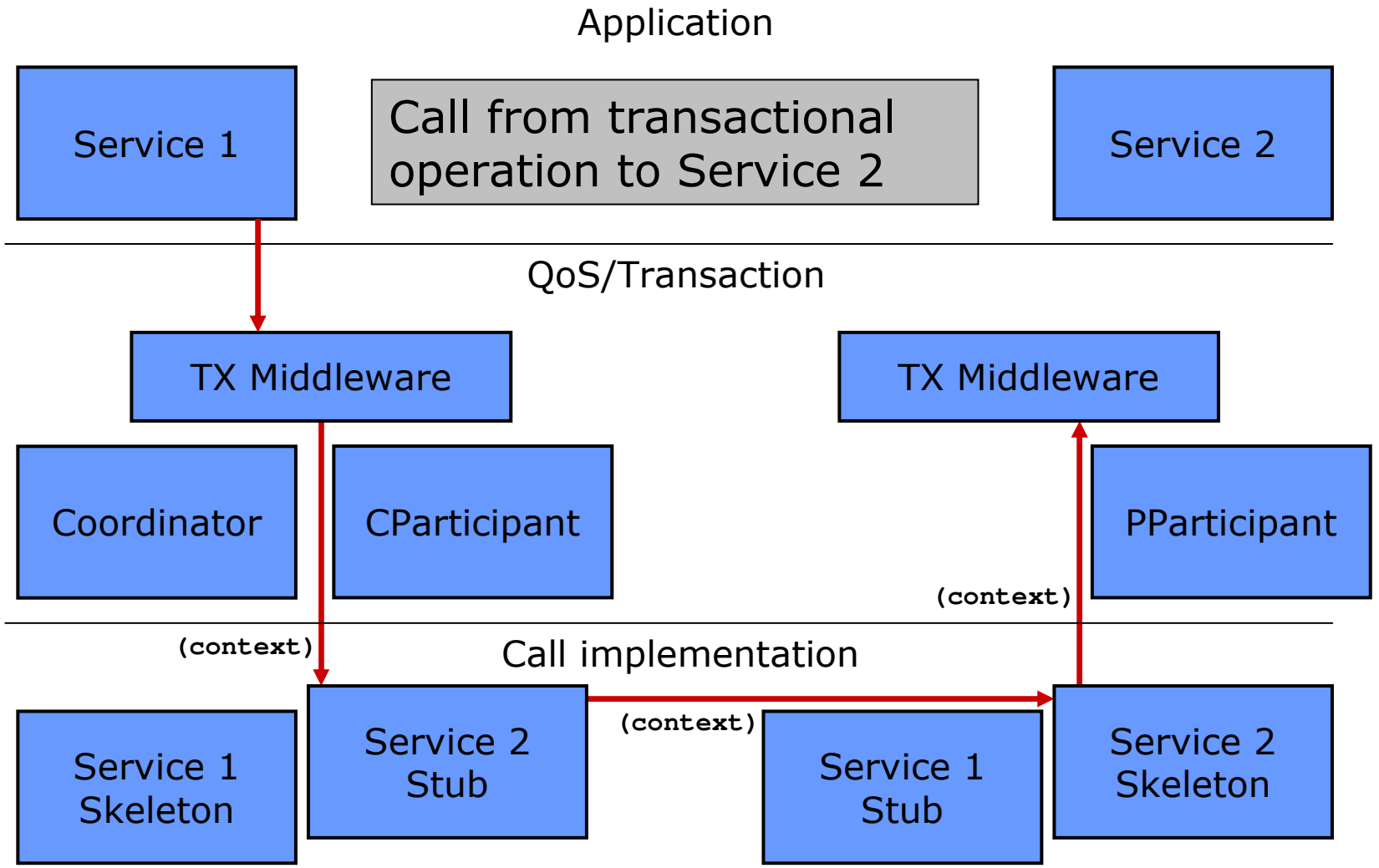
Initial ideas :: Conceptual overview / Architecture



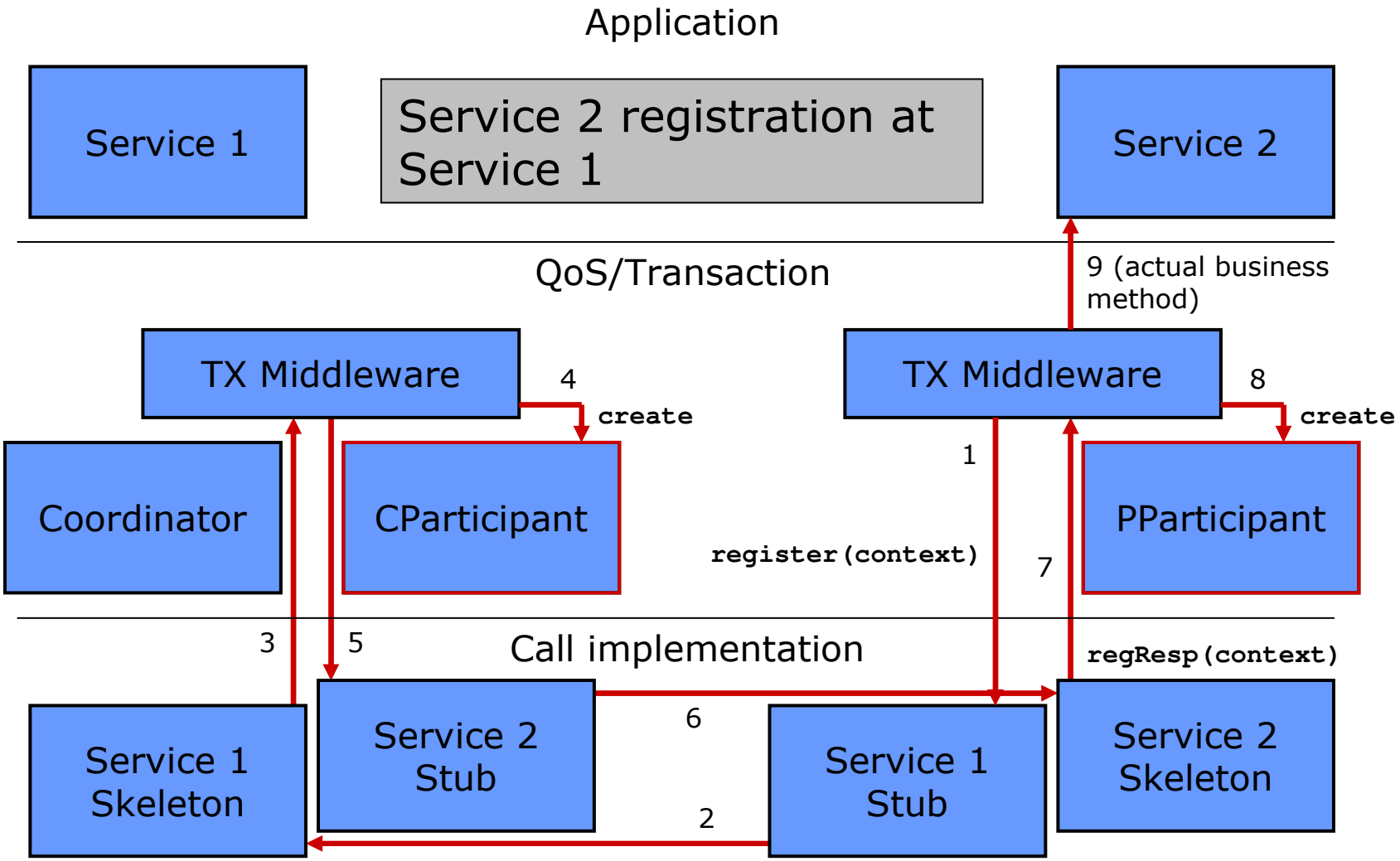
Initial ideas :: Conceptual overview / Architecture



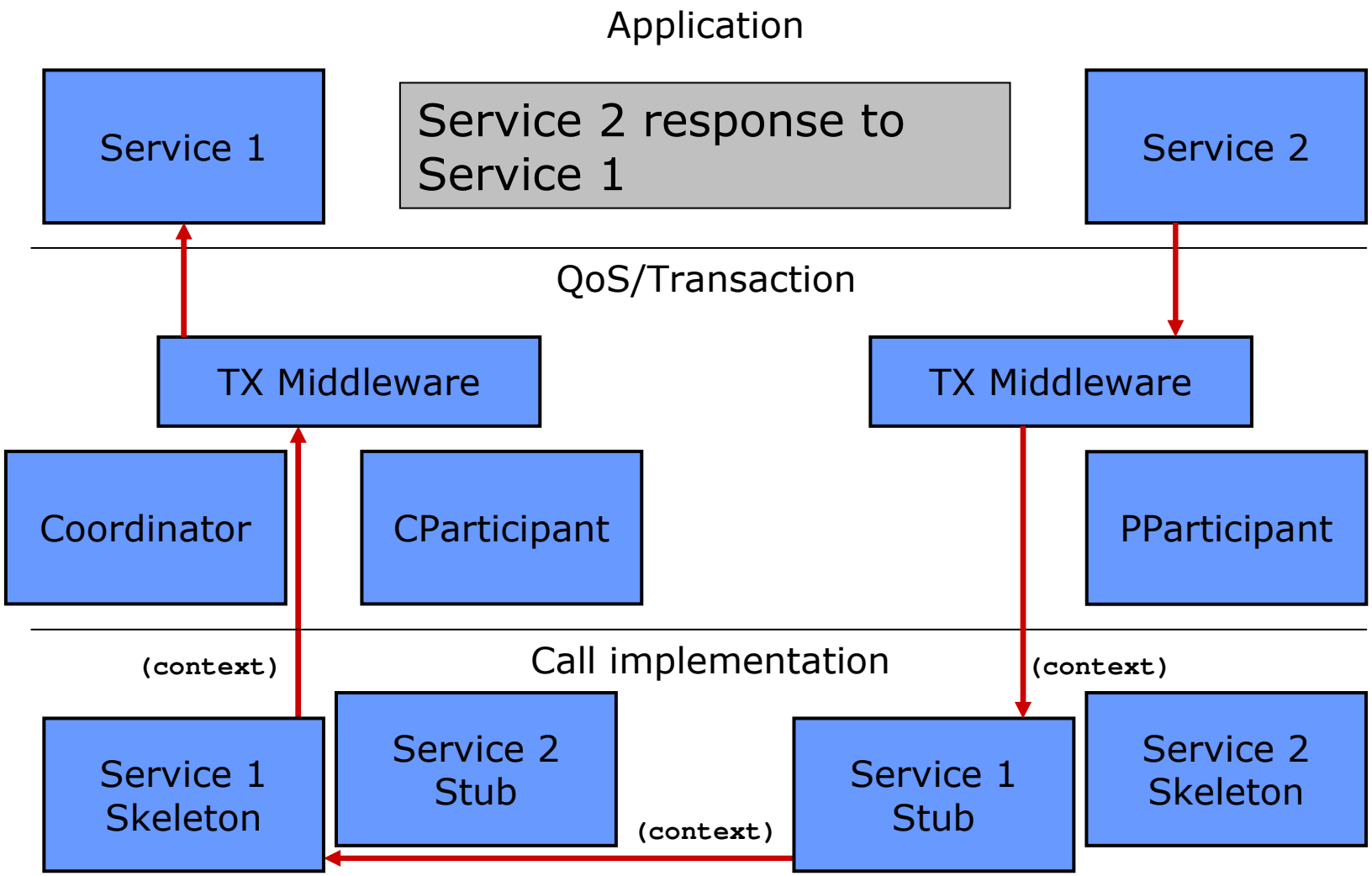
Initial ideas :: Conceptual overview / Architecture



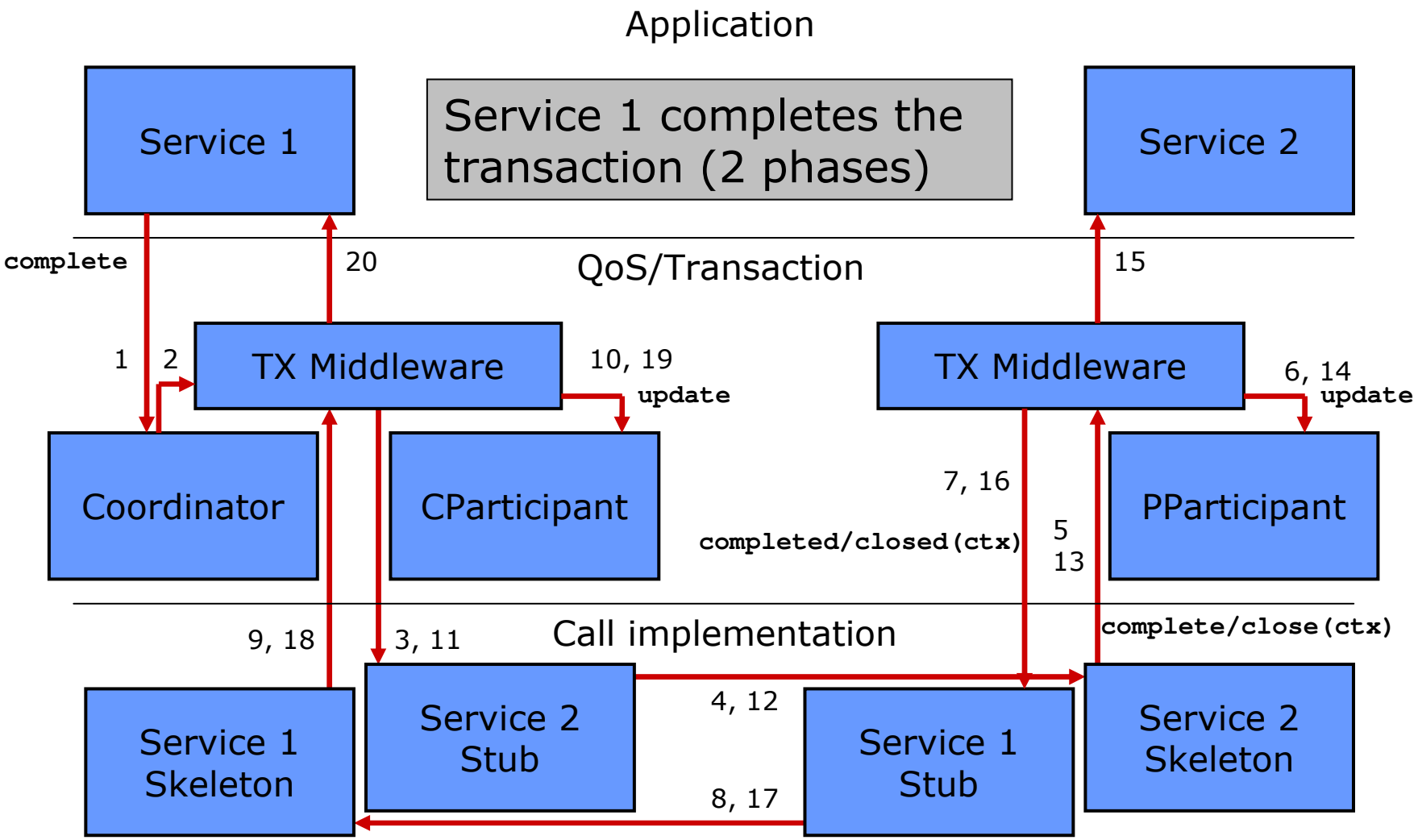
Initial ideas :: Conceptual overview / Architecture



Initial ideas :: Conceptual overview / Architecture



Initial ideas :: Conceptual overview / Architecture



Initial ideas :: Notes

- Asynchronous call handling
  - Model services and protocols as state machines, persist state after each operation
  - Registry for contexts, coordinators, participants, operation call backs, …
  - One state machine per service and per coordinator/participant pair

- Transparency
  - Hide as much protocol processing as possible from developers
  - Some completion semantics may depend on business logic
    - Need to expose operations on coordinator/participants for service managed tx
  - Use / extend code generation (e.g. Axis WSDL2Java)
    - Like EJB: Container managed tx, Service managed tx
    - Generated code could use JAXTX
  - Generated code (possibly) different for each service role
    - Coordinator / participant / coordination type / protocol type / operation

Project experiences / Outlook

- Project: The usual difficulties
  - Technical
    - New to JBoss
    - Setup of development environment with debugging, SOAP monitor, …
    - Web Services container (Axis 1.2.1) issues
  - Logistical
    - Not enough time to produce meaningful results
    - Had to revise quite a few TX/J2EE/Web Services concepts
    - Lower targets would not have made much sense, but maybe three to four motivated people as "TX team"
- Project: Successes (by 01/2006)
  - "Rapidly" (net time) prototyped activation, registration, participant initiated BA completion with mixed outcome
  - Foundation for a framework prototype with code generation
- Outlook
  - Project: Integration concept with ESB and Security
  - Thesis: Detailed framework design
  - Thesis: "TX-Lite" framework for devices with limited resources

Thank you!
In case you like to learn more...

- Gray, Reuter, **Transaction Processing: Concepts and Techniques**, Morgan Kaufman, 1993
- Szyperski, **Component Software: Beyond Object-Oriented Programming**, Addison-Wesley, 1998
- Siegel, **CORBA 3 Fundamentals and Programming**, OMG Press, 2000
- Pavlik, Maron, Little, **Java Transaction Processing**, Prentice Hall, 2004
- Weerawarana, Curbera, Leymann, Storey, Ferguson, **Web Services Platform Architecture**, Pearson, 2005

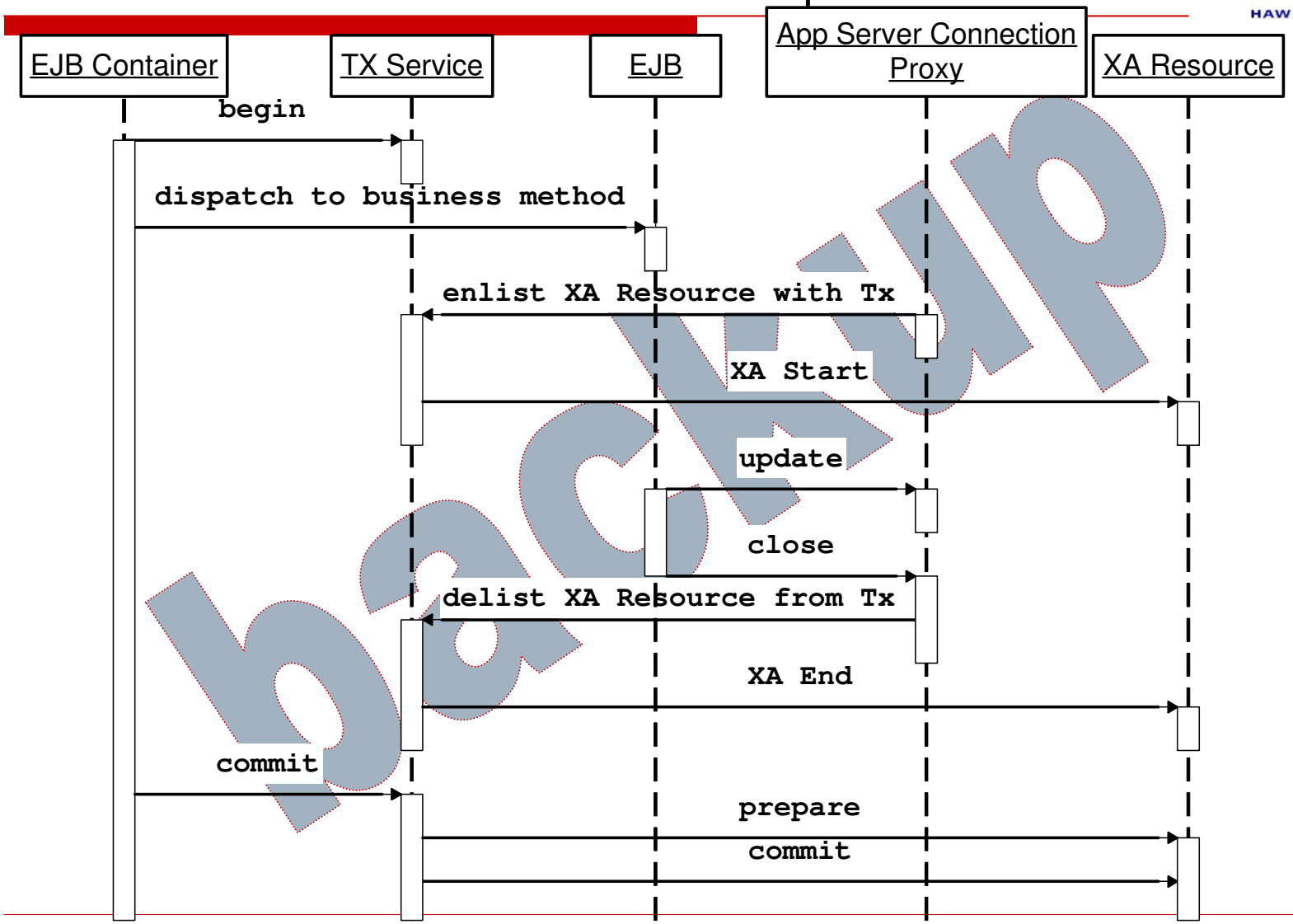
- … plus several articles and papers (see report)

J2EE JTS :: Transactional J2EE components :: EJB with bean managed transaction code sample

HAW HAMBURG

```
public class MySessionEJB implements SessionBean {
  EJBContext ejbContext;
  public void someMethod(...) {
    javax.transaction.UserTransaction ut;
    DataSource ds; Connection dcon; Statement stmt;
    QueueConnectionFactory qcf; QueueConnection qcon; Queue q;
    QueueSession qsession; QueueSender qsender; Message message;
    // obtain db conn and queue session objects through JNDI InitialContext lookup
    ...
    // Now do a transaction that involves the two connections.
    ut = ejbContext.getUserTransaction();
    // start the transaction
    ut.begin();
    // Do database updates and send message. The container automatically enlists
    // dcon and qsession with the transaction.
    stmt.executeQuery(...);
    stmt.executeUpdate(...);
    stmt.executeUpdate(...);
    message = qsession.createTextMessage(); message.setText("some message");
    qsender.send(message);
    // commit the transaction
    ut.commit();
    // release connections
    stmt.close(); qsender.close(); qsession.close(); dcon.close(); qcon.close();
  }
  ...
}
```

J2EE JTS :: Transactional J2EE components



Excursus: Experiences @ Techniker Krankenkasse

- Form based case management system
- Problems
  - 2PC not supported by many systems (e.g. SAP)
  - Distributed tx not supported by many (legacy) systems
  - TX manager overhead
  - Object TX (CORBA etc.): Traffic!
  - Atomic only manageable within one company/enterprise
- DB transactions: Should never be long
- Solutions
  - Long tx on in memory object models only
  - O/R mapping
  - Write once (≈ACID), fail if concurrent changes occurred
  - Optimistic locking (change counter, select for update)
  - Time frame: absolute max is several hours, but mostly rather minutes for one work item
- Session groups long running tx in "user interaction"
  - Logical sequence of atomic high level tx (→workflow)
  - Only manageable with coarser grained higher level calls (→services)
- Design system so that inconsistencies are also avoided by the nature of the work flow
  - Compensation, etc.