Enterprise Undo

Mark Little, JBoss
Jonathan Halliday, JBoss
Maciej Machulak, Newcastle University

Java

Sun microsystems

Learn how to enable your business logic to participate in extended transactions by using annotations

GOAL

Agenda

- Transactions 101
- Development of extended transactions
- Business Activity Framework
- Future Work
- Summary
- Questions and Discussion

Agenda

- Transactions 101
- Development of extended transactions
- Business Activity Framework
- Future Work
- Summary
- Questions and Discussion

Transactions 101

- Transactions are a way of structuring application execution
- Client applications manage transaction boundary demarcation
  - begin, commit, rollback
- Business logic runs within the scope (context) of the transaction
- Transactions may run for a short time or a long time
  - Database updates
  - Business process workflows

Atomic transactions

- Scoping mechanism that provides “all-or-nothing” semantics
- ACID properties: Atomic, Consistent, Isolated, Durable
- Use locking of resources
- Rollback to original state on failure
- Two-phase commit protocol for consistent outcome determination
- Well standardized (JTA, XA), Widely used (Enterprise JavaBeans™ (EJB™) technology, databases)

Atomic transaction - Client Application

```
javax.transaction.UserTransaction transaction = ...

        // JTA call, associate tx context to thread
        transaction.begin();

        try
        {
                hotelService.bookRoom(1);
                hotelService.bookRoom(2);

                // run 2PC, clear thread tx context
                transaction.commit();
        }
        catch (Exception ex)
        {
                transaction.rollback();
        }
```

Atomic transaction – EJB 3.0 technology

```
@Stateless
public class HotelImpl implements Hotel
{
    @TransactionAttribute(REQUIRED)
    public int bookRoom(int roomNumber)
    {
        // method implementation:
        // pure business logic
        // no transaction logic
    }

    // ...
}
```

Extended Transactions

- A way of structuring business processes that may
  - Span multiple applications or organizational boundaries
  - Take a long time
- ACID will not work
  - Locking would limit throughput
  - Partial undo is desirable for failure isolation
- Extended transaction models are required
  - Non is as well supported as traditional transactions
  - Many use Roll forward compensation

WS-Business Activity

- Defines an extended transaction model for the Web Services environment - **Business Activity** (BA)
- Designed specifically for **long-duration interactions**
- Consensus within a single BA is achieved through **agreement protocol**
- Services use **compensation** - forward recovery
- **Compensation actions** are **application-specific** and **must be explicitly provided** by business application programmers

Extended transaction – Java™ Platform, Enterprise Edition Server side code

```
public class HotelImpl implements Hotel
{
    @TransactionAttribute(REQUIRED)
    public int bookRoom(int roomNumber)
    {
        // mixed business logic and transaction logic
        // elisting Participants,
        // maintaining multiple persistent
        //     copies of state
        // ...
        // (actual code censored to protect your sanity)
    }

    // ...
}
```

Agenda

- Transactions 101
- Development of extended transactions
- Business Activity Framework
- Future Work
- Summary
- Questions and Discussion

Problem statement

- Business programmers should be able to declare desired transactional behavior using annotations (like EJB 3.0 technology)
- The transaction plumbing should be provided automatically
  - Difficult issues such as serialization of state, concurrency control, locking and versioning of data, logging, rollback and crash recovery handling
  - There's no need to write code for participants (resources) that respond to transaction protocol messages (prepare, compensate, close)

Problem statement (2)

- Business programmers should not need transaction expertise
- Decouple transaction code and business logic
  - but there is unlikely to be a silver bullet...
- Compensation actions are application-specific and thus need to be provided explicitly by business logic developers
- Nevertheless, we can still find ways to help make life easier for business programmers

Agenda

- Transactions 101
- Development of extended transactions
- Business Activity Framework
- Future Work
- Summary
- Questions and Discussion

Objectives

- Make it easier for business programmers to use extended transactions
- Separate the transaction related code from the application's business logic
- Developers should **declaratively** specify transactional requirements of their applications
- All necessary transaction processing should be provided basing upon those requirements in a **transparent way** and should be applied at **runtime**

Business Activity Framework

- **Annotations** used to describe transactional requirements
  - **Units of work** can be **related** to their **compensation actions** with few simple declarative statements
- **Transparent transaction management**
  - Automated execution of compensation actions
- **Flexible management of data** required throughout a single transaction
- **Business-logic separation:** Framework provides implementation of the transaction participant

Business Activity Framework (2)

- The initial framework implementation is focused on allowing Web Services to take part in WS-BA driven extended transactions
- The concepts are generic and can be applied to other extended transactions protocols

Client application

```
com.arjuna.mw.wst.UserBusinessActivity businessActivity =
                                                        ...

businessActivity.begin();
try
{
        hotelService.bookRoom(1);
        hotelService.bookRoom(2);
        businessActivity.close();
}
catch (Exception ex)
{
        businessActivity.cancel();
}
```

Server side

```
@BAMethod
@BACompensatedBy("cancelRoom")
public int bookRoom(int roomNumber)
{
    // ... method implementation: business logic only
    // ... return reservation number
}

public boolean cancelRoom(int reservationNumber)
{
    // ... method implementation: business logic only
}
```

Server side

```
@BAMethod
@BACompensatedBy(value="cancelRoom",type=DataMatch.CUSTOM)
@BAResult("reservation")
public int bookRoom(@BAParam("user") String userName,
                    @BAParam("password") String password,
                    int roomNumber)
{
    // ... method implementation
}

public boolean cancelRoom(@BAParam("user") String userName
            @BAParam("password") String password,
            @BAParam("reservation") int reservationNumber)
{
    // ... method implementation
}
```

Server side

```
@BADataManagement
DataManager dm;

@BAMethod
@BACompensatedBy(value="cancelRoom",type=DataMatch.CUSTOM)
@BAResult("reservation")
public int bookRoom(@BAParam("user") String userName,
                    @BAParam("password") String password,
                    int roomNumber)
{
    // ...
    dm.put("refund",refundValue);
    // ...
}
```

Server side

```
public boolean cancelRoom(@BAParam("user") String userName
            @BAParam("password") String password,
            @BAParam("reservation") int reservationNumber)
{
    // ...
    Integer refund = (Integer) dm.get("refund");
    if (refund != null)
    {
        // ...
    }
    // ...
}
```

Server side

```
@BAMethod(agreement=AgreementType.COORDINATOR_COMPLETION)
@BACompletedBy(value="getBooking",type=DataMatch.CUSTOM)
@BACompensatedBy(value="cancelRoom",type=DataMatch.CUSTOM)
@BAResult("reservation")
public int bookRoom(@BAParam("user") String userName,
                    @BAParam("password") String password,
                    int roomNumber)
{
    // Retrieve existing array of booked rooms. If null,
    // create new one. Add room of "roomNumber" to array.
    // Store the array with DataManager
    dm.put("0",rooms);
    // ...
}
```

Server side - using 3rd party service

```
@BACompletedBy(
    value = "getBooking",
    type = DataMatch.CUSTOM,
    mode = ExecutionMode.WS_PORT,
    serviceName = "RemoteHotelService",
    wsPort = RemoteHotel.class,
    wsdl = "http://host/RemoteHotel/RemoteHotelImpl?wsdl",
    namespace = "http://example.com"
)
```

3rd party server side

```
@WebMethod
public boolean getBooking(Integer[] roomNumbers)
{
    // ...
}
```

Business Activity Framework - API

- Clean and intuitive **annotation-based API** - keeps with the current trend in enterprise environments
  - **@BAService**
  - **@BAMethod**
  - **@BACompensatedBy**
  - **@BACompletedBy**
  - **@BAParam**
  - **@BAResult**
  - **@BADataManagement**

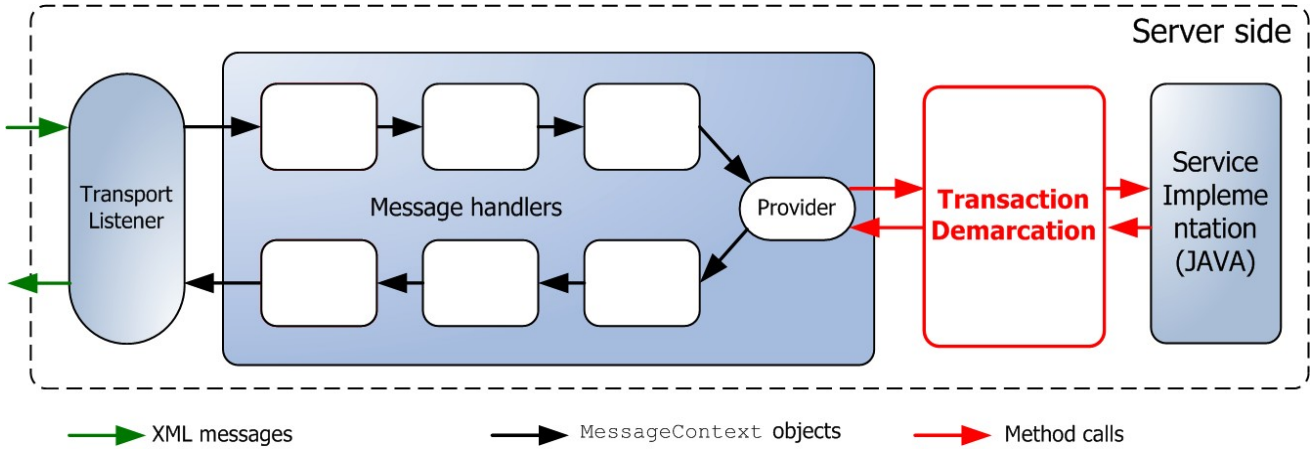
Business Activity Framework - API (2)

- Leverages programmer familiarity with EJB 3.0 technology and Java Specification Request (JSR) 181 - reduces the learning curve for new users, adaptable to other environments than Web Services
- All annotations have reasonable default values - programmers do not need to explicitly provide all configuration options
- Flexible **Data Manager** for storage and retrieval of transaction business data between calls
  - Map semantics: put / get
  - Crash tolerant

Business Activity Framework Demo

Business Activity Framework - design

- Transaction Interceptors using AOP
- Not SOAP (or even Web Service) specific
- Annotations on business logic control transactional behavior



Business Activity Framework - design (2)

- Business method calls are transparently associated associated with participants provided by the framework
- Participants are responsible for responding to transaction control messages e.g. ***complete()***, ***close()***, ***compensate()***
- Each participant stores data related to method calls for later use e.g. Compensation
- Automatic reverse ordering of compensation actions

BA Framework - Developing Web Services

- Business Activity Framework consists of two modules:
  - Compile time client library with API
  - Server side AOP, resides in the container or deploys with the app
- Transactional Web Services use a SOAP header processor to manage transaction context
- AOP may be compile time (offline instrumentations) or runtime

Agenda

- Transactions 101
- Development of extended transactions
- Business Activity Framework
- Future Work
- Summary
- Questions and Discussion

Future Work

- Support for other extended transaction protocols and transport, to move beyond WS-BA & Web services.
- Gather feedback from the community about their requirements
- Search for generic solutions that can apply to any extended transaction model
  - Re-use annotated business logic in different transaction types.
- Standardize (through JSR?) once we have a model that is widely applicable

- Meanwhile you can download and use it today...

Agenda

- Transactions 101
- Development of enterprise transactions
- Business Activity Framework
- Future Work
- Summary
- Questions and Discussion

Summary

- Extended transaction models are highly useful in enterprise integration scenarios
- More development-friendly support for business programmers is required
- The Business Activity Framework uses annotation driven approach to compensations (forward recovery)
- Existing business logic can be reused (like JSR 181)

For More Information

- Business Activity Framework website
  - http://labs.jboss.com/jbosstm
    http://labs.jboss.com/jbosstm/baframework
- Java API for XML Transactions (JSR-156)
  - http://jcp.org/en/jsr/detail?id=156
- Email
  - jonathan.halliday@redhat.com
  - m.p.machulak@newcastle.ac.uk

THANK YOU

Mark Little, JBoss
Jonathan Halliday, JBoss
Maciej Machulak, Newcastle University

Reserve slides follow

BA Framework - Developing Web Services

```
...
<handler-chain>
<protocol-bindings>##SOAP11_HTTP</protocol-bindings>
<handler>
<handler-name>
WebServicesTxContextHandler
</handler-name>
<handler-class>
com.arjuna.mw.wst.service.JaxWSHeaderContextProcessor
</handler-class>
</handler>
</handler-chain>
...
```

BA Framework - Developing Web Services

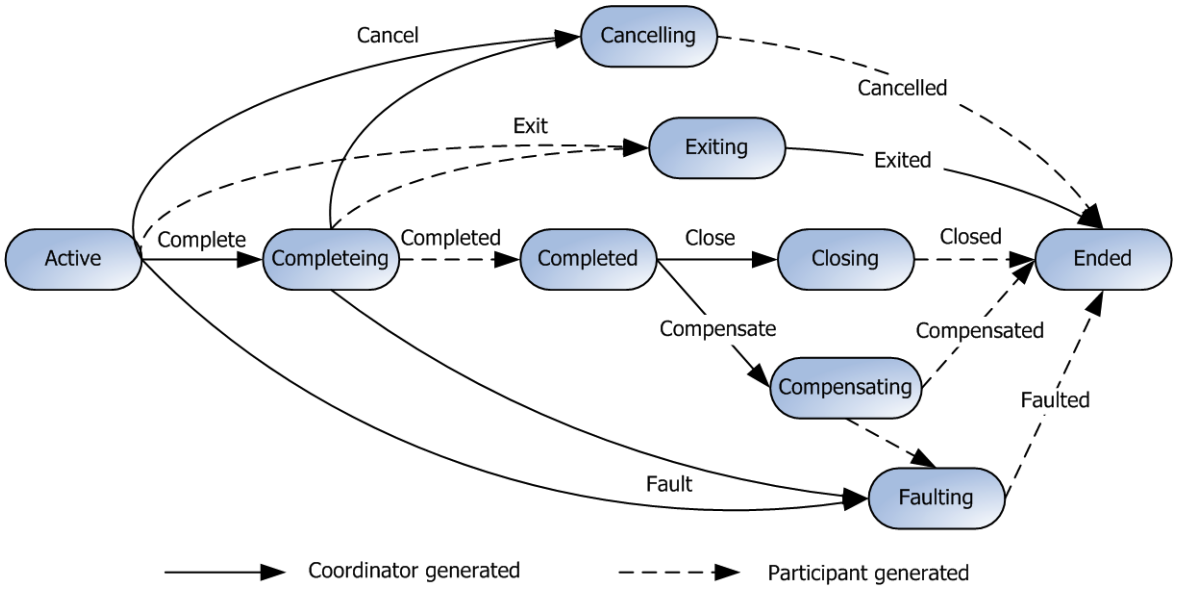
```
@WebService(serviceName="Hotel",
            targetNamespace="http://hotel.tm.jboss.org")
@SOAPBiding(style=SOAPBinding.Style.RPC)
@HandlerChain(file="jaxws-handlers-server.xml")
public  class HotelImpl implements Hotel
{
    // ... class implementation
}
```

BA Framework - Developing Web Services

```
<?xml version="1.0" encoding="UTF-8"?>
<application>

        <!-- omitted elements -->

        <module>
                <java>baframework.aop</java>
        </module>

</application>
```

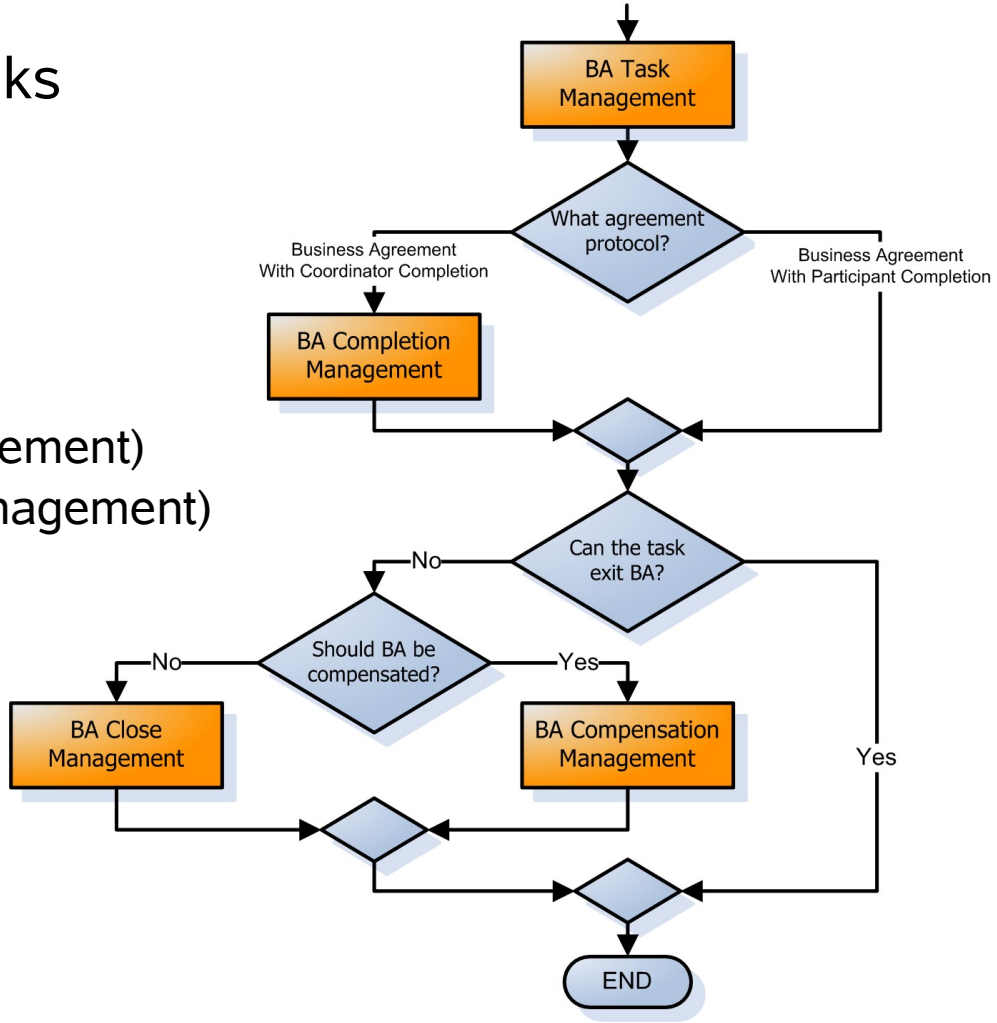
WS-Business Activity (2)

- Agreement protocols:
  - Business Agreement with Participant Completion
  - Business Agreement with Coordinator Completion

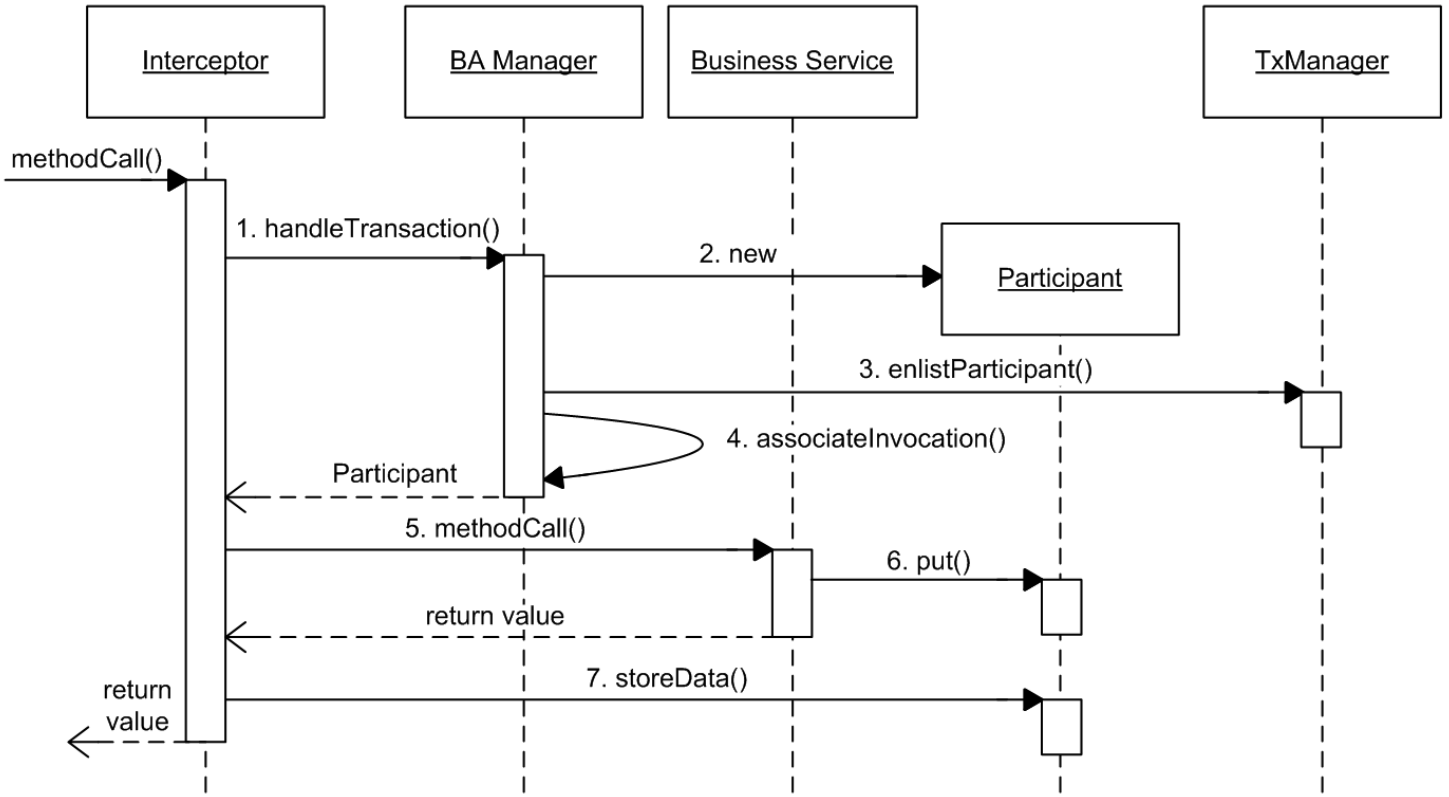


BA Framework - transaction management

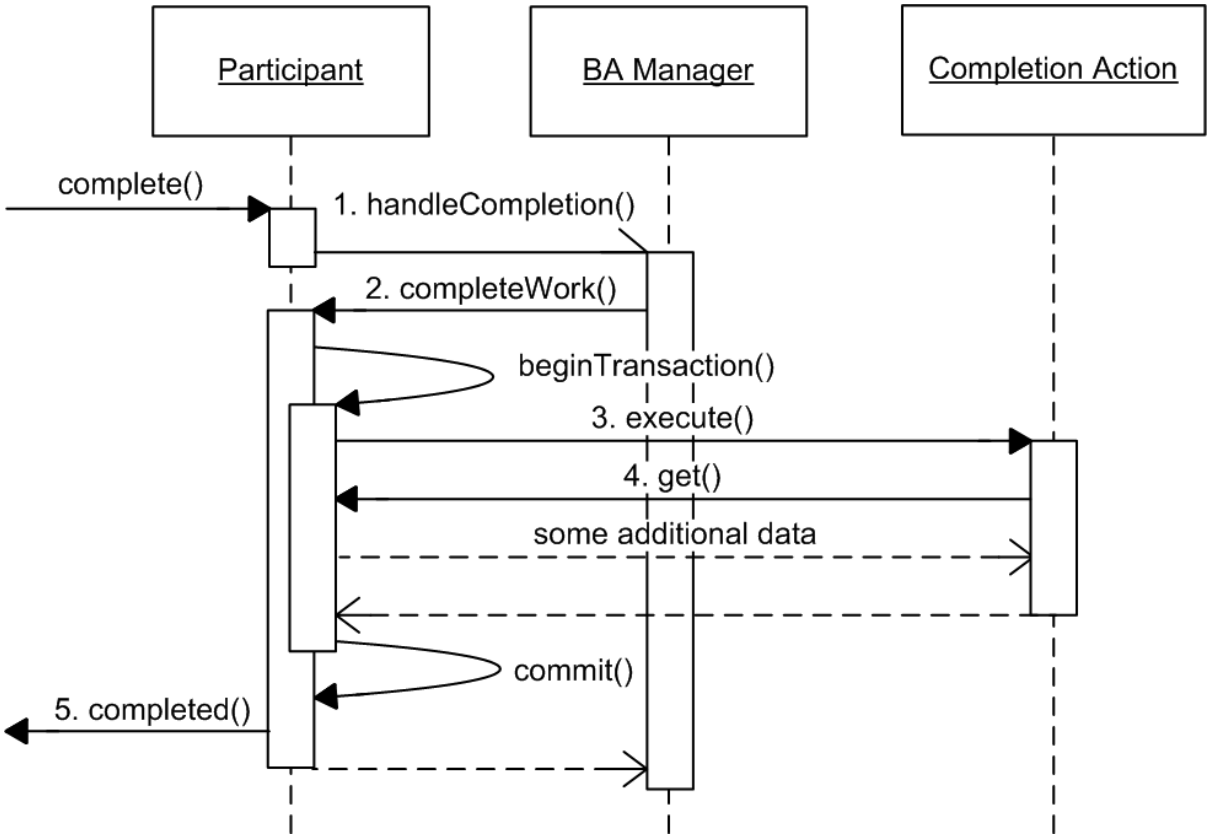
- Framework manages tasks
  - task: original invocation
    (+ completion action)
    (+ compensation action)
  - 4-step processing:
    - BA Task Management
    - (BA Completion Management)
    - (BA Compensation Management)
    - BA Close Management



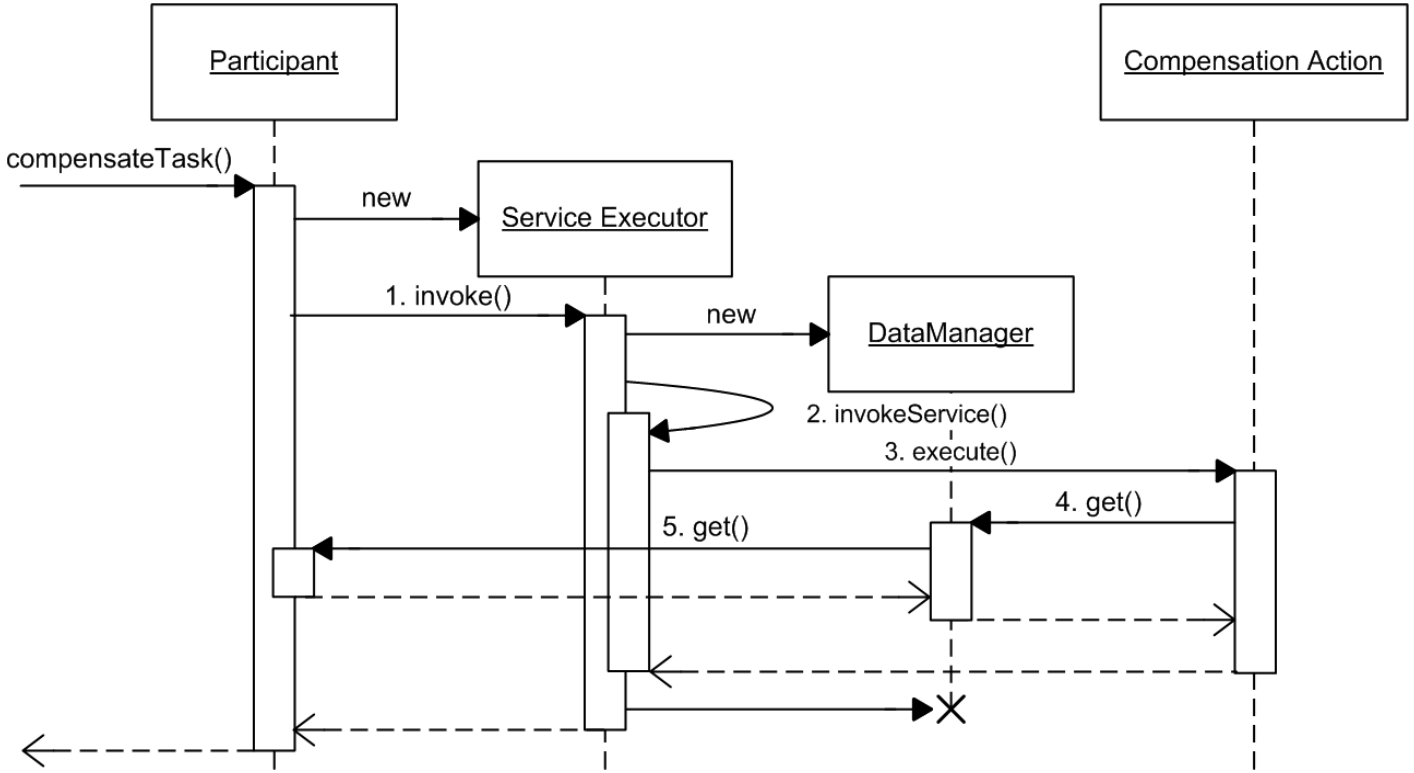
BA Task Management



BA Completion Management



BA Compensation Management



BA Close Management