Grid Business Process: Case Study

Authors
Asif Akram[1], Sanjay Chaudhary[2], Prateek Jain[2], Zakir Laliwala[2] and Rob Allan[1]
Chapter for a book
"Securing Web Services: Practical Usage of Standards and Specifications",
a book edited by Dr. PeriorellisPanos and published by Idea Group Inc.

[1]CCLRC Daresbury Laboratory, Daresbury, UK
[2]Dhirubhai Ambani Institute of Information and Communication Technology

Grid Related Specifications and Standards

- Web services architecture requires support for state management, event and notification management, and resource lifecycle management to share and coordinate the use of diverse resources of real life dynamic in 'virtual organizations' (Foster, 2002).
- Recent convergence between Web services and the Grid computing community (Czajkowski, Ferguson, Foster, & et al., 2004) towards the re-factoring and evolution of Grid standards aimed at aligning OGSI functions with the emerging consensus on Web services architecture (Booth, Haas, McCabe, & et al, 2004).
- This effort has produced two important sets of specifications:
  - WS-RF (WS-ResourceFramework, 2005) and
  - WS-Notification (WS-Notification, 2005)

Grid Related Specifications and Standards

Web Services Addressing

- The WS-Addressing specification defines a standard for incorporating message addressing information into web services messages.
- WS-Addressing provides a uniform addressing method for SOAP (SOAP, 2000) messages traveling over synchronous and/or asynchronous transports.
- SOAP does not provide a standard way to specify where a message is going, how to return a response, or where to report an error.
- WS-Addressing has two basic functions:
  - to provide a standard way of referencing endpoints, and
  - to provide a standard set of headers providing information about a message, such as where it's going and where replies and faults should go.
- To enable this, WS-Addressing incorporates
  - delivery, reply-to, and fault handler addressing information into a SOAP envelope.

Grid Related Specifications and Standards

Web Services Addressing - Example

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
                xmlns:wsa="http://www.w3.org/2004/12/addressing">
    <S:Header>
        <wsa:MessageID>
            http://......
        </wsa:MessageID>
        <wsa:ReplyTo>
            <wsa:Address>http://client.application</wsa:Address>
        </wsa:ReplyTo>
        <wsa:FaultTo>
            <wsa:Address>http://logging.application</wsa:Address>
        </wsa:FaultTo>
        <wsa:To>http://service.to.invoke</wsa:To>
        <wsa:Action>http:// method.to.invoke</wsa:Action>
    </S:Header>
    <S:Body>
         <!-- The message body of the SOAP request appears here -->
    </S:Body>
</S:Envelope>
```

Grid Related Specifications and Standards

Web Services Addressing

- WS-Addressing also defines a standard for including service-specific attributes within an address for use in routing the message to a service or for use by the destination service itself.
- These attributes are particularly useful for the creation of stateful web services, which are services that can receive a series of requests from a particular client and remember some state information between requests.
- WS-Addressing introduces two new constructs for Web services vocabulary:
  - endpoint references and
  - message addressing properties.
- "Endpoint" is an established term for a destination at which a Web service can be accessed. Endpoint references are a new model for describing these destinations.

Grid Related Specifications and Standards

Endpoint Reference

- An endpoint reference is a data structure that is defined to encapsulate all the information required to reach a service endpoint at runtime.
- Endpoint reference can have five different types of information only the address of the service is mandatory:
  - address: The address is a URI (such as a URL), a potential source or destination of Web service.
  - reference properties: The reference properties help to identify the resource to be used during service invocation.
  - reference parameters: The reference parameters wrap the information required for successful invocation of the service which is not required to identify the resource.
  - port type: The portType of the EPR filters the operation to be invoked, as defined in the WSDL file. Single WSDL can have multiple portType and each portType can have multiple operations.

Grid Related Specifications and Standards

Endpoint Reference

- service-port: Each portType can have multiple service ports supporting various transportation mechanism. The service-port identifies the specific implementation of the portType from the possible set of transportation bindings.
- policy: The policy details the information required to follow for successful interaction with the service according to the WS-Policy specification.

- A significant aspect of an endpoint reference is the ability to attach data from your own XML namespace via reference properties or reference parameters.
- Both of these elements are collections of properties and values that you can use to incorporate elements from your own XML namespace (or any XML namespace) into the endpoint reference.
- The key distinction between a reference property and a reference parameter is not the format but the intended usage.

Grid Related Specifications and Standards

Web Service Resource Framework (WS-RF)

- Web Service Resource Framework is a specification approved by Organization for the Advancement of Structured Information Standards (OASIS) for various aspects related to stateful Web services.
- It provides definition of message exchange pattern as to how stateful Web services should be created, addressed, and destroyed. (Foster, Czajkowski, & et al, 2005).
- WS-RF defines conventions within the context of established Web Services standards for managing 'state' so that applications can reliably share changing information, and discover, inspect and interact with stateful resources in a standard and interoperable way.

Grid Related Specifications and Standards

Web Service Resource Framework (WS-RF)

- WS-RF is a collection of four specifications, namely
  - WS-ResourceProperties,
  - WS-ResourceLifetime,
  - WS-ServiceGroup, and
  - WS-BaseFaults.
- It also refers to two related specifications, namely
  - WS-Notification and
  - WS-Addressing.

Grid Related Specifications and Standards

- WS-Notification
  - WS-BaseNotification
    - NotificationProducer
    - NotificationConsumer
    - NotificationMessage
- WS-Topics
  - Topics and Topic Namespaces
  - Topic Expression Dialects
  - Topic Set
- WS-BrokeredNotification
  - Interface

Grid Related Specifications and Standards

- Web Services Distributed Management
  - Management Using Web Services (MUWS)
    - Manageable Resource
    - Metadata
    - Relationship
- Management of Web Services (MOWS)

WS-Resource

- The WS-Resource represents state in a Web services context.
- This state has components, called Resource Property elements, which represent atomic elements of state that can be read and written.
- A set of resource property elements are gathered together into a resource property document: an XML document that can be queried by client applications using XPath or other query languages.
- In many cases, a designer chooses to expose only a portion of the resource's state as resource properties.
- WS-RF supports dynamic insertion and deletion of the resource property elements of a WS-Resource at run time.

WS-Resource

- A resource itself is a distributed object, expressed as an association of an XML document with a defined type attached with the Web service portType in the WSDL.
- Each Resource can be serialized in XML format to embed it in the message before sending across the network.
- Each resource has a unique identity and distinguishable handler. Although resource itself is not attached to any Uniform Resource Locator (URL), it does provide the URL of the Web service that manages it.
- The unique identity of the resource and the URL of the managing Web service is called an Endpoint Reference (EPR), which adheres to Web Services Addressing (WSA) (WS-Addressing, 2004).
- Instances of a Resource have a certain lifetime, which can be renewed before they expire as specified by WS-ResourceLifetime specification.
- They can also be destroyed pre-maturely as required by the application.
- The lifetime of an instance of a Resource is managed by the client itself or any other process interacting as a client, independent of the Web service and its container.

WS-Resource

- WS-Resources are not bound to a single Web service; in fact multiple Web services can manage and monitor the same WS-Resource instance with different business logic and from a different perspective.
- Similarly, WS-Resources are not confined to a single organization and multiple organizations may work together on the same WS-Resource, which leads to the concept of collaboration (Akram).
- The Resource EPR's are generated dynamically and can be discovered, inspected and monitored dynamically with the help of dedicated Web services.
- Resource sharing is extensively used for load balancing. Semantically similar or cloned Web services are deployed independently for multiple client access.
- At run time, appropriate EPR's of the WS-Resource are generated with the same unique Resource identity but with different URLs of managing Web services.

WS-Resource

- Resources are composed of Resource Properties, which reflect their state.
- These can vary from simple to complex data types and even reference other Resources.
- Referencing other Resources through Resource Properties is a powerful concept, which defines and elaborates inter-dependency of the Resources at a lower level.
- This eliminates complicated business logic in a similar way similar to the mapping of Entity Relationships in a Relational Database through primary and foreign keys.
- Resources are not only dependent on the state of other Resources but can even query and modify them.

WS-Resource

- Resources that are shared by various Web services, in particular Web services deployed on different domains and within different organizational boundaries, must be globally accessible.
- Resources, which reside solely in memory can't survive server restart and cannot be shared.
- Long running applications managing a large number of Resources can drastically affect the performance of an application; Application, which manage large number of resources suffer from performance aspects.
- For such applications, the best approach is to store these resources in a persistent storage and load them into memory as required. Various implementations of WS-RF have the notion of Resource Persistence, which is mainly in the form of flat files.
- Persistence interfaces provided by the various WS-RF frameworks can be extended for either local or remote persistence and for physical storage in relational, object or XML databases.

Implied Resource pattern

- The WSA is used to define the relationship between Web services and stateful resources. This is the core of the WS-Resource Framework.
- When dealing with multiple resources, the WS-RF specifications recommend the use of the Factory/Instance pattern i.e. Implied Resource pattern.
- The implied resource pattern is a convention on XML, WSDL, and WS-Addressing that defines how a particular WS-Resource is referred within a Web service message context for processing.
- The term implied is used because the identity of the resource isn't part of the request message, but rather is specified using the reference properties feature of WSA.
- The identity of the resource is implied by the context of the message and its association with an endpoint reference, not directly in the signature of the message.
- The use of the WSA is to identify the stateful resource to be used while processing the Web service message. The endpoint reference provides means to point to both the Web service and the stateful resource in one convenient XML element.

Implied Resource pattern

- This factory pattern is well understood in computer science: the notion of an entity that is capable of creating new instances of some component.
- A Factory service is responsible for the resource creation assigning it an identity, and creating a WS-Resource qualified endpoint reference to point to it.
- An Instance service is required to access and manipulate the information contained in the resources according to the business logic.
- Implied Resource Pattern can be extended in various ways to effectively capture the requirements of the application. Different possible extensions are discussed below.

Factory/Instance Pair Pattern

- The Factory/Instance Pair Model is the simplest model, in which for each resource there is a Factory Service to instantiate the resource and corresponding Instance Service to manage the resource.
- In a typical application different Factory Services are independent of each other and can work in isolation.
- This is the simplest approach: repeating the similar resource instantiating logic in multiple Factory Services or even the same Factory Service can be deployed multiple times.
- In this model the user manually interacts with different Factory Services; which instantiate the appropriate resources and return the corresponding EPRs to the client.
- The client application contains the logic of deciding when and which resources should be instantiated.

Factory/Instance Collection Pattern

- The Factory/Instance Collection Model is an extension of the Factory/Instance Pair Model.
- The difference being that a single Factory Service instantiates multiple WS-Resources managed by different Instance Services.
- In any Business Process various entities can be tightly coupled and due to this inter-dependency all of these WS-Resources must co-exist before a client may interact with them successfully

Master-Slave Pattern

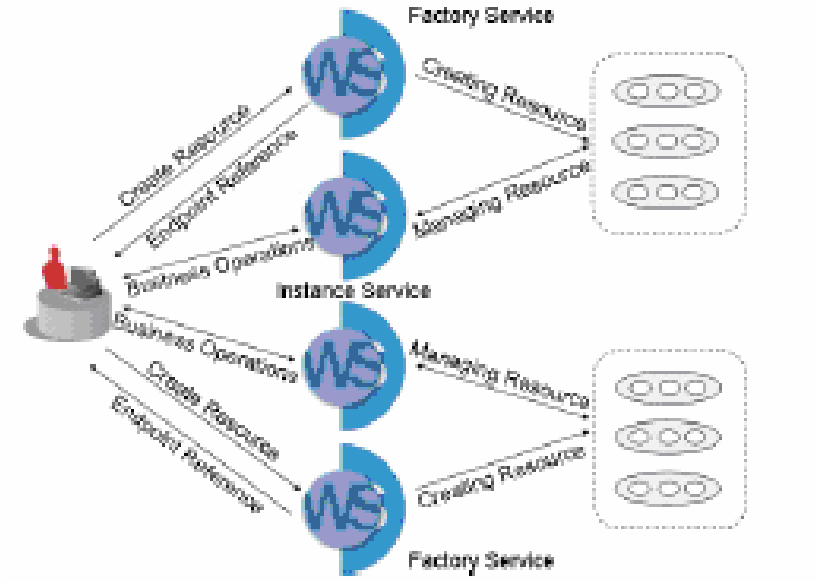
- In a security dominated era with its unpredictable request traffic, different security and load balancing measures are required for an application to run smoothly.
- These measures should be planned at design time, irrespective of the technologies to be used for implementation.
- Business Processes are frequently protected by a firewall. It has to be anticipated that firewall policies will limit direct access from external clients to Resources (i.e. it is most likely that these Resources will be located inside private firewalls, and can only be accessed via known gateway servers).
- Consequently, an extensible Gateway model is required for accessing these resources. This model mandates that all client requests are sent to an externally visible Gateway Web Service before being routed through the firewall to the actual requested service.
- In addition, firewall administrators may implement additional security measures such as IP-recognition between gateway server and service endpoint in the form of Web Services handlers.

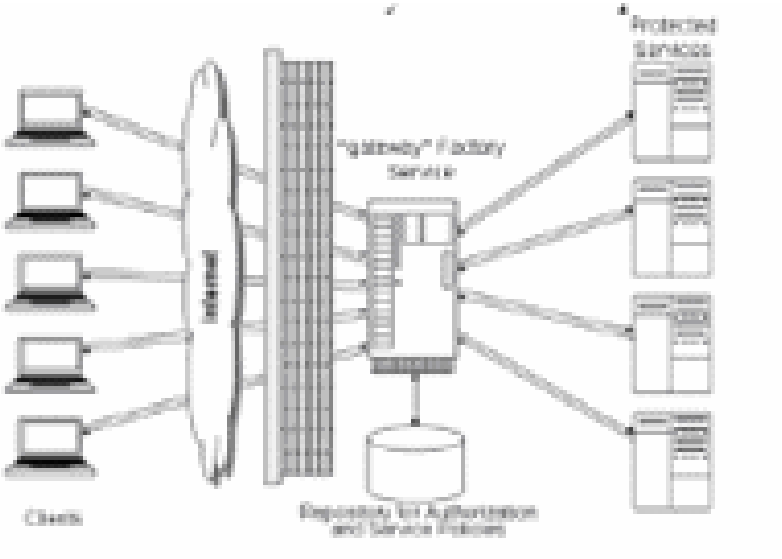
Master-Slave Pattern

- One approach is to use a Gateway Service to manage multiple Factory Services in the Master-Slave format; the client interacts only with the Master Factory Service without knowing the inner details of the application.
- The Master Factory Service performs authentication and authorization of the client before invoking respective Factory Services (Slaves) which are behind the firewall and restricted by strict access polices.

Hybrid Approach

- The best approach for any complicated Business Process is to combine these variations of the Implied Resource Pattern as follows.
- The client still interacts with a single Factory Service which instantiates all mandatory WS-Resources and returns a single EPR.
- Subsequent client interactions invoke the ‘create’ operation of the Factory Service with different ‘parameters’ with the Factory Service instantiating the corresponding WS-Resources according to those parameters.
- Optional WS-Resources are supported using a Factory/Instance Pair model due to their limited usage.





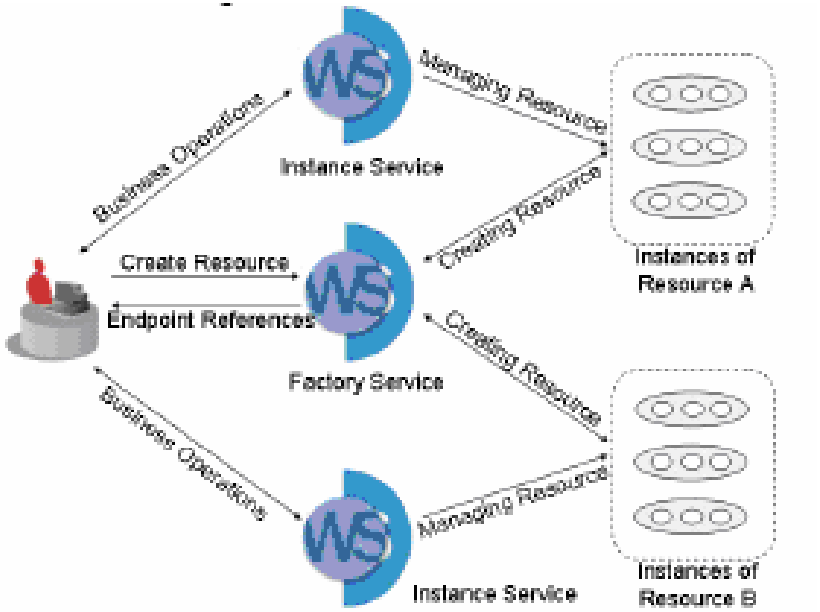


Figure: Factory/Instance pattern

Notification Pattern

- The Event-driven or Notification-based interaction pattern is a commonly used pattern for inter-object communications.
- WS-RF and WSN bring notification-based interaction pattern in Web Services domain.

Notification Pattern

Client as Notification Consumer

- In this approach client application acts as a Notification Consumer, which is notified of any change in the “state” of subscribed WS-Resource instance.
- The client processes the notification messages and updates instances of other related Resources through corresponding Instance Service(s).
- It is the responsibility of a client to inter-relate dependent WS-Resource instances.
- Client application exposes “notify” operation to receive asynchronous notification messages; and implements complicated logic of associating different Resources instances together, whereas Enterprise Application is simple and independent of notification.

Notification Pattern

Service as Notification Consumer

- At application level, services managing different Resource instances are associated together.
- Due to inter-dependency of WS-Resources these managing services have interest in the state of other Resource instances.
- Thus handling the notifications at service level is more appropriate without involving client applications.
- This is the situation where automatic and quick actions are required. The client does not manage actions, and it does not have a role in the decision-making.
- Actions required are related to the main functionality of the application and not with a specific client. In this approach client applications are independent of notification processing.

Notification Pattern

Resource as Notification Consumer

- The two notification approaches discussed above have their own limitations and benefits.
- A third notification model can provide the best of both approaches with an even cleaner design.
- Applications are still easy to manage and maintain and WS-Resource instances can have one-to-one associations.
- In this approach WS-Resource itself is a notification consumer, yet may also act as a producer.
- Each instance of the WS-Resource can subscribe to ‘state’ changes of specific WS-Resource instances whilst broadcasting notification messages related to its own ‘state’. Overall this mechanism gives tighter control on the business logic without interference from the client side.

Notification Pattern

- Implementing a WS-Resource as a notification consumer or a consumer-producer can result in large numbers of messages which can overload the Subscription Manager Service, thus affecting the overall performance of the application.
- The more inter-related instances, the worse the problem becomes. This model should be applied with caution and WS-Resources serving as notification consumers should obey the following guidelines:

1. There is a controlled number of instances of each WS-Resource at any given time;
2. Each WS-Resource has limited dependency on other WS-Resources and is not involved in complicated association linkage;
3. At least one of each WS-Resource instance is available all the time, Brokered Notification is required for persistent WS-Resources;
4. Producer-consumer WS-Resources should be avoided if possible to avoid cyclic notification chains.

Trading scenario

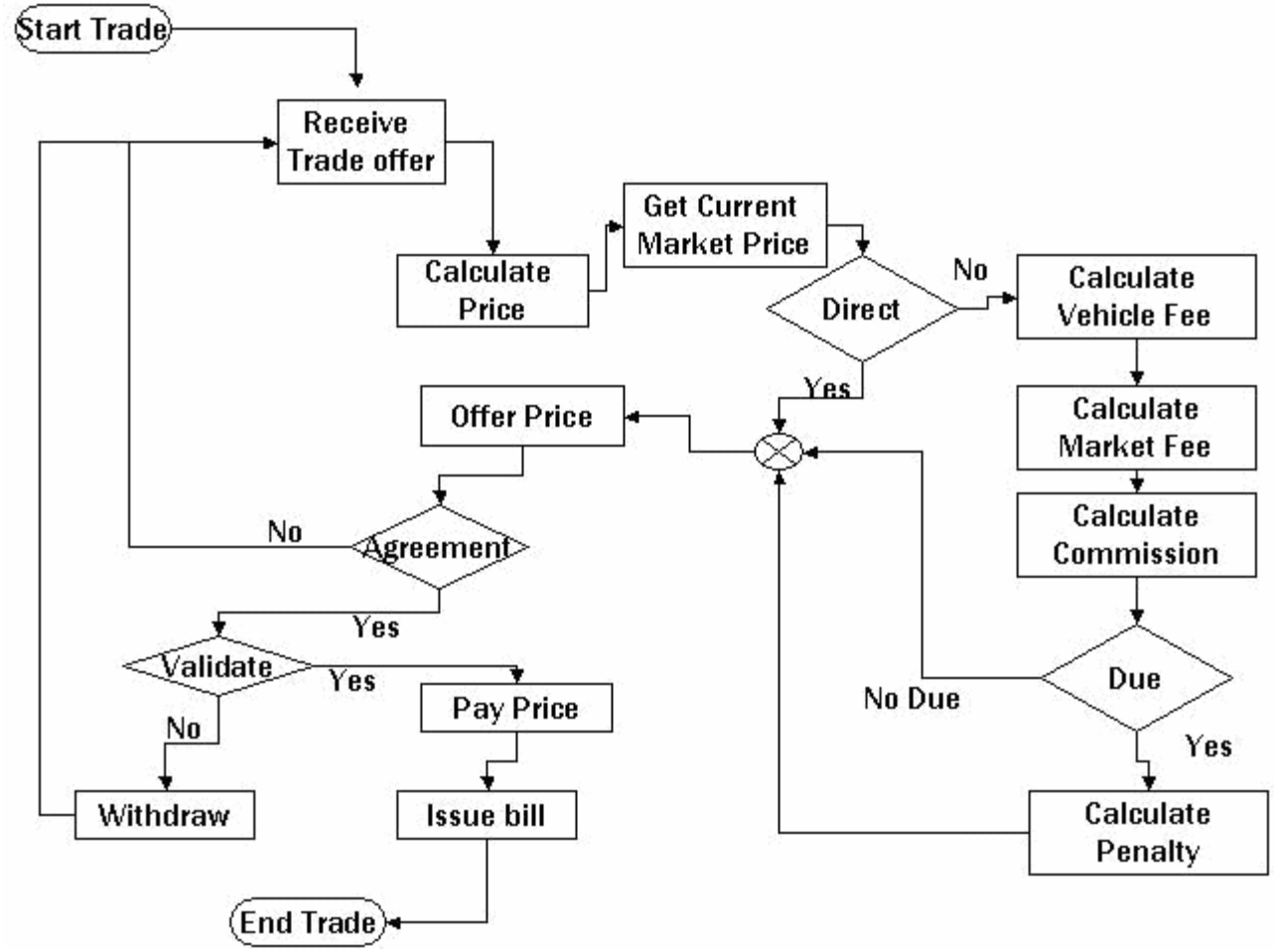
- To demonstrate the application of WS-RF specifications in the Business Process Grid, an application to demonstrate Agricultural Marketing Process in India has been developed (Sorathia, Laliwala, & Chaudhary, 2005).
- In any typical Asian Agro-market, wholesale spot markets and derivative markets are emerging hubs of agricultural marketing business.
- Agro trade in these markets is heavily influenced by local, socio-economic and cultural characteristics.
- This leads to variation in the crop prices of same crop in different markets. The producers have little choice to search for the best available price and are forced to sell their products in a local market.
- Inhibitive transport and storage cost can also play a pivotal role apart from urgency to sell perishable products. Buyers and wholesalers experience difficulties in purchasing desired quality of products at competitive prices.

Trading scenario

- The Model Act (Modal Act, 2003) is expected to bring reforms in the Agricultural Marketing Process. Additional responsibilities are assigned to the existing Agricultural Produce Market Committee (APMC) to realize the reforms and following objectives:
  - To promote new privately-owned market
  - To promote direct sale and contract farming
  - To provide transparency in trading transactions
  - To provide market-led extension
  - To ensure payment on the same day
  - To enable value addition in agricultural produce by promoting agro-processing

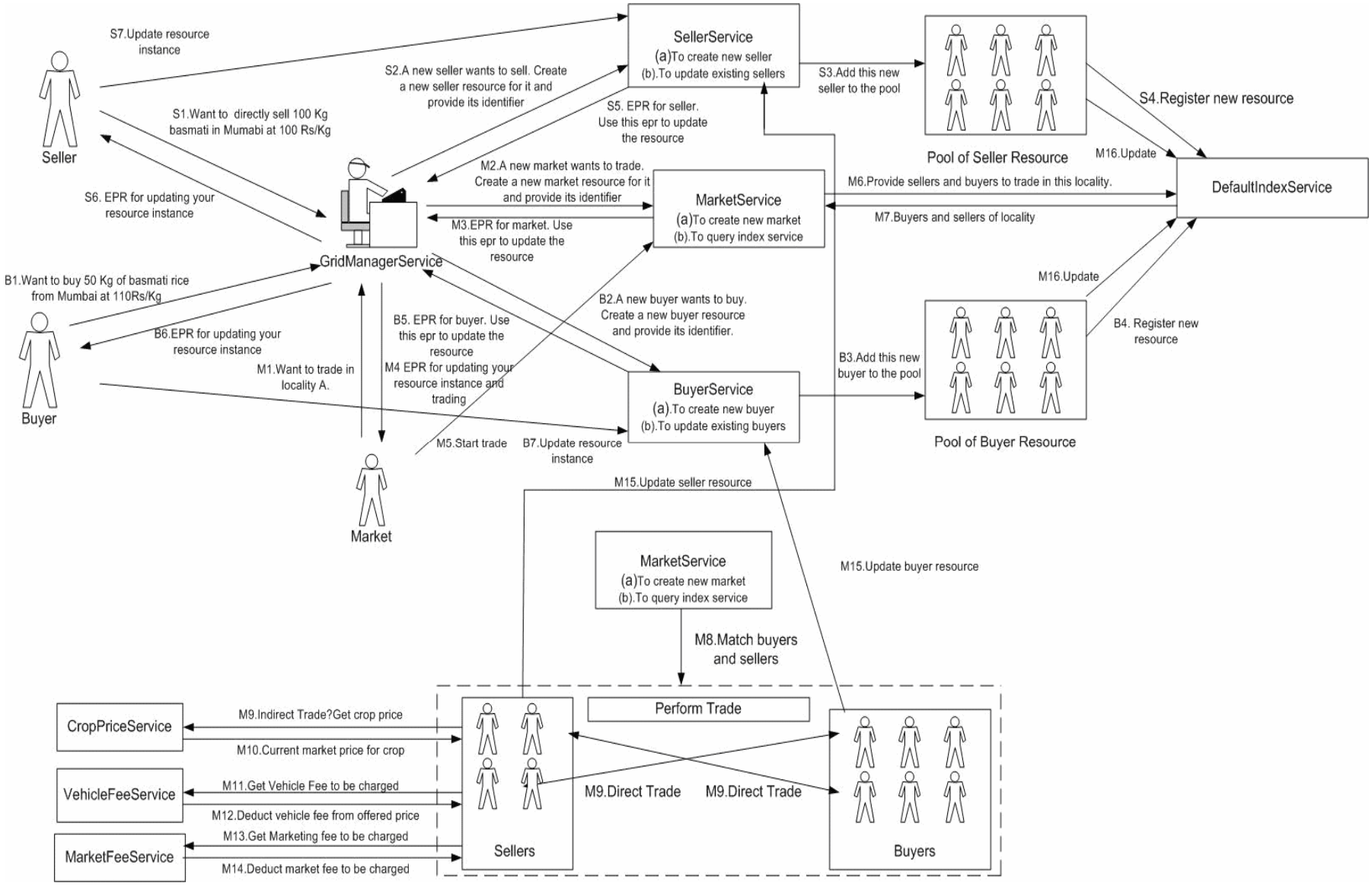
Trading scenario

- According to Model Act, a typical trade can span across the markets located at various places.
- Privately owned markets are allowed and food processing and other related industries are allowed to trade directly with the farmers.
- Contract farming is also an adopted practice in the Model Act.
- Hence the trading in such a competitive market will be more complex than that in the existing scenario.
- This section provides a brief account of a use case to sell agro-produce.



Workflow of Marketing of Agro-produce

Execution of Trade Workflow



Interaction Diagram of Agricultural Marketing System

Implementation Challenges

- State and Execution Monitoring Challenges
- Event and Notification Challenges

System design

- After identification of the implementation challenges, this section discusses system design of the application. It has been observed that business services might be implemented and hosted by a separate organization depending upon the specialization of the organization.
- If development is done using Service Orientation, it is likely that such business services are implemented and exposed as Web services (Laliwala, Jain, & Chaudhary, 2006).
- There is a possibility that other organizations can be enabled to utilize these loosely coupled services to meet their requirements.
- From the service provider's point of view, as these services can be joined together to constitute a complete business process, it has become essential to make provisions for efficient integration with heterogeneous client environments.

Components of Grid Application

- The case study is designed following the service-oriented grid architecture to provide state, notification, execution and monitoring, and scalability.
- The role of the targeted system is to automate the business process with interoperable integration of scattered services. The proposal is mainly dependent upon various WS-* specifications, specifically set of WSRF and WS-Notification (WSN) specifications and WSDM specifications for monitoring, to achieve required compliance.
- The business logic adopted for this case study follows the proposed framework for marketing of agricultural produce.

Components of Grid Application

Grid Manager

- Grid Manager Service is implementation of the Factory/Instance Collection Pattern which itself is extension of the Implied Resource Pattern commonly used in WS-RF.
- It initiates different type of resources i.e. Seller, Buyer, Crop and Market according to the client request; which can be hosted by different Grid Nodes.
- The Grid Manager orchestrates different components of the system in predefined manner for smooth working of the whole application.
- The Grid Manager is the first point of the contact with the system.

Components of Grid Application

Grid Nodes

- Grid Nodes are hosting environment which can host different vanilla or stateful Web Services.
- The services hosted on different nodes are used in various combinations to capture the requirement of the application.
- Each initialized WS-Resource has Endpoint Reference (EPR) attached to it, which identify the resource itself and the location of URL of the managing Instance Service.
- Clients use the corresponding EPR to interact with the specific resource.
- Ideally these Grid Nodes should be geographically dispersed but in this case study different nodes can be on the same machine.

Components of Grid Application

Grid Registry

- Grid Registry is a special Service, which keeps track of various resources initialized by the Grid Manager during the course of application.
- The Grid Client (seller or buyer) can query the Grid Registry to search appropriate resources and can update the resource related to it.
- Grid Registry is based on the WS-ServiceGroup specification, which aggregates information related to different resources for search and query.

Components of Grid Application

Grid Client

- Grid Client is an external application which either request for initialization of the resource/s or interact with the existing resource/s.
- The client has compatibility with WSA specification to work with WS-Resources and corresponding services through their EPR.
- Manipulation of state-full resources requires WS-RF API available on the client side to create new resource, query, destroy or modify existing resources etc.
- Seller and Buyer fall in the category of the Grid Client.
- In this case study the Grid Client has three flavors, for buyers, sellers and administrator which instantiate the market instance for the trade.

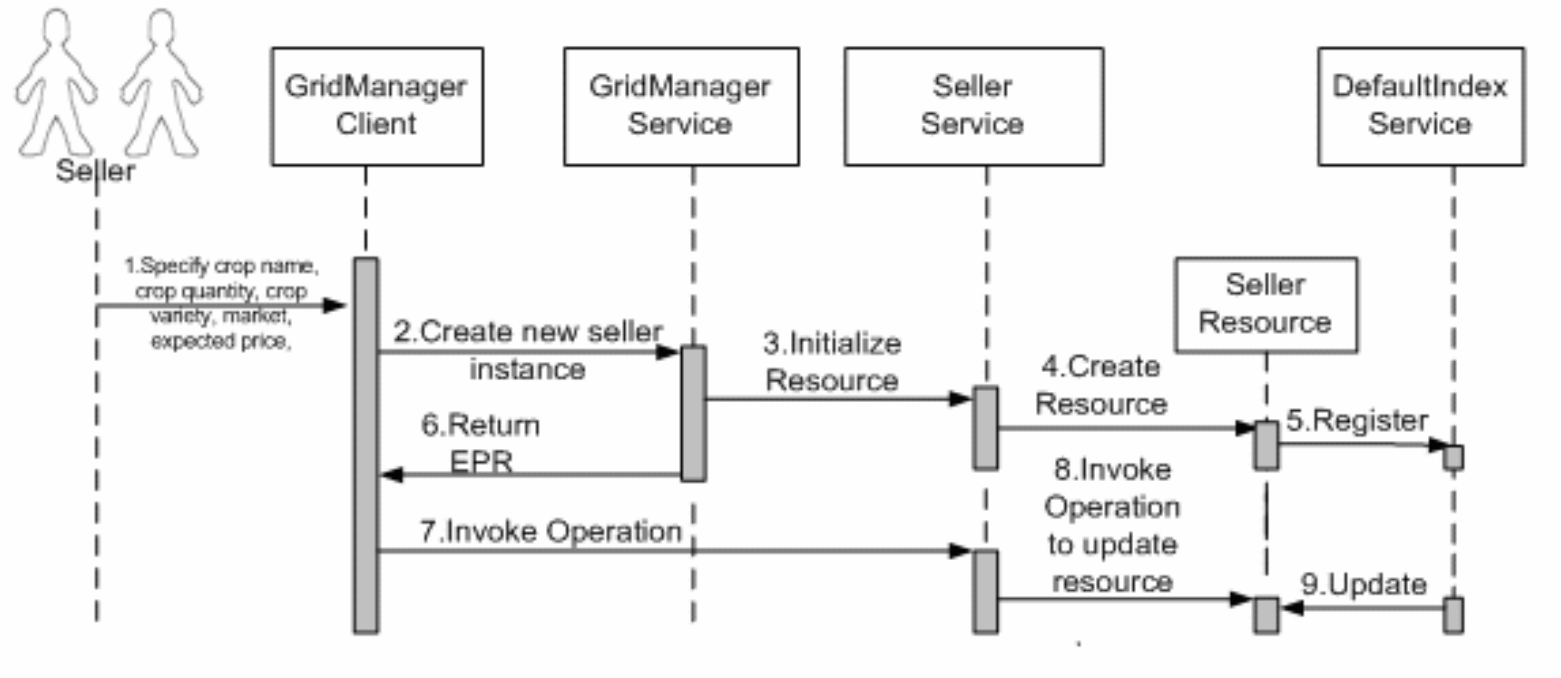
Interactions among Components

- In the Grid architecture for our agrocultural case study, various WS-Resources are involved which are initialized during different stages of the business process. The WS-Resources involved in a trading scenario of our agro market case study are:
  - Seller
  - Buyer
  - Market
- At any time there can be various instances of similar WS-Resource instantiated during different phases of the business process lifecycle.
- For example at any time there can be various buyers and sellers in a single market and similarly the same buyer or seller may be participating in different transactions initiated in different markets.
- In the following sub-sections, steps involved the instantiation of different WS-Resources and interactions among different components involved in the instantiation are illustrated through sequence diagram.

Interactions among Components

Instantiation and Interaction with the Seller entity

- The first interaction of the seller with the system results in the creation of the new seller WS-Resource.
- This resource captures all details of the specific seller and is used for the future interaction with that seller.
- The Figure shows the sequence of events, which allow registration of the buyer with intention to purchase the required produce in the market.



Seller Service Sequence Diagram

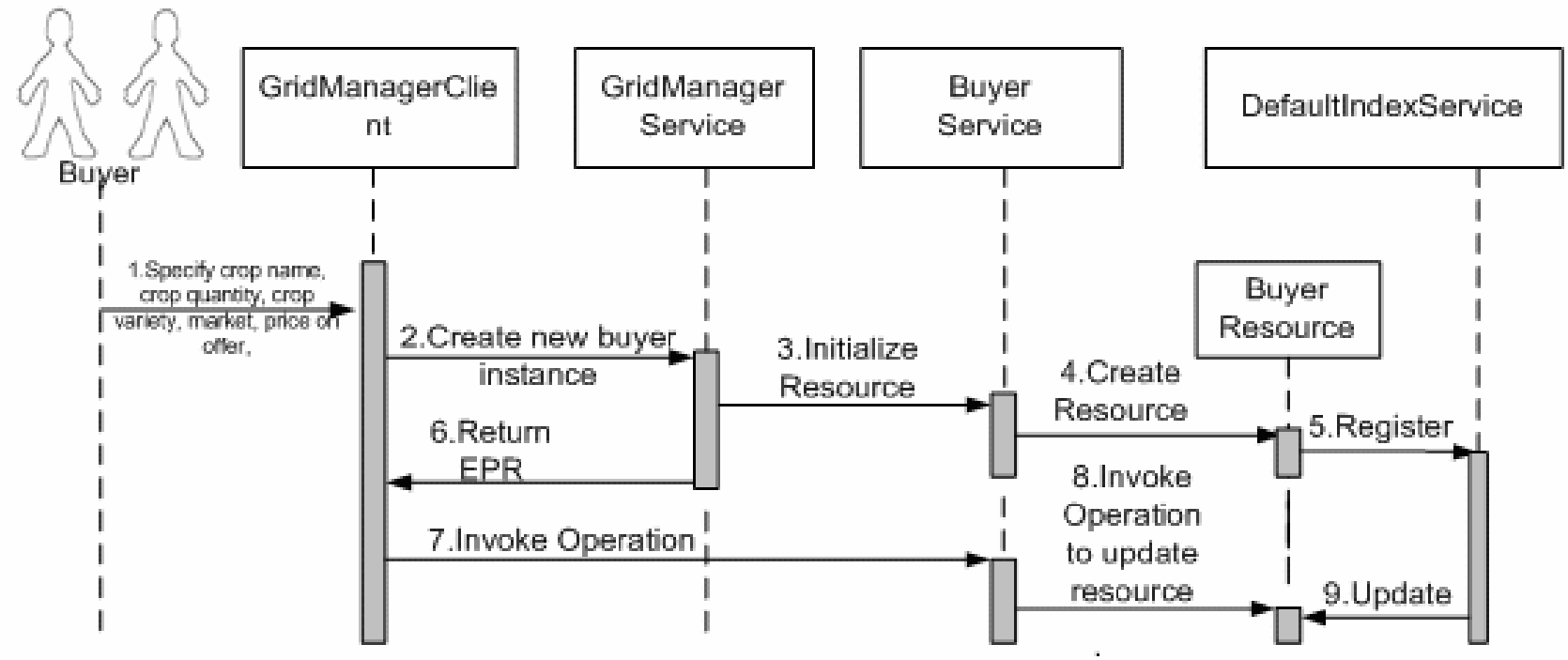
Interactions among Components

1. The trading for a seller begins with the expression of intent to sell the available crops by supplying crop name, crop quantity, crop variety, expected price/kg of crop and market city for trade by using the GridManagerClient.
2. The GridManagerClient sends a request to GridManagerService, which looks up the instance service of a Seller to create an instance of SellerResource.
3. Newly created Seller resource, registers with the DefaultIndexService (i.e. Grid Registry) exposed by Globus container to register the details of the newly created seller resource.
4. The GridManager returns back the EPR of a newly created SellerResource to the GridManagerClient.
5. The GridManagerClient, utilizes the EPR to invoke different operations to update the newly created resource.

Interactions among Components

Instantiation and Interaction with the Buyer entity

- Similarly the first interaction of the buyer with the system results in the creation of the new buyer WS-Resource.
- This resource captures all details of the specific buyer and is used for the future interaction with that seller.
- The Figure shows the sequence of events, which allow registration of the buyer with intention to participate in the trade.



Buyer Service Sequence Diagram

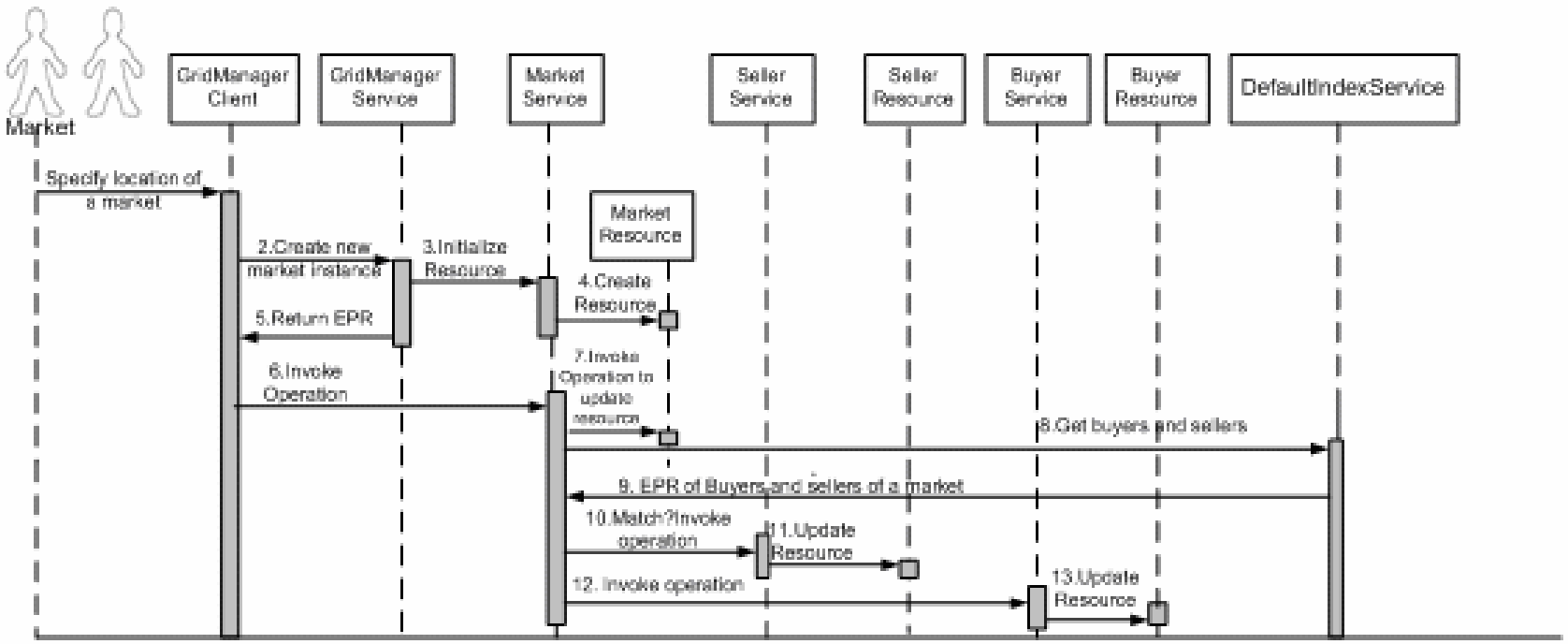
Interactions among Components

1. The trading for a buyer begins with the expression of intent to buy the available crops by supplying crop name, crop quantity, crop variety, offered price/kg of crop and market city for trade by using the GridManagerClient.
2. The GridManagerClient sends a request to GridManagerService, which looks up the instance service of a Buyer to create an instance of BuyerResource.
3. Newly created Buyer resource, registers with the DefaultIndexService (i.e. Grid Registry) exposed by Globus container to register the details of the newly created buyer resource
4. The GridManager returns back the EPR of a newly created BuyerResource to the GridManagerClient.
5. The EPR is utilized by the GridManagerClient to invoke different operations to update the newly created resource.

Interactions among Components

Market Service for Direct Trading

- In Direct Trading the system matches resources for different sellers and buyers which are trading in the same location.
- Once any such match is found then it means there is atleast one buyer interested in the produce of a seller.
- To simulate the successful trade the resource properties for seller and buyers are updated. Below is the sequence diagram of events involved in direct trading.



Sequence Diagram of Market Service for Direct Trading

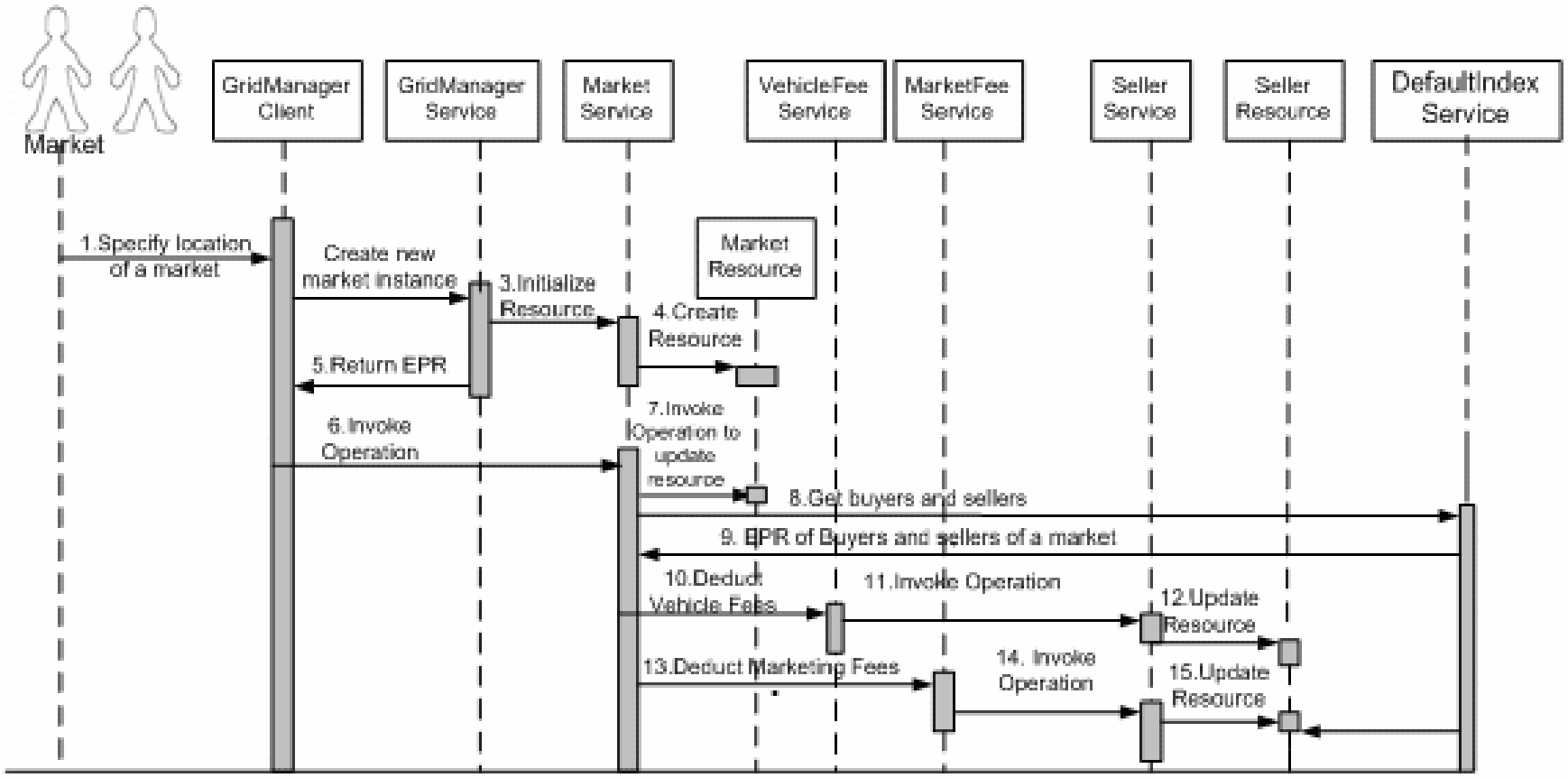
Interactions among Components

1. The first step for successful trade is the instantiation of the MarketResource by passing the required location. The market only starts its operation once MarketResource is instantiated through GridManagerClient.
2. The GridManagerClient invokes the GridManagerService to create a new instance of Market Resource. The GridManagerService, triggers the creation of a new MarketResource using the Market instance service and returns the EPR to the GridManagerClient.
3. The newly created EPR is utilized to invoke operations to start trade.
4. The market service queries the DefaultIndexService (i.e. Grid Registry) by supplying the location to obtain the information about sellers and buyers registered with preferences to trade at particular location for which this market service instance is running.
5. The DefaultIndexService, returns the EPR of Buyer and Seller resources interested in trading at a given location.
6. The MarketService compares for the resource properties of the buyer and seller resources by utilizing retrieved EPR's. If the trading mode for a seller is direct and if its resource properties matches with that of required by a buyer, trading is performed by invoking operations on the seller and buyer resources to adjust the crop quantities and earning for a seller, crop quantity and amount spent for a buyer.

Interactions among Components

Market Service for Indirect Trading

- The sequence of events for indirect trade is quite similar to the direct trade.
- The initial matching of seller and buyer resources in the market is followed by invoking operations for the services specific to indirect trading and calculation of the final price after deducting appropriate charges.
- Below is the sequence diagram for the Indirect Trading:



Sequence Diagram of Market Service for Indirect Trading

Interactions among Components

1. The market starts its operation by expressing its intent to trade. Market uses the GridManagerClient and informs about the location, where it wants to trade.
2. The GridManagerClient invokes the GridManagerService to create a new instance of Market Resource. The GridManagerService, triggers the creation of a new MarketResource using the Market instance service and returns the EPR to the GridManagerClient.
3. The newly created EPR is utilized to invoke operations to start trade.
4. The MarketService queries the DefaultIndexService (i.e. Grid Registry) by supplying the location to obtain the information about sellers and buyers registered with preferences to trade at particular location for which this market service instance is running.
5. The DefaultIndexService, returns the EPR of Buyer and Seller resources interested in trading at a given location.
6. The MarketService, compare for the resource properties of the seller and buyers trading within the same location. On any successful match, the trade preference of the seller is quried for indirect trading. The indirect trading is simulated by deducting transportation and marketing fees from the asked price.
7. As a result of successful transaction the seller resource is updated to reflect the new values i.e. crop still available to sell and total earnings.
8. Similarly the buyer resource is updated to reflect new quantity of crop required and amount spent in single year.

Implementation