Web Service Composition Application Framework

Malik SAHEB
Arjuna Technologies Ltd

arjuna
middleware for reliability

What is WS-CAF?

arjuna
middleware for reliability

- Collection of 3 specifications designed to be used independently or together
  - WS-Context
    - Context service
  - WS-Coordination Framework
    - Framework for pluggable coordination protocols
  - WS-Transaction Management
    - Three transaction models for Web services

WS-Context

arjuna

middleware for reliability

WS-Context

arjuna
middleware for reliability

- Allows composite applications to share common information.
- Defines Notion of An “Activity”
  - An execution of a series of related interactions with a set of cooperating web services.
  - Operations correlated by a context associated with activity
- A Context
  - A way of doing correlation of messages
  - Context bound to one activity
  - Examples
    - Single-session sign-on
    - Transactions
    - Database session identifier

Context Structure

- An XML document containing a unique identifier and optional data specific to a related activity.
- Typically included in the SOAP header of messages to and from web services participating in an activity.
- Can be passed as a referenceable URI (by reference) or in its longer form (by value)

Context schema

```
<xs:complexType name="ContextType">
     <xs:sequence>
         <xs:any namespace="##other" processContents="lax" minOccurs="0"
                   maxOccurs="unbounded"/>
         <xs:element name="context-identifier" type="tns:contextIdentifierType"/>
         <xs:element name="context-service" type="ref:ServiceRefType"
                  minOccurs="0"/>
         <xs:element name="type" type="xs:anyURI"/>
         <xs:element name="context-manager" type="ref:ServiceRefType"
                  minOccurs="0"/>
         <xs:element name="parent-context"   type="tns:ContextType"
                   minOccurs="0"/>
     </xs:sequence>
     <xs:attribute name="timeout" type="xs:int" use="optional"/>
     <xs:attribute ref="wsu:Id" use="optional"/>
</xs:complexType>
```

Contains

- *Context identifier* - URI, MUST be unique
  - with optional wsu:Id attribute.
- *Context service* – ServiceRefType (OPTIONAL)
  - locate the authority having generated the context
  - ServiceRefType = Generic structure to deal with addressing
- *Type* – URI
  - the type of the protocol supported by the context,
- *Context-manager* ServiceRefType (OPTIONAL)
  - to get data associated with a context-identifier
  - if present, the context has been passed by reference
- *parent-contexts (*OPTIONAL)
- *timeout* attribute (OPTIONAL)
  - indicates for how long the context information is valid;
- *wsu:Id* attribute (OPTIONAL)
  - used to support signing or encrypting the context structure*.*

WS-Context Services

- Defines web services for maintaining contexts
- Ability to pass contexts by reference or by value.

➢ Context Service

- – creating - *begin*
- – completing - *complete*
- – getting status of a context – *getStatus*
    - *ACTIVE, COMPLETING, COMPLETED, NO_ACTIVITY, UNKOWN*
- – Setting and getting timeout – *setTimeout*, *getTimeout*

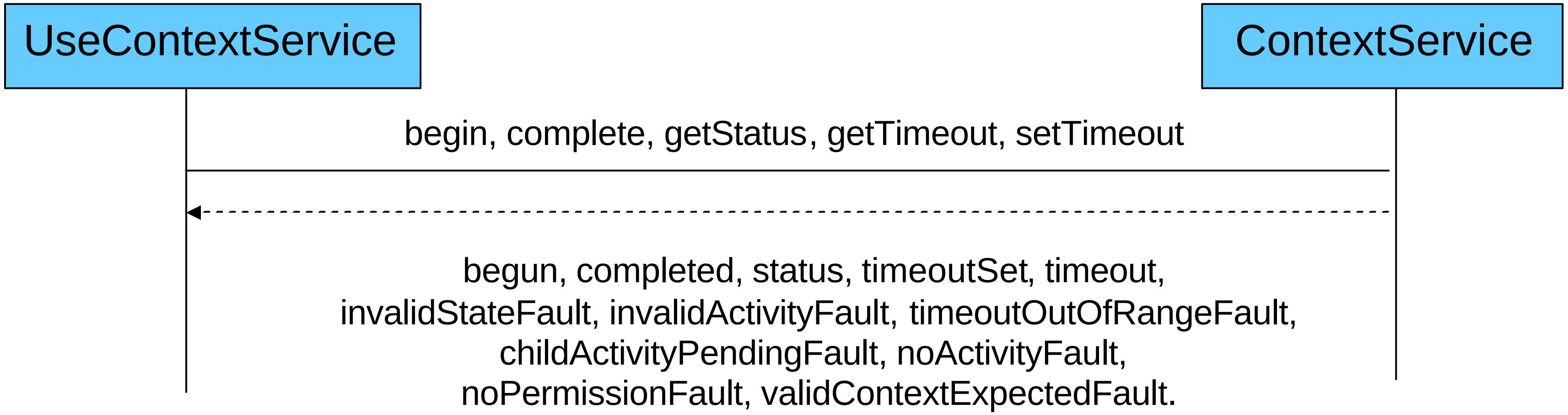
➢ Context Manager Service

- – Obtaining/setting a content of a context got by reference – *getContents*, *setContents*

Interactions by Callback

arjuna

middleware for reliability

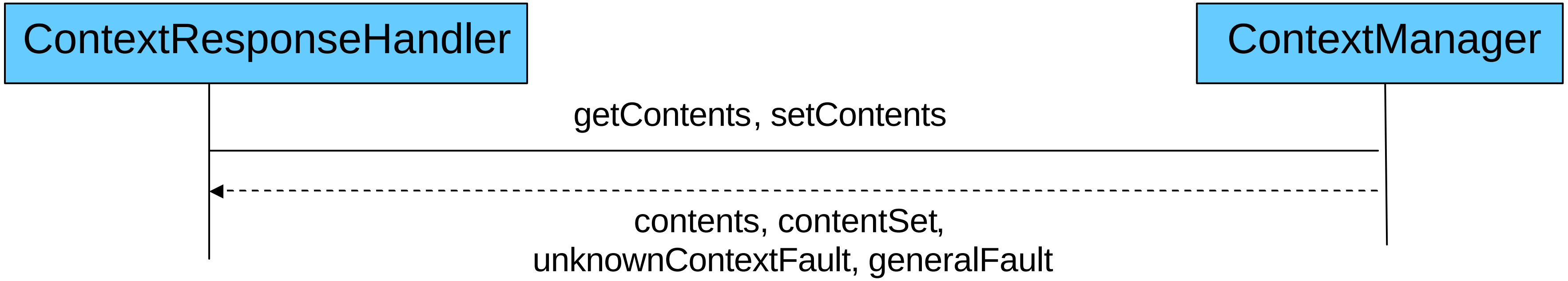


UseContextService

ContextService

begin, complete, getStatus, getTimeout, setTimeout

begun, completed, status, timeoutSet, timeout,
invalidStateFault, invalidActivityFault, timeoutOutOfRangeFault,
childActivityPendingFault, noActivityFault,
noPermissionFault, validContextExpectedFault.



ContextResponseHandler

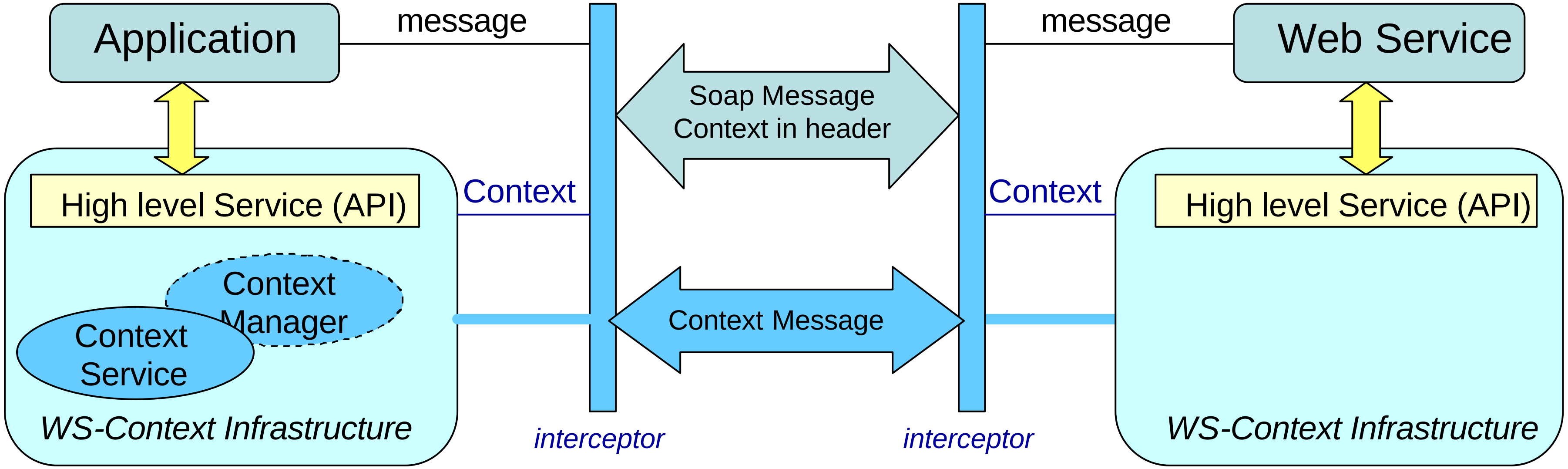
ContextManager

getContents, setContents

contents, contentSet,
unknownContextFault, generalFault

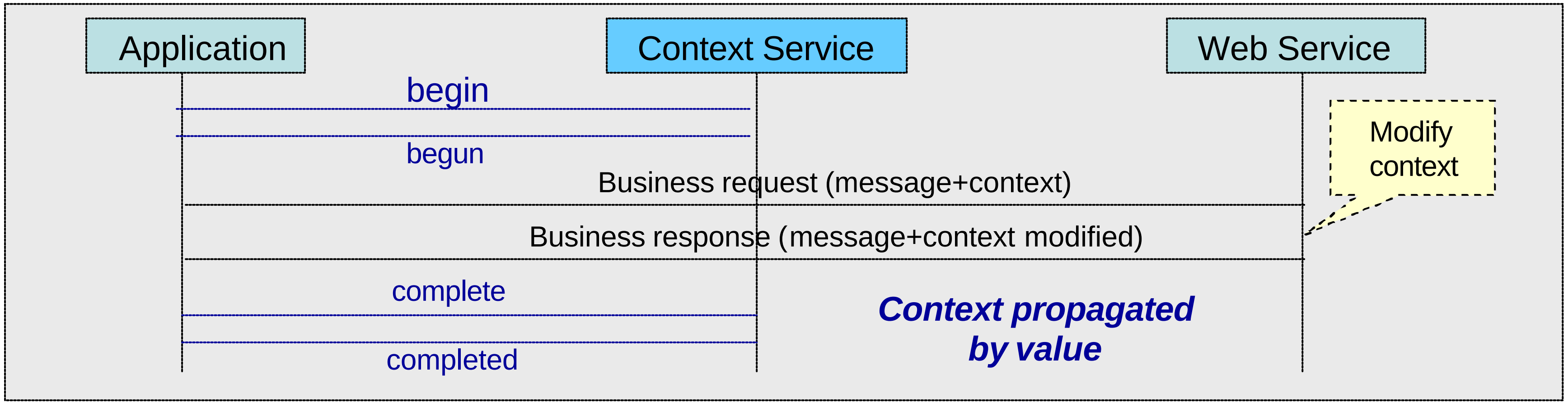
Architectural Overview

arjuna
middleware for reliability

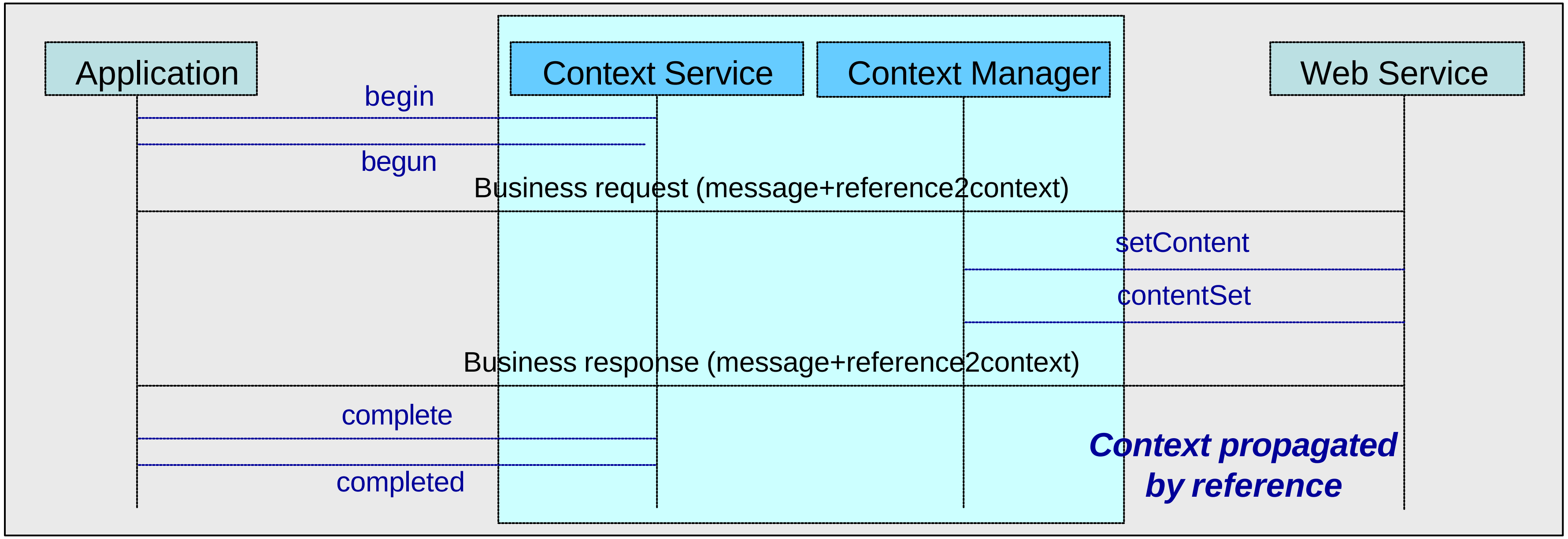


Interactions Overview

arjuna
middleware for reliability


Application
Context Service
Web Service
begin
begun
Modify context
Business request (message+context)
Business response (message+context modified)
complete
completed

Context propagated by value



Context propagated by reference

WS-Coordination Framework

arjuna

middleware for reliability

WS-CF

- Coordination is more fundamental than transactions
  - Security
  - Replication
  - Transactions
  - …
- Coordination could be seen as “disseminating information by a *coordinato*r to a number of *participants* to guarantee that all participants obtain a specific message”.

Goals

arjuna
middleware for reliability

- Provide a general framework for coordination protocols
  - Existing implementations to be plugged in
  - New implementations can be supported
    - Defines coordinator and participant relationships
- Work with WS-Context
  - Augment context
  - ➢ Coordination Context
- Scope of activity becomes scope of coordination boundary

Context type

```
<xs:complexType name="ContextType">
  <xs:complexContent>
    <xs:extension base="wsctx:ContextType">
      <xs:sequence>
      <xs:element name="protocol-reference" type="tns:ProtocolReferenceType"/>
      <xs:element name="coordinator-reference" type="tns:CoordinatorReferenceType"
        maxOccurs="unbounded"/>
      <xs:any namespace="##any" processContents="lax" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

WS-CF
Main components

middleware for reliability

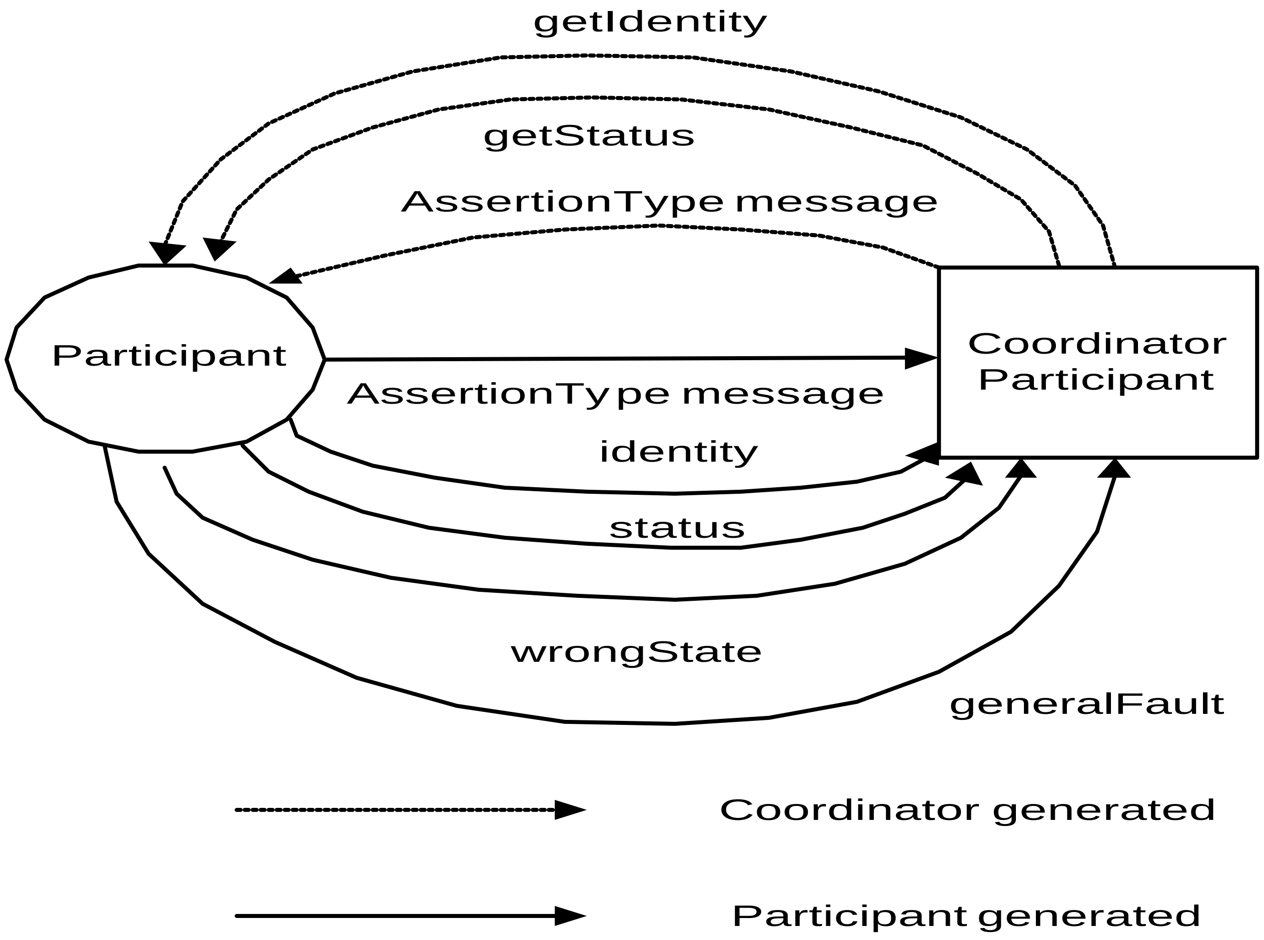
- *Coordinator*:
  - enable registration of participants triggered at *coordination points*.
- *Participant*:
  - The operation or operations that are performed as part of coordination sequence processing
  - A *Coordination Service*: Defines the behavior for a specific coordination model.
- *Coordination Service:*
  - provides a processing pattern that is used for outcome processing.
  - For example
    - ACID (prepare, commit, rollback)
    - Sagas
    - Real-time transactions
    - …

Coordination protocol

- Defined by message interactions between Coordinator and participants
  - Coordinator-to-participant
    - coordinator sends a protocol message to the participant and will eventually get a response.
    - Coordination status and identity
  - Participant-to-coordinator
    - a participant may autonomously communicate protocol messages to the coordinator.
      - Works in terms of AssertionTypes
- WS-CF protocol neutral
- Protocols Identified by URI

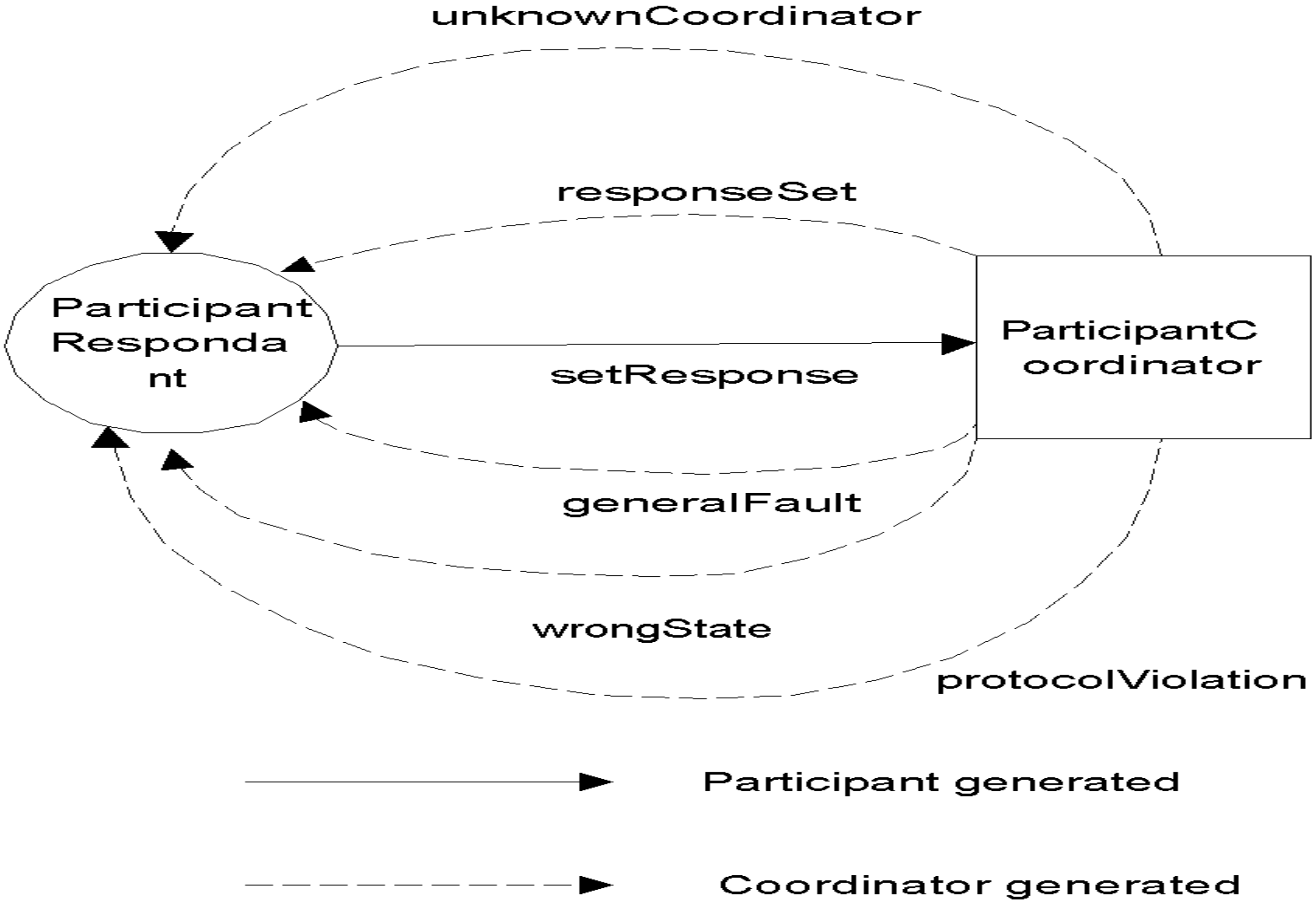
Coordinator-to-participant

arjuna middleware for reliability



Participant-to-coordinator

arjuna
middleware for reliability



AssertionType

arjuna middleware for reliability

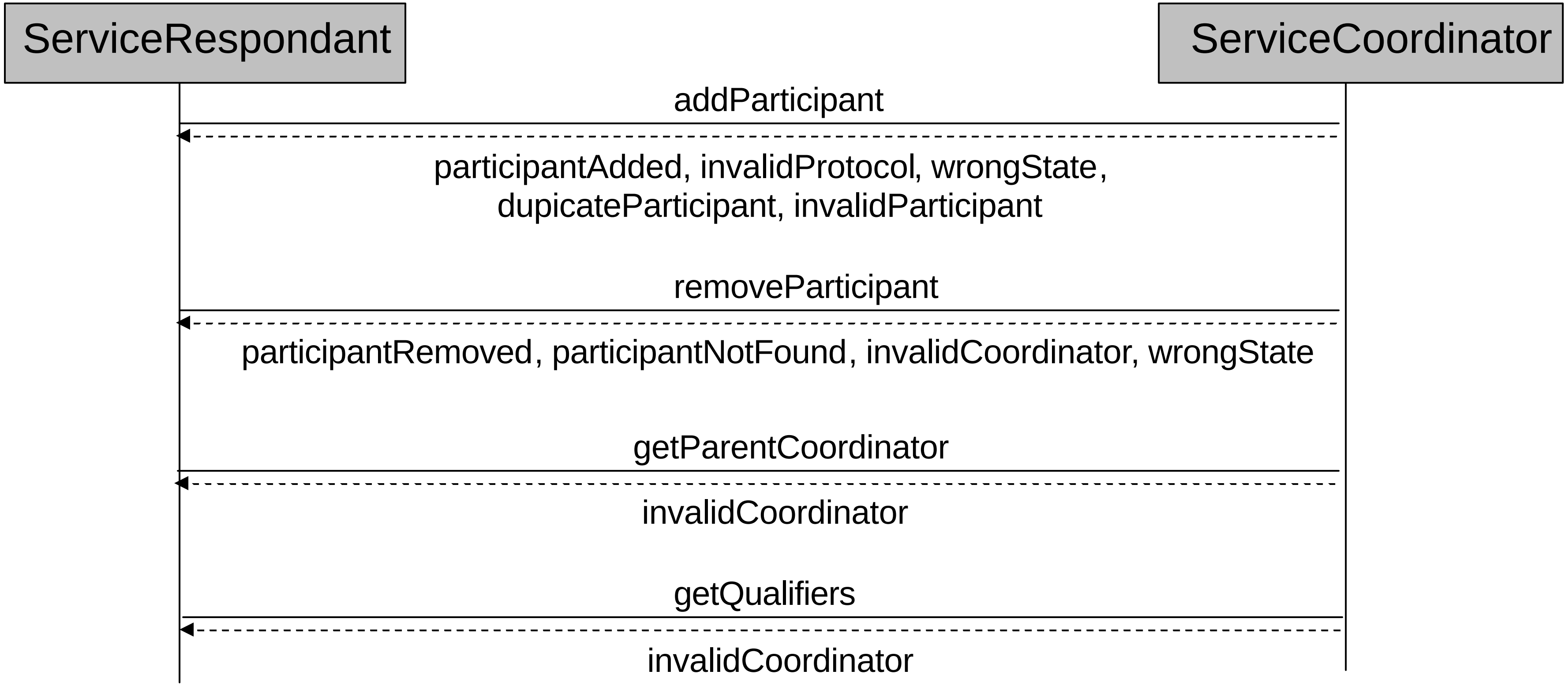
- “Base class” for all coordinator-to-participant message interactions
  - Requests and responses
- All protocol specific messages enhance this type
- One service (participant or coordinator) can accept multiple protocols

Qualifiers

- Runtime protocol extensibility option
- Typically in enlist/delist
  - For coordinator/participant information
  - E.g., will cancel in 24 hours

Service-to-coordinator interactions

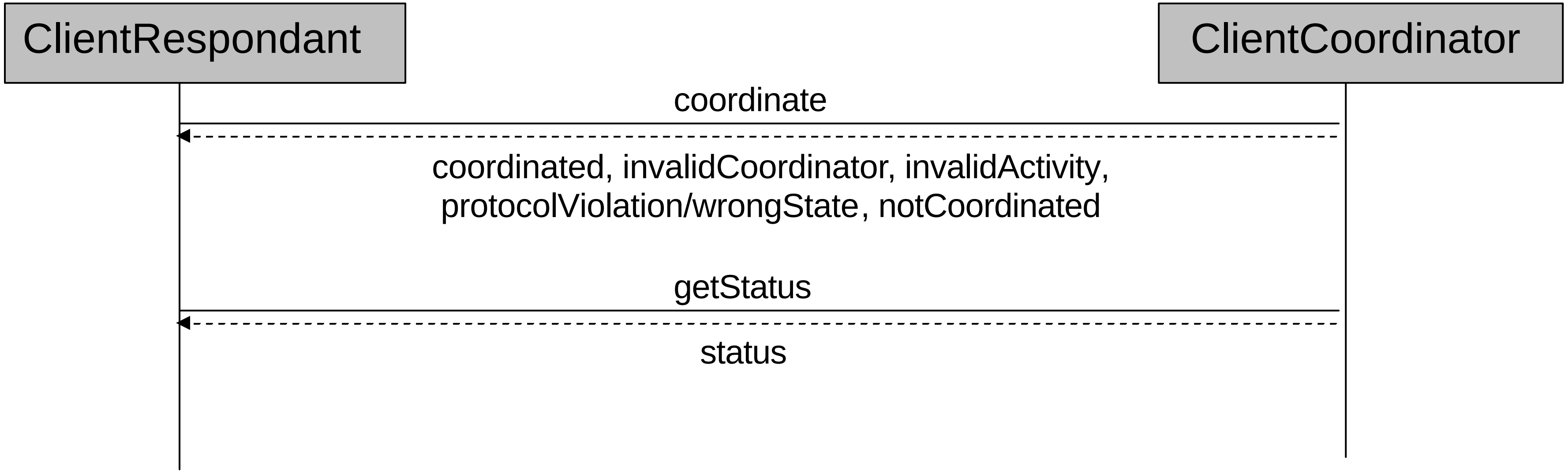
- define how a service may enlist or delist a participant with the coordinator



ServiceRespondant
ServiceCoordinator
addParticipant
participantAdded, invalidProtocol, wrongState, dupicateParticipant, invalidParticipant
removeParticipant
participantRemoved, participantNotFound, invalidCoordinator, wrongState
getParentCoordinator
invalidCoordinator
getQualifiers
invalidCoordinator

Client-to-coordinator interactions

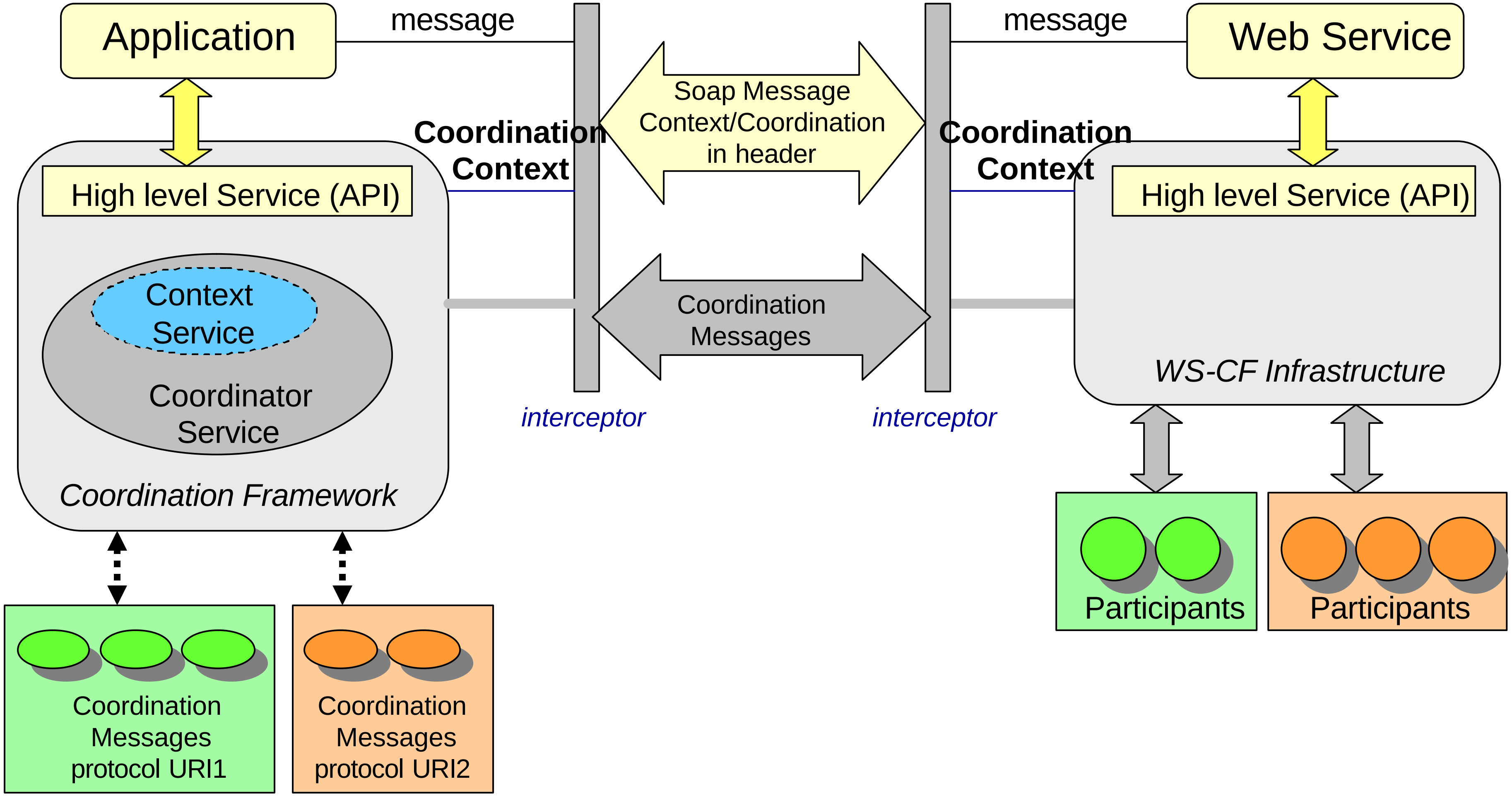
- Define how a client can obtain the status of the coordinator or ask it to perform coordination.



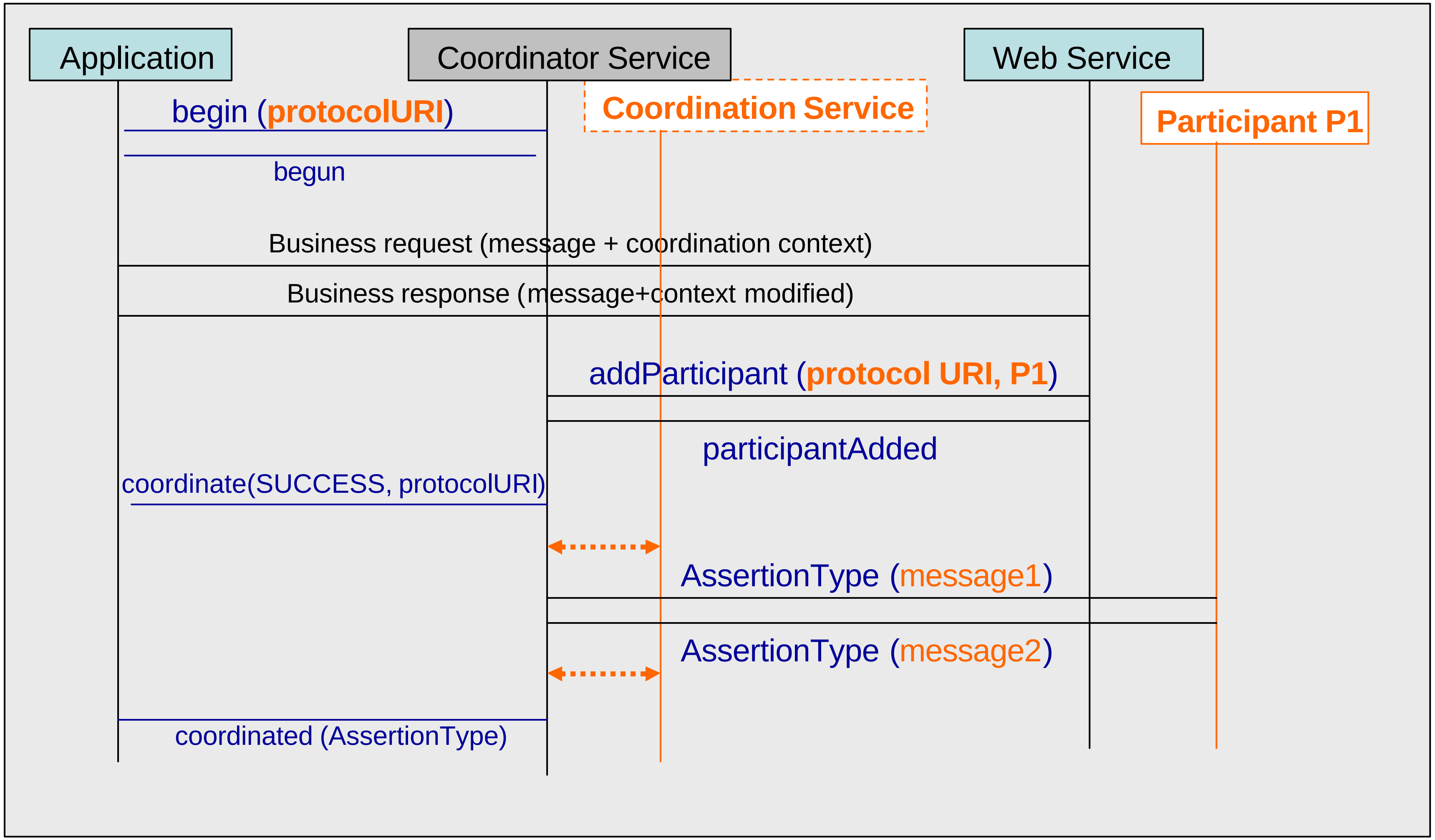
ClientRespondant
ClientCoordinator
coordinate
coordinated, invalidCoordinator, invalidActivity,
protocolViolation/wrongState, notCoordinated
getStatus
status

Architectural Overview

arjuna middleware for reliability



Interactions Overview


Application
Coordinator Service
Web Service
Coordination Service
Participant P1
begin (protocolURI)
begun
Business request (message + coordination context)
Business response (message+context modified)
addParticipant (protocol URI, P1)
participantAdded
coordinate(SUCCESS, protocolURI)
AssertionType (message1)
AssertionType (message2)
coordinated (AssertionType)

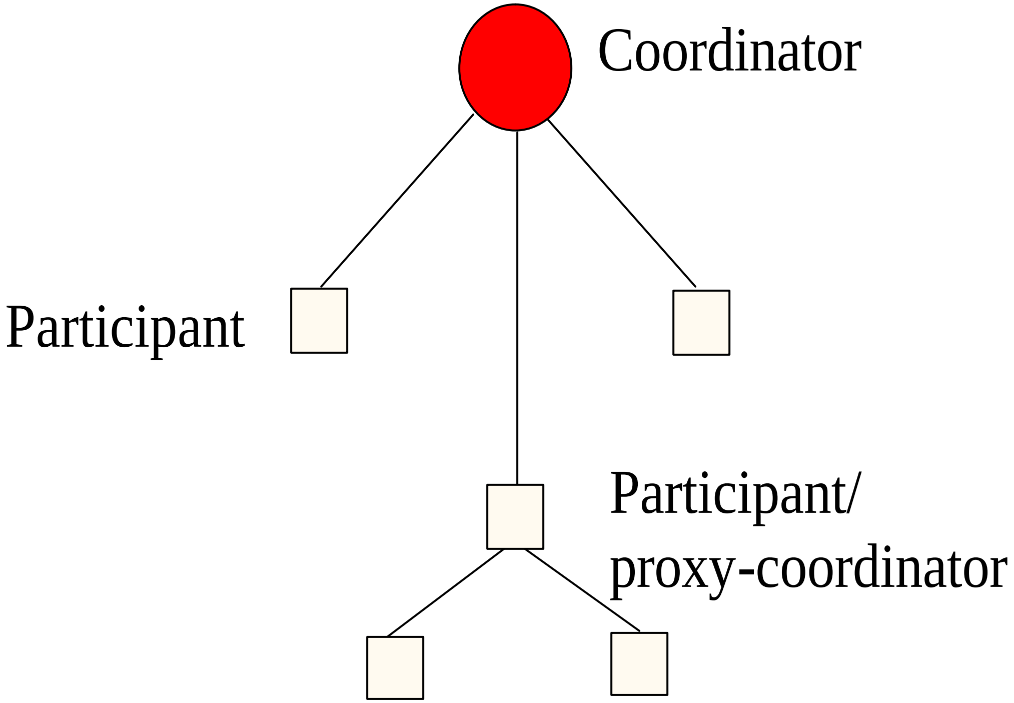
Interposition

arjuna middleware for reliability

- Important for security and performance reasons
  - Part of most distributed transaction protocols
- Subordinate coordinator
  - Participant as far as coordinator is concerned
  - Coordinator as far as participant is concerned
- Supported by WS-CF
  - Not mandated

Example

arjuna middleware for reliability



Recovery

- Distributed application federated into natural recovery (admin) domains
  - Can’t mandate one specific recovery mechanism
    - Very application specific anyway
  - Have to allow administrative domains autonomy
- Therefore, support but not mandate

Recovery support

arjuna middleware for reliability

- RecoveryCoordinator
  - Drives recovery on behalf of participant
  - Participants may not be able to recover at same URI
    - Machine crash, domain migration, …
- Coordinator can replace one endpoint with another to continue protocol

WS-Transaction Management

arjuna

middleware for reliability

WS-TXM

arjuna middleware for reliability

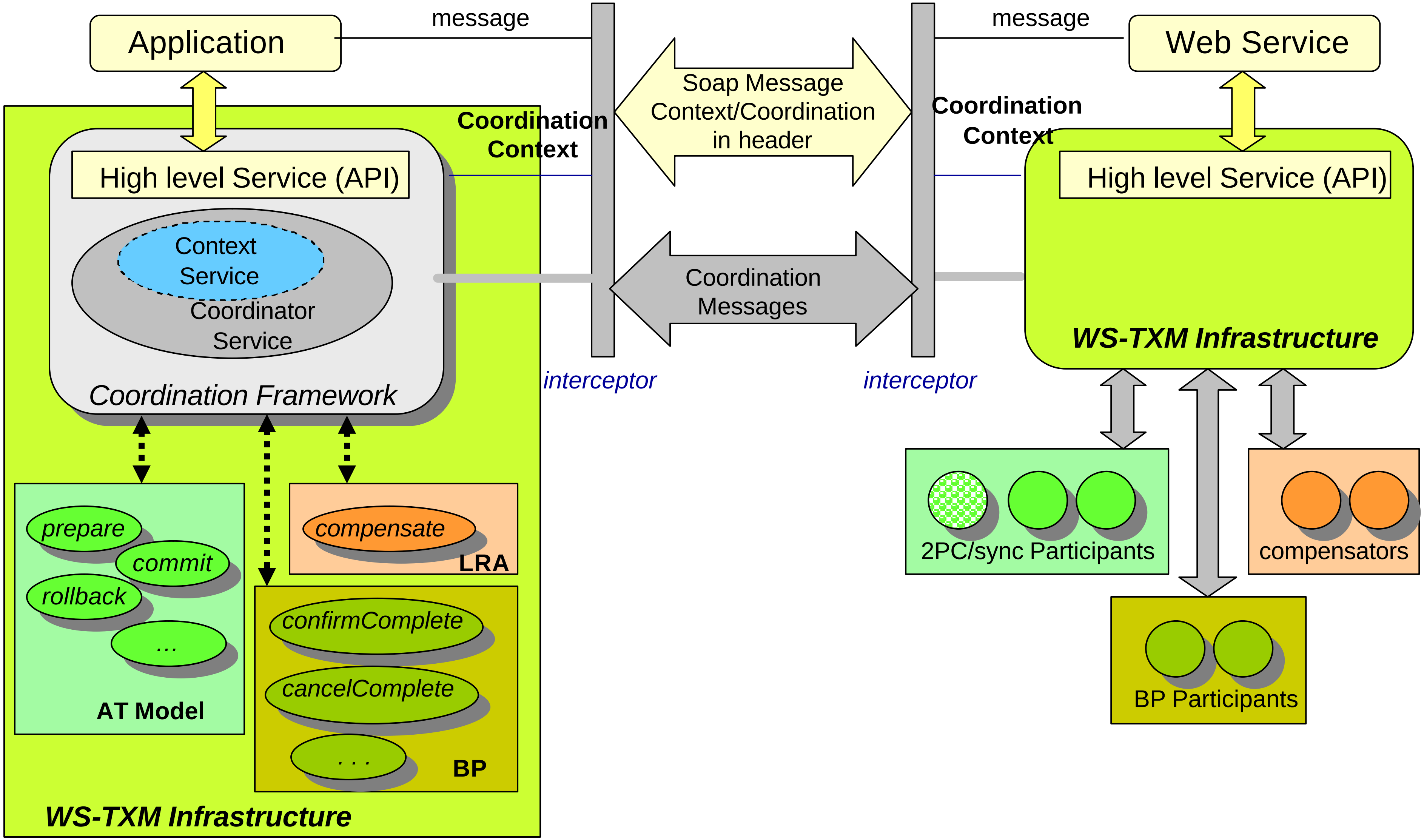
- Goals
  - Support range of use cases
  - “One-size does not fit all”
  - Therefore a single protocol cannot cope with all requirements
  - All requirements aren’t “two-phase”
- Builds on WS-CF and WS-Context
  - Define specific coordinators and participants
  - Augment context

Defines

- Three transaction models
  - ACID transaction
    - For interoperability and high-cost services where ACID transactions are a requirement
  - Long running action
    - Loosely coupled, long duration work that uses compensations
  - Business process
    - For treating all steps in an automated business process as part of a single logical transaction

Architectural Overview

arjuna
middleware for reliability



ACID Transaction (AT)

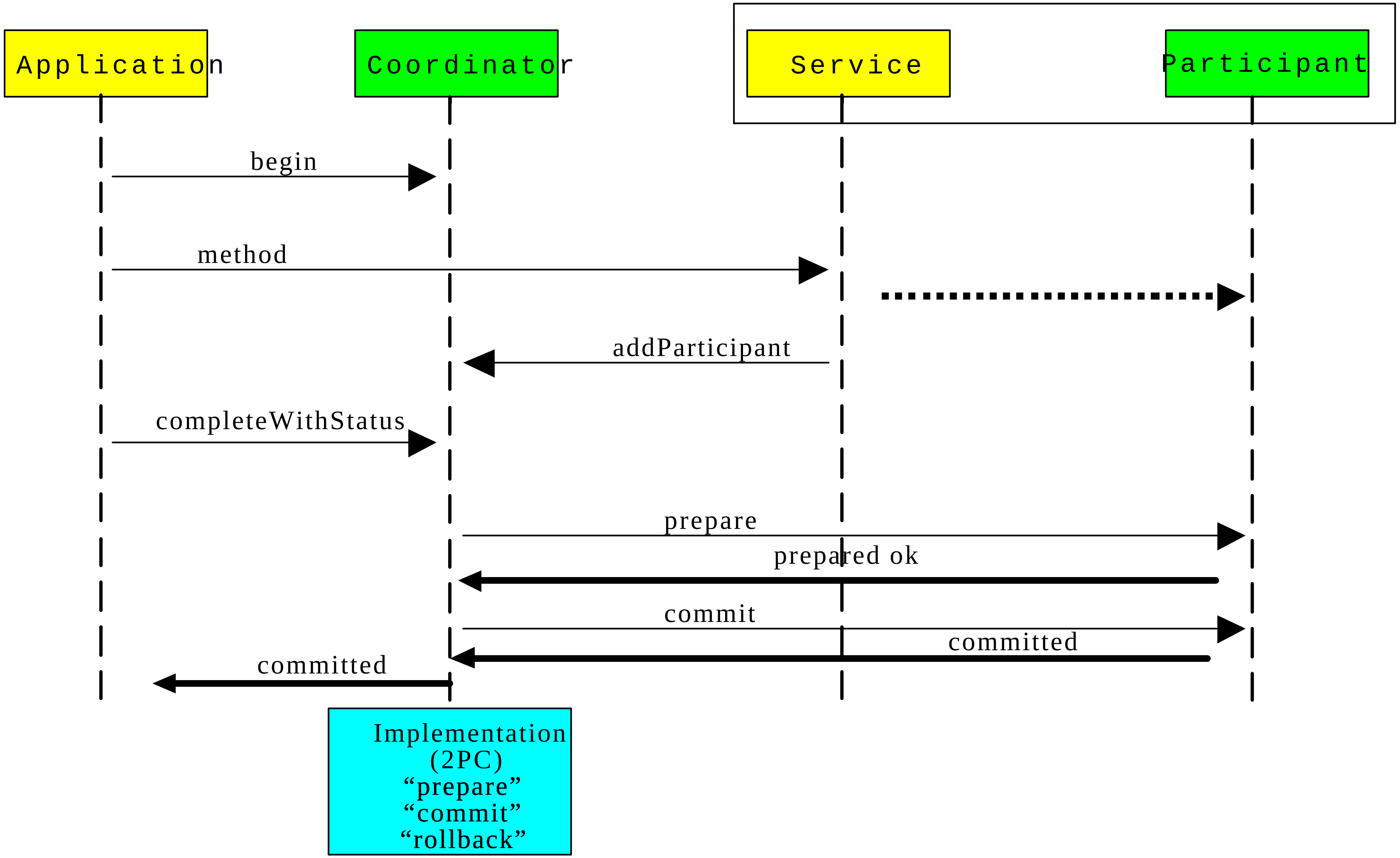
- Coordination-type URI
  - *http://www.webservicestransactions.org/wsdl/wstxm/tx-acid/2003/03*
- Traditional *ACID transaction* with two sub-protocols (different URIs)
  - Two-phase commit
    - *http://www.webservicestransactions.org/wsdl/wsTXM/tx-acid/2pc/2003/03*
  - Synchronizations
    - *http://www.webservicestransactions.org/wsdl/wsTXM/tx-acid/sync/2003/03*
- Interoperability across different vendor implementations
  - *removeParticipant* illegal
    - wrongState returned by coordinator
  - *coordinate* cannot be used
    - Bind the scope of activity to the scope of transaction

Model (AT)

arjuna middleware for reliability

- ACID semantics explicitly required
- Presumed rollback
- One-phase optimization
- Read-only optimization
- Heuristics

Example (AT)


Application
Coordinator
Service
Participant
begin
method
addParticipant
completeWithStatus
prepare
prepared ok
commit
committed
committed
Implementation
(2PC)
“prepare”
“commit”
“rollback”

2PC protocol messages (AT)

- Usual for two-phase commit
  - prepare
    - voteReadonly, voteCommit, voteRollback
    - And heuristics
  - commit
    - Heuristics
  - rollback
    - Heuristics

Synchronization messages (AT)

- beforeCompletion
  - Runs before two-phase commit begins
- afterCompletion
  - Runs after two-phase protocol

Long running action model (LRA)

arjuna middleware for reliability

- Protocol URI
  - http://www.webservicestransactions.org /wstxm/tx-lra/2003/03
- Specifically for long duration interactions
- ACID transactions are not appropriate
  - Can’t lock resources for the duration
  - No assumed trust relationships
- Compensation actions used
  - Forward work to return the business state to consistency
    - E.g., credit your credit card and give you back interest payments

Relationship to context and coordination

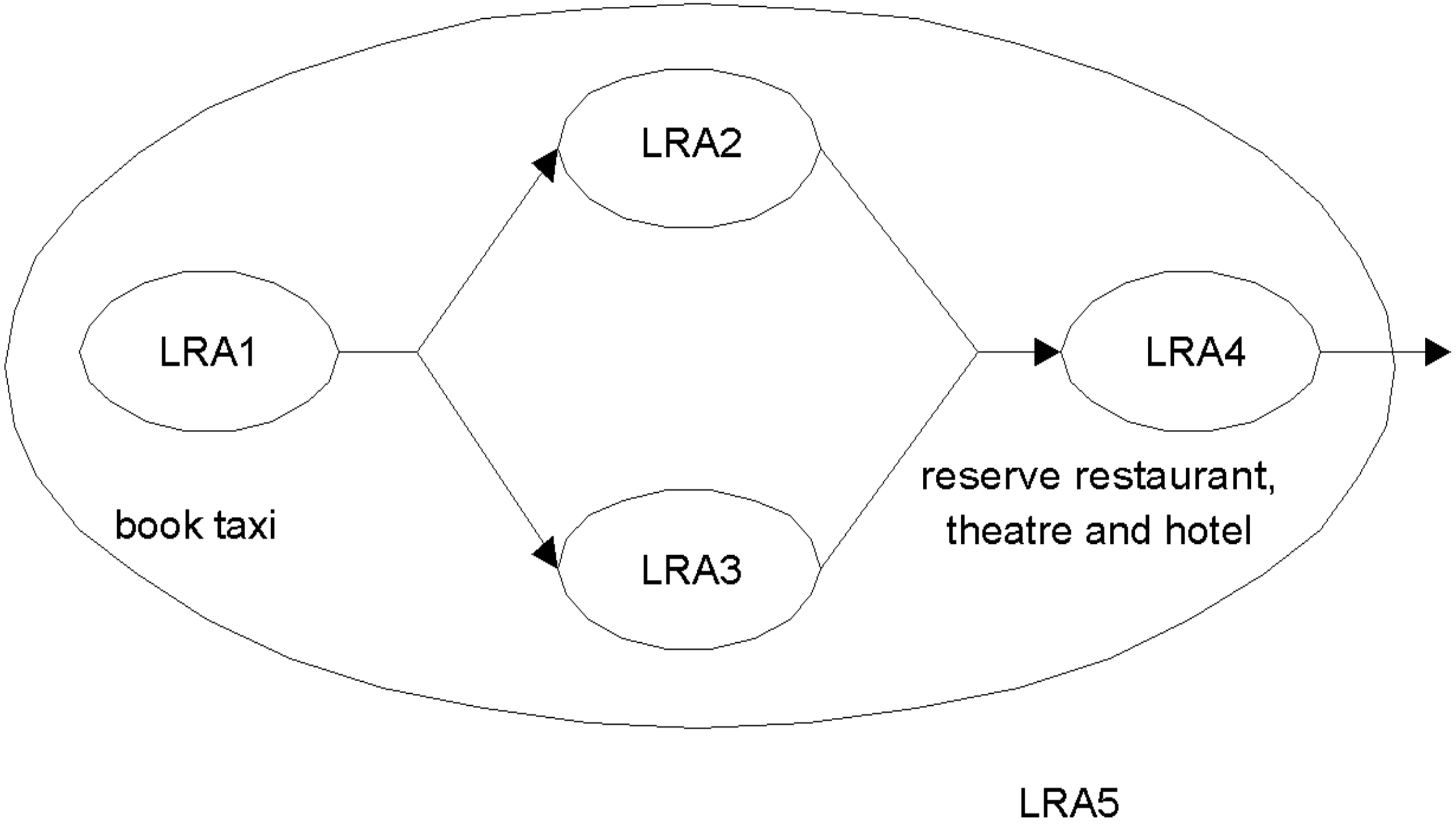
- Activity becomes the scope of business interactions
  - Travel agent, computer construction, etc.
- How services do work is not important
  - Back-end implementation choice
- If work can be compensated then compensation is bound to the activity
  - Non-atomic behaviour
- Activities can be nested

Activities and compensators

arjuna middleware for reliability

- Each activity is a unit of compensatable work
- Work performed must remain compensatable for duration of activity
  - TimeLimit qualifier
- Nested activities imply nested compensators
  - Could be different compensator from child to parent

Travel agent example



Context

```
<xs:element name="context">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="wstxm:ContextType">
        <xs:sequence>
          <xs:element name="lra-id" type="xs:anyURI"/>
          <xs:element name="coordinator-hierarchy">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="coordinator-location" type="xs:anyURI" minOccurs="0"
                  maxOccurs="unbounded"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

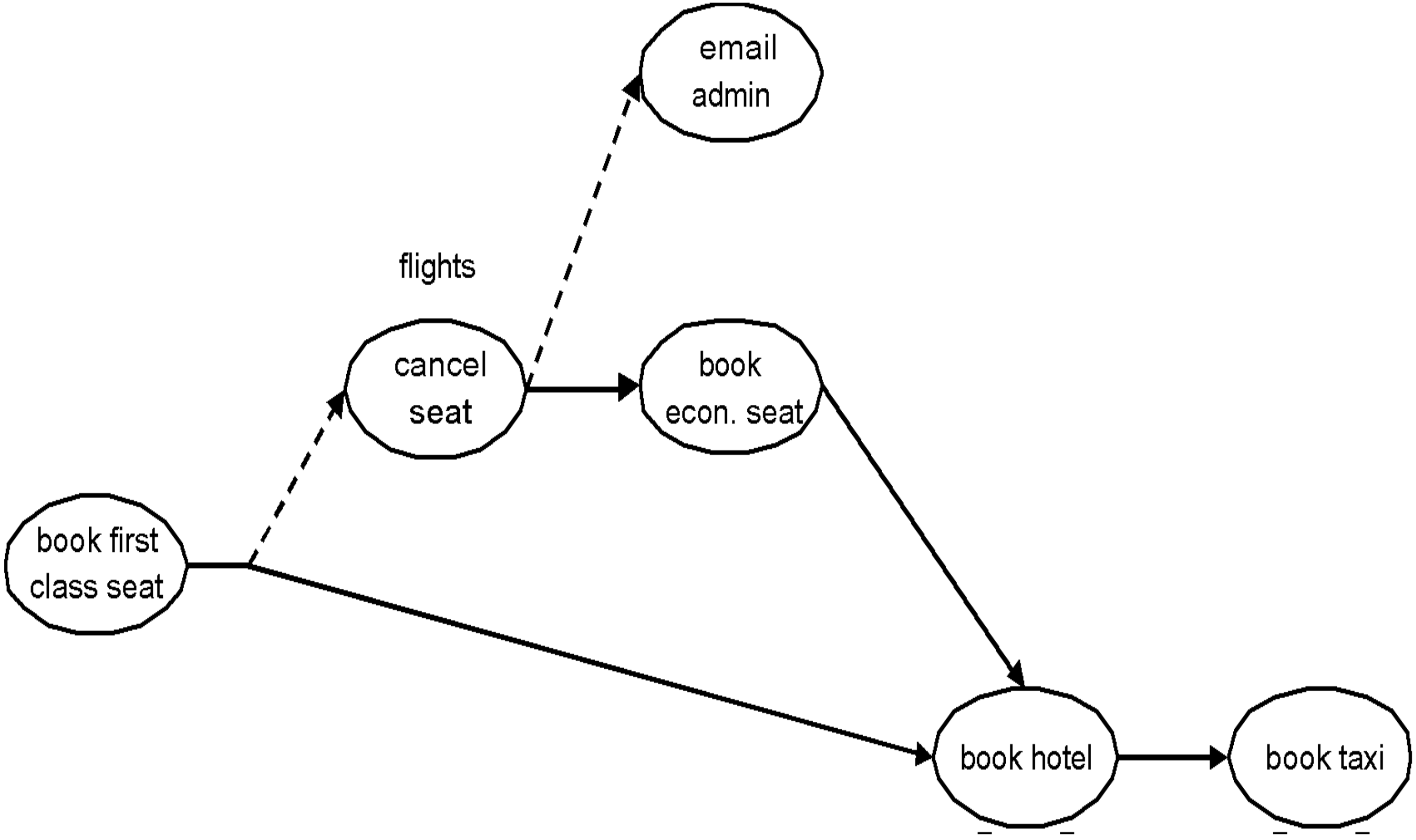
To compensate or not?

arjuna
middleware for reliability

- Some services may not be able to compensate
- The user defines whether or not this is important
  - MustUnderstand
- Must be an explicit choice
  - Adverse consequences otherwise

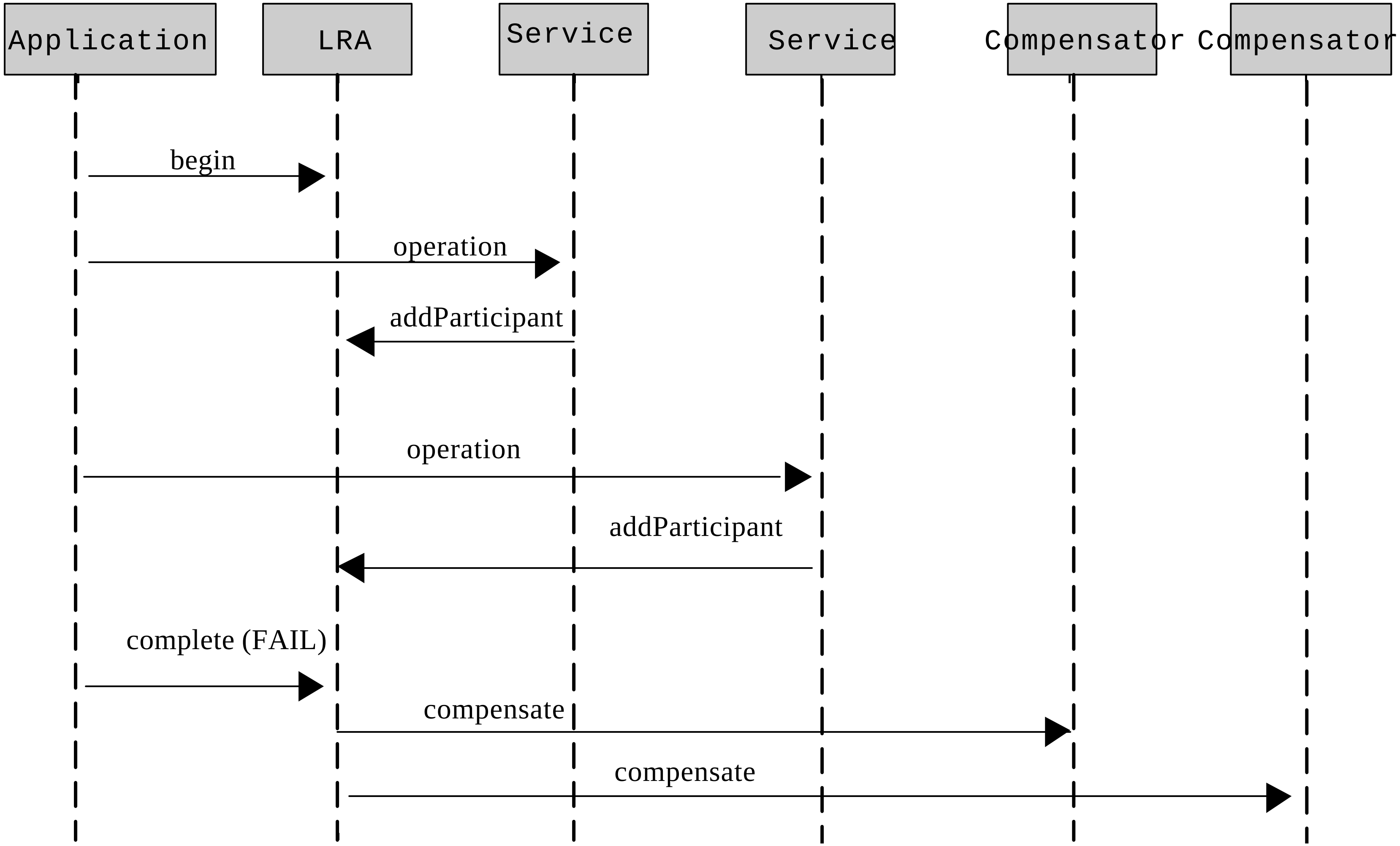
Compensators

arjuna
middleware for reliability



Example

arjuna
middleware for reliability



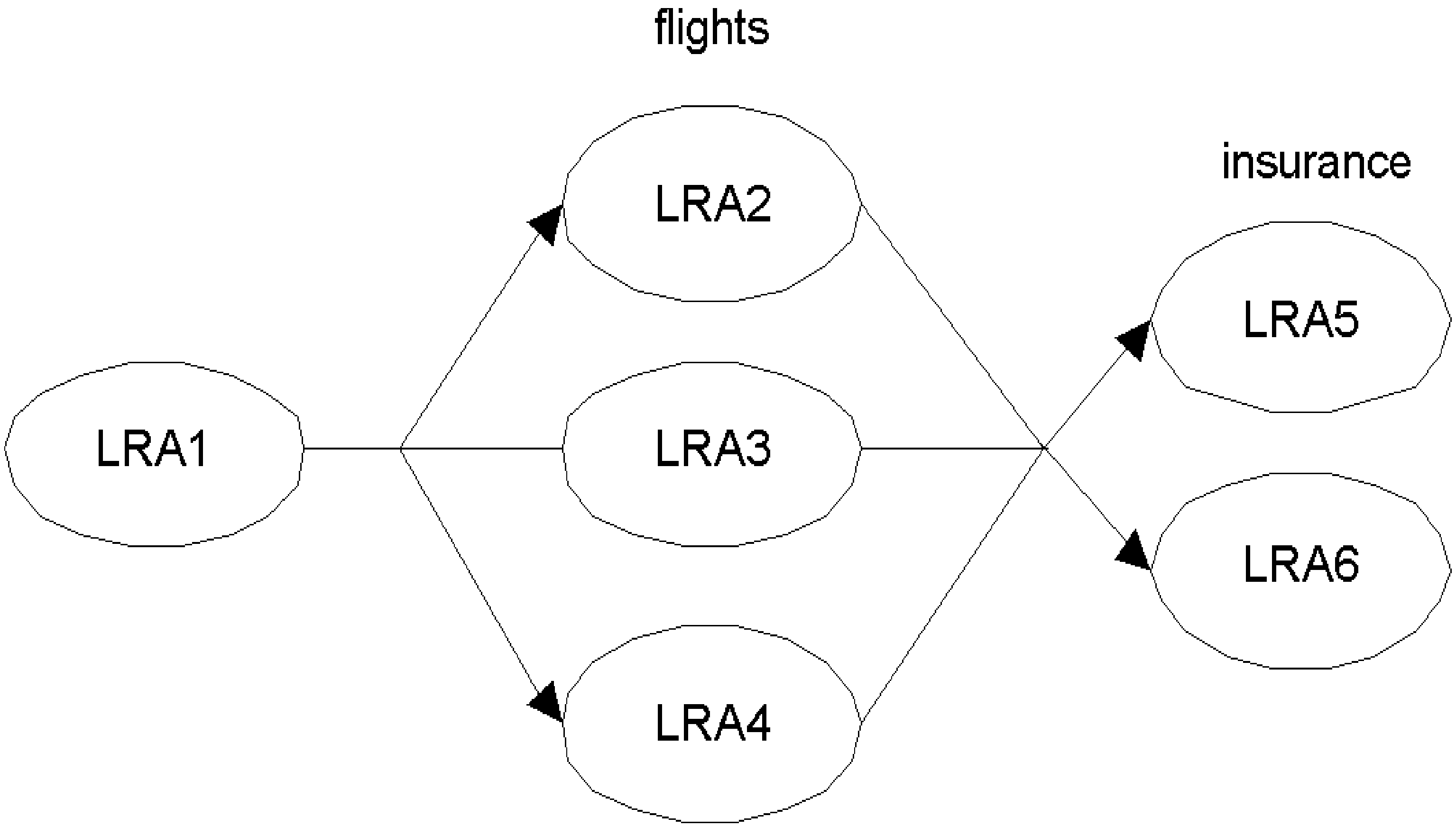
Concurrent LRAs

arjuna middleware for reliability

- Activities can be concurrent
  - Therefore, LRAs can be too
- Allows selection of work within overall activity
  - E.g., choosing the cheapest flight

Selection of work

arjuna
middleware for reliability



Business process model

arjuna middleware for reliability

- Aimed at *long running* interactions that span different domains *and* models
  - Workflow
  - Messaging
  - Traditional ACID transactions
- Federated systems that can’t/won’t expose back-end implementations

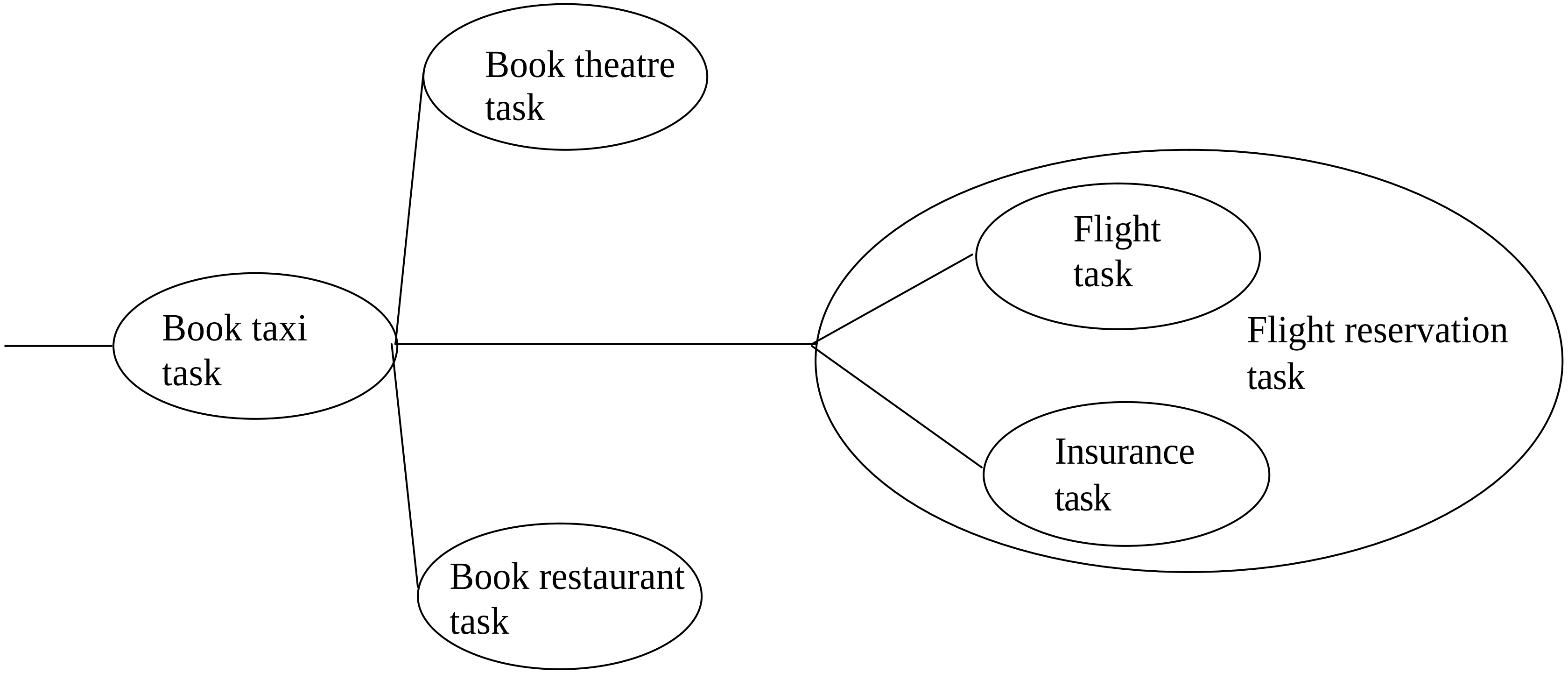
Domains

arjuna
middleware for reliability

- All parties reside within *business domains*
  - Recursive structure is allowed
  - May represent a different transaction model
- Business process is split into *business tasks*
  - Execute within domains
  - Compensatable units of work
    - Forward compensation during activity is allowed
      - Keep business process making forward progress

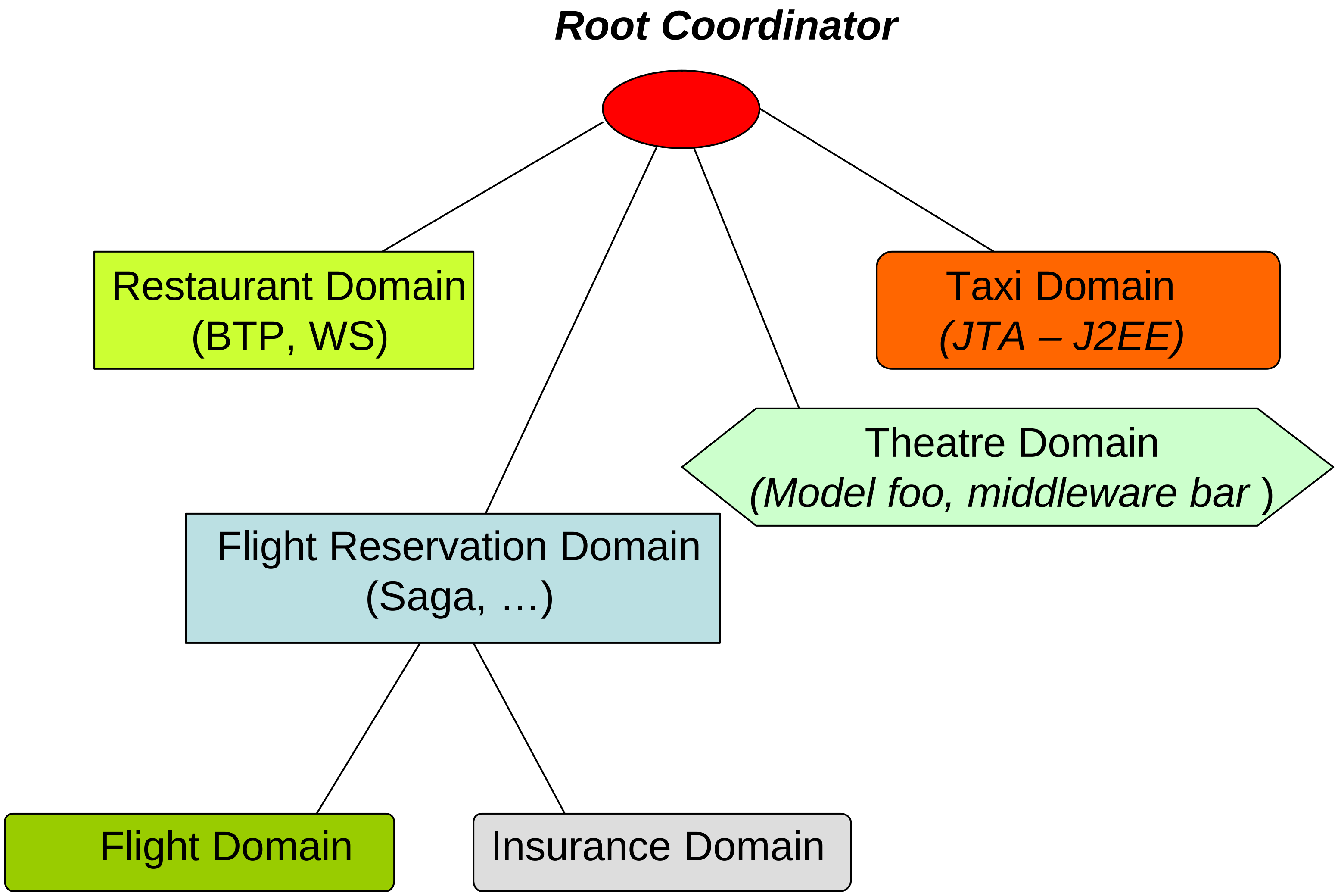
Travel agent

arjuna
middleware for reliability



Business Process interposition

arjuna middleware for reliability



Model

- Supports synchronous and asynchronous interactions
  - Users can submit work and call back later
  - Or interact synchronously (traditionally)
- Business Process manager
- Optimistic rather than pessimistic
  - Assumes failures are rare and can be handled offline if necessary

How does it work?

- Each domain is exposed as a subordinate coordinator
  - Responsible for mapping incoming BP requests to domain specific protocol
- Protocol messages
  - checkpoint, confirm, cancel, restart, workStatus

checkpoint/restart

- Application driven
  - E.g., via *coordinate* message
- Checkpoints are created across the domains
  - Uniquely identified
- Domains can then roll back to specific checkpoint

workstatus

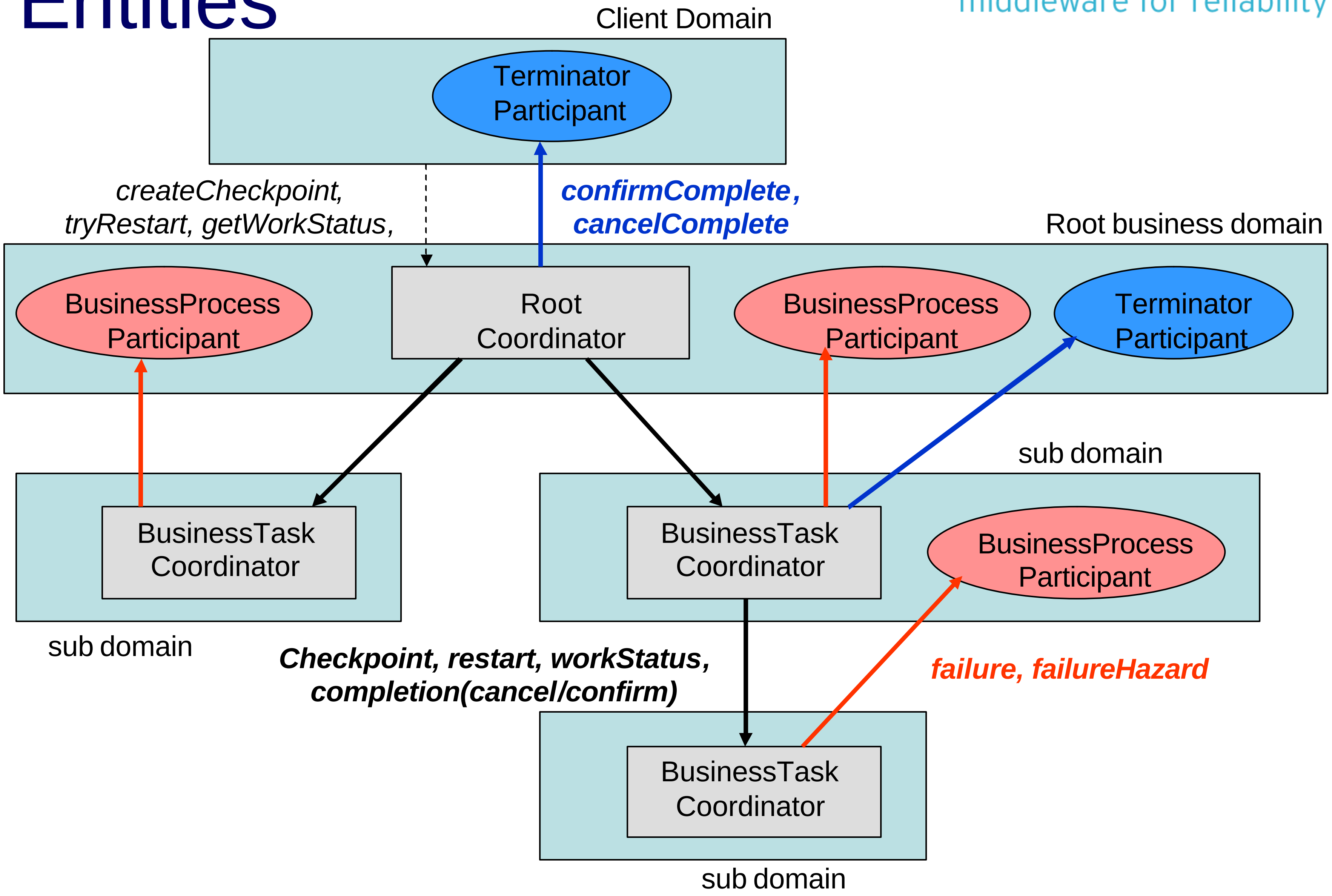
arjuna
middleware for reliability

- Domain calls back to coordinator to inform it of final status
- Or application can enquire
  - WorkCancelled
  - WorkCompleted
  - WorkProcessing

confirm/cancel

- BP termination protocol messages
- Map to success/failure of activity
- Because long-running, heuristics may occur
  - Mixed responses from domains
  - Sufficient information to allow administrative handling

Business Process Entities



arjuna
middleware for reliability