# SENG422/522-Final Exam Review

**Exercise 1:**

While many systems can be implemented using a range of architectural styles, there are usually features of a problem that encourage the choice of a particular style in the absence of any other constraints. For the following types of system, identify what you consider to be their major features and identify the architectural styles that are likely to be the most appropriate:

(a) A bank auto-teller machine;
(b) A spell-checker used to analyze static files of text;
(c) A program that reformats ‘raw text’ into the page description language employed by a particular printer.

**Exercise 2:**

1. What does IIOP stand for, and what is its significance?

2. What is a client stub?

3. What is an object reference?

4. What is marshaling?

5. What does the following IDL code define:

```
typedef sequence<sequence<float>> temperatureSequence;
```

6. Because an IDL method can return a value, what is the purpose of out and inout parameter types?

7. The IDL *union* type can be used for a parameter that will need to pass one of a small number of types. Use it to define the type of a parameter that is sometimes empty and sometimes has the type *Value*.

**Exercise 2 (ctd):**

8. you think of a mechanism, using oneway calls, to return a result to the caller?

9. What are the eight dimensions of transparency in distributed architectures?

10. What can transparency dimensions be used for?

11. What is the difference between location and access transparency?

12. What are the differences between performance and scalability transparency?

13. Which ISO/OSI layers does an object-oriented middleware like CORBA implement?

14. Explain the difference between naming and trading in the CORBA framework.

15. When would you use naming and would use trading for locating objects?

**Exercise 2(ctd.):**

16. What are two most important operations in naming?

17. What are the principal trading operations?

18. What is the difference between service type and an object type?

19. How are service type hierarchies and object type hierarchies related?

20. What are policies used for in trading?

21. What is the impact of preferences passed to a trader query on the sequence of offers returned?

22. Draw a sequence diagram to illustrate how a push consumer connects to an event channel.

**Exercise 3:**

Consider an aircraft navigation system that computes and displays its global position. The current position is updated on a regular basis.

The system consists of three units: a Rate Generator, a Global Positioning Sensor (GPS), and a Display device.

The rate Generator generates periodic pulses refreshing the GPS. The GPS then computes the current coordinates, and notifies the Display device accordingly. The display device –receives refresh signal from the GPS, then reads the current coordinates and updates the display.

1. Propose a CORBA CCM component-based design for the system. Show the different components involved and the relationships between them by providing a component diagram.

2. Provide a CCM IDL V 3.x specification for the system; convert the IDL code for one of the components to its equivalent CORBA V 2.x IDL.

## Exercise 4:

A telecommunication company decides to extend its services by allowing its users to place calls through the Internet. A user may place a call from a client workstation using a particular VoIP protocol, or using an IP phone, or from home using his regular phone.

Phone Servers will register their capabilities with a central server. The central server plays the role of a load balancer by directing requests to the proper server. Load balancing will be achieved in this case using CORBA trading service. Every server or service provider will send to the load balancer the following information:

- Its location (area codes it can terminate calls to)
- VoIP protocols it can support (MGCP, H323, or SIP)
- Price
- Link quality

When a client places a call it will specify to the load balancer the area code and phone number it wants to call, the protocol it uses, and the type of service plan the user has subscribed for. The load balancer will be able to match the request to the proper server.

1. Specify the factors that affect the load balancing process.
2. Provides a sequence diagram showing the interactions between the client, the load balancer, and the server.

**Exercise 5:**

Consider a shopping cart application used in an e-commerce website. The application consists of a *Customer* component, a *Shopping Cart* component, and a *Checkout* component.

The *Customer* component maintains the customer account information. When a customer comes to the site and logs in, the application will find his record and account number for use during the session, and create a new shopping cart so he can start shopping.

A shopping session starts when a customer puts the first item into his shopping cart, and ends when he checks out. The checkout component generates a bill and a shipping order from the contents of the shopping cart.

1. Design graphically the CCM component assembly corresponding to the shopping cart application.

2. Provide a CCM IDL V 3.x specification for the system.

3. Convert the IDL code for one of the components to its equivalent CORBA V 2.x IDL.

**Exercise 6:**

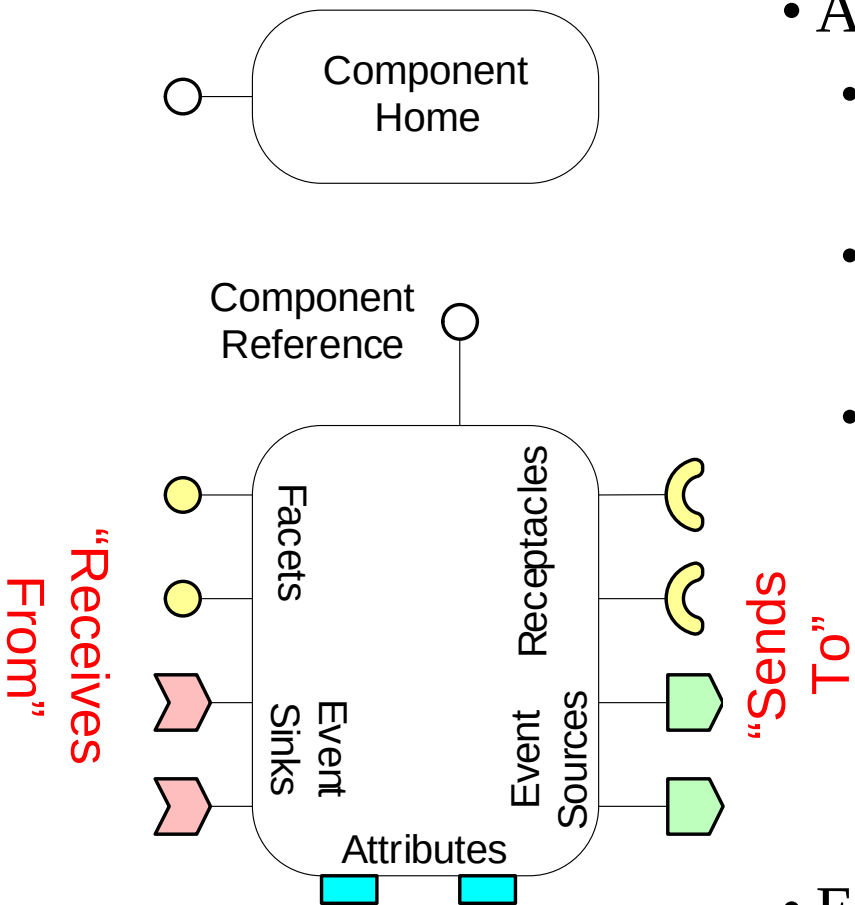
Naming contexts have the interface CosNaming::NamingContext from the OMG's name service specification. The IDL definition for this interface is given below:

```
interface NamingContext {
   void bind(in Name n, in Object obj);
   Object resolve (in Name n);
   void unbind(in Name n);
   NamingContext new_context();
   NamingContext bind_new_context(in Name n);
   void destroy();
   void list (in unsigned long how_many,
      out BindingList bl, out BindingIterator bi);
};
```

1. Draw a use case diagram that illustrates how a client and administrator use the NamingContext interface, the respective operations they perform, and the relationships between these use cases.

2. Draw two sequence diagrams to illustrate the binding and resolving scenarios described in the interface.

# CORBA Component Model




- A CORBA component can contain *ports*:
  - *Facets* (`provides`)
    - Offers operation interfaces
  - *Receptacles* (`uses`)
    - Required operation interfaces
  - *Event sources* (`publishes` & `emits`)
    - Produced events

  *Event sinks* (`consumes`)
    - Consumed events

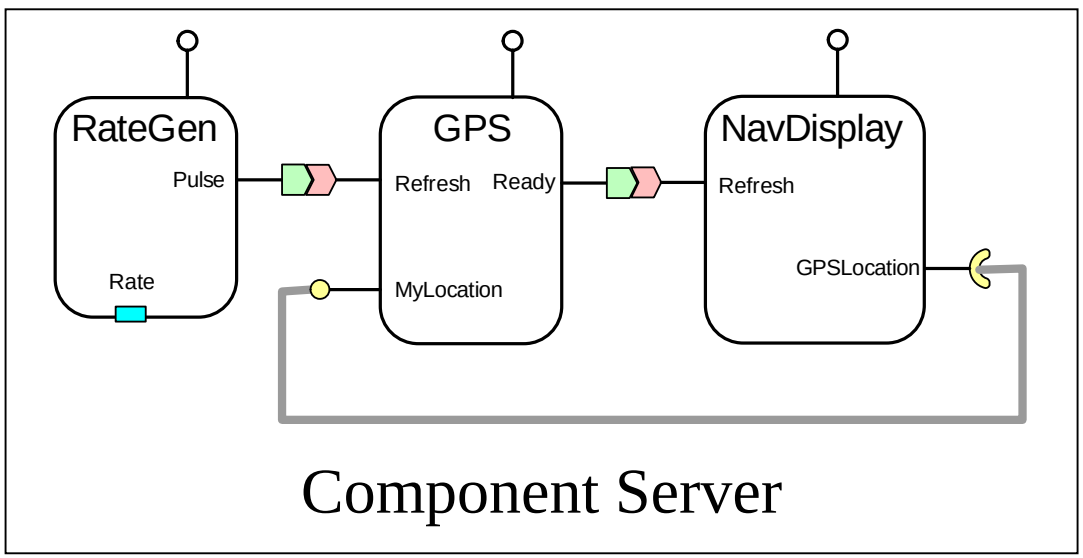
  *Attributes* (`attribute`)
    - configurable properties
- Each component instance is created & managed by a unique component **`home`**

# Exercise 3: Airplane Application

Rate Generator

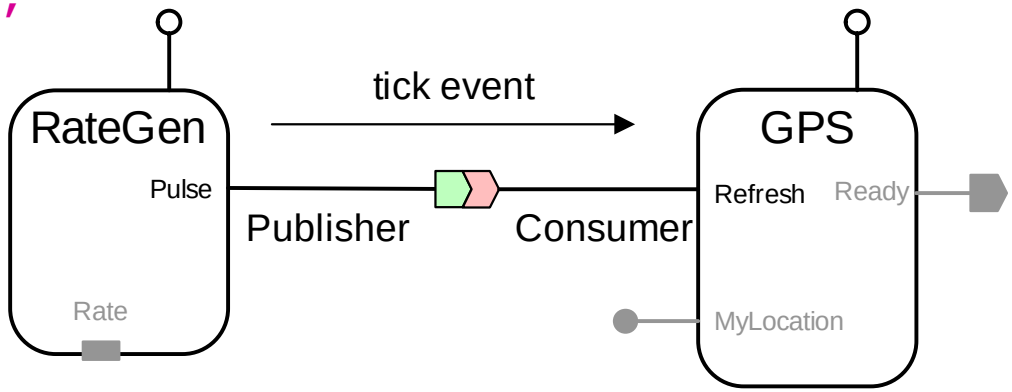
Positioning Sensor

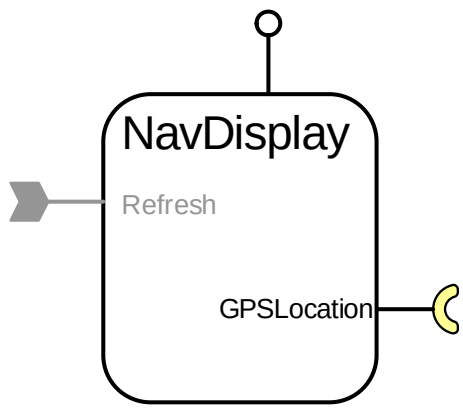
Display Device




- ***Rate Generator***
  - Sends periodic **`Pulse`** events to consumers
- ***Positioning Sensor***
  - Receives **`Refresh`** events from suppliers
  - Refreshes cached coordinates available thru **`MyLocation`** facet
  - Notifies subscribers via **`Ready`** events
- ***Display Device***
  - Receives **`Refresh`** events from suppliers
  - Reads current coordinates via its **`GPSLocation`** receptacle
  - Updates display

```
// IDL 3
typedef unsigned long rateHz;
eventtype tick{
  public rateHz Rate;
};

interface position{
  long get_pos ();
};

component RateGen{
    attribute rateHz Rate;
    publishes tick Pulse;
};
home RateGenHome manages RateGen {};

component GPS {
        provides position MyLocation;
        consumes tick Refresh;
        publishes tick Ready;
  };
home GPSHome manages GPS {};

component NavDisplay{
        uses position GPSLocation;
        consumes tick Refresh
};
home NavDisplayHome manages NavDisplay {};
```

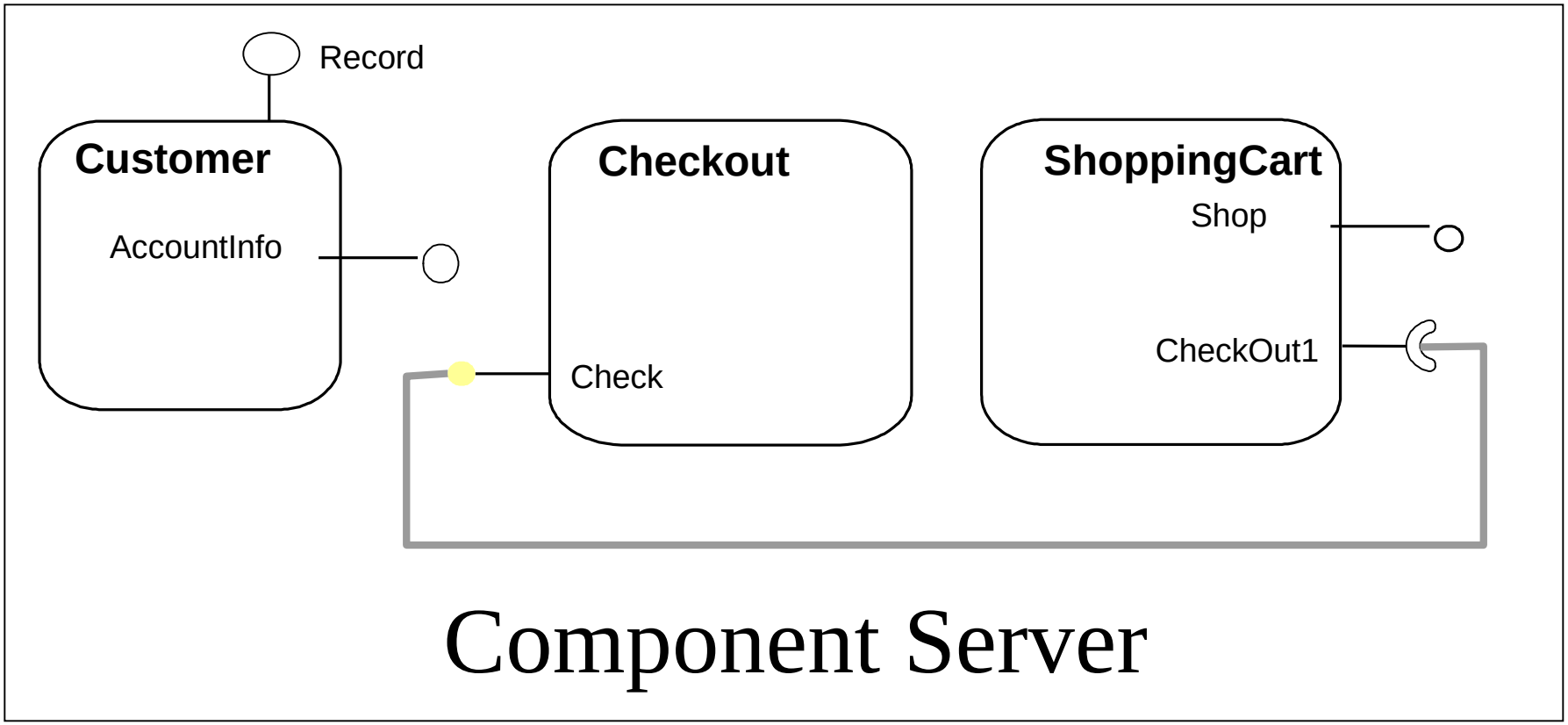


```
// Equivalent IDL 2

valuetype tick : Components::EventBase{
        public rateHz Rate;
};

interface tickConsumer:Components::EventConsumerBase {
        void push_tick (in tick the_tick);
};

interface RateGen : Components::CCMObject {
        attribute rateHz Rate;
        Components::Cookie subscribe_Pulse(in tickConsumer c);
        tickConsumer unsubscribe_Pulse(in Components::Cookie ck);
};

interface GPS : Components::CCMObject{
        position provide_MyLocation ();
        tickConsumer get_consumer_Refresh ();
        Components::Cookie subscribe_Ready(in tickConsumer c);
        tickConsumer unsubscribe_Ready(in Components::Cookie ck);

};
```

```
//Equivalent IDL2 (ctd.)

interface NavDisplay : Components::CCMObject{
        void connect_GPSLocation (in position c);
        position disconnect_GPSLocation();
        position get_connection_GPSLocation ();

        tickConsumer get_consumer_Refresh ();
 };
```

# Exercise 5: Shopping Cart Application



```
//IDL3
module store{
        interface Record {
                string  name();
                string  address();
                string creditCard();
        };

        interface Account {
                long id();
                double balance();
                string history();
        };

        struct Item {
                long thisItem;
                long qty;
                double price;
        };
typedef sequence<Item> ItemList
```

//IDL3 (ctd.)

```
        interface Shopping {
                void setCust(long custid);
                void add(Item selectedItem);
                long remove(long itemNum);
                void buy();
        };

        interface CheckoutIntf {
                boolean buy(in Customer cust, in ItemList cartStuff);
        };
        component Customer support Record {
                provides Account AccountInfo;
        };
        home CustomerHome manages Customer{};

        component Checkout {
                provides CheckoutIntf Check;
        };
        home CheckoutHome manages Checkout {};

        component ShoppingCart {
                provides Shopping Shop;
                uses CheckoutIntf Checkout1;
        };
        home ShoppingCartHome manages ShoppingCart {};
};
```

# CORBA Trading Service

## *Rationale*

- Locating objects in location transparent way
- Naming simple but may not be suitable when
  - clients do not know server
  - there are multiple servers to choose from
- Trading supports locating servers based on service functionality and quality
- Naming ⇔ White pages
- Trading ⇔ Yellow Pages

# *Trading Operation*

•Trader operates as broker between client and server.

• Enables client to change perspective from ´who?´ to ´what?´

• Similar ideas in:

– mortgage broker

– insurance broker

•Clients ask trader for

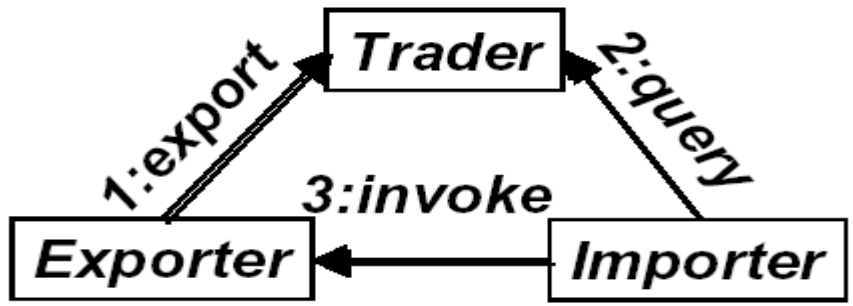
– a service of a certain type

– at a certain level of quality

•Trader supports

– service matching

– service shopping




•Server registers service with trader.

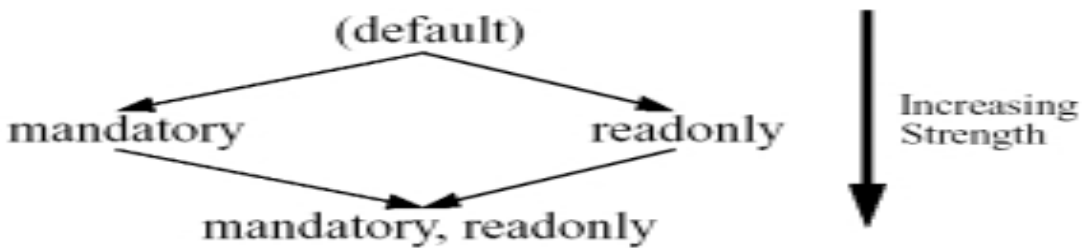
• Server defines assured quality of service:

–Static QoS definition

–Dynamic QoS definition

# *Service Type Definition*

- Service types define
  - Functionality provided by a service and
  - Qualities of Service (QoS) provision.
- Functionality defined by object type
- QoS defined based on properties, i.e.
  - property name
  - property type
  - property value
  - property mode
    - mandatory/optional
    - readonly/modifiable



# Exercise 4: Telephone Service

**Factors affecting the Load balancing**

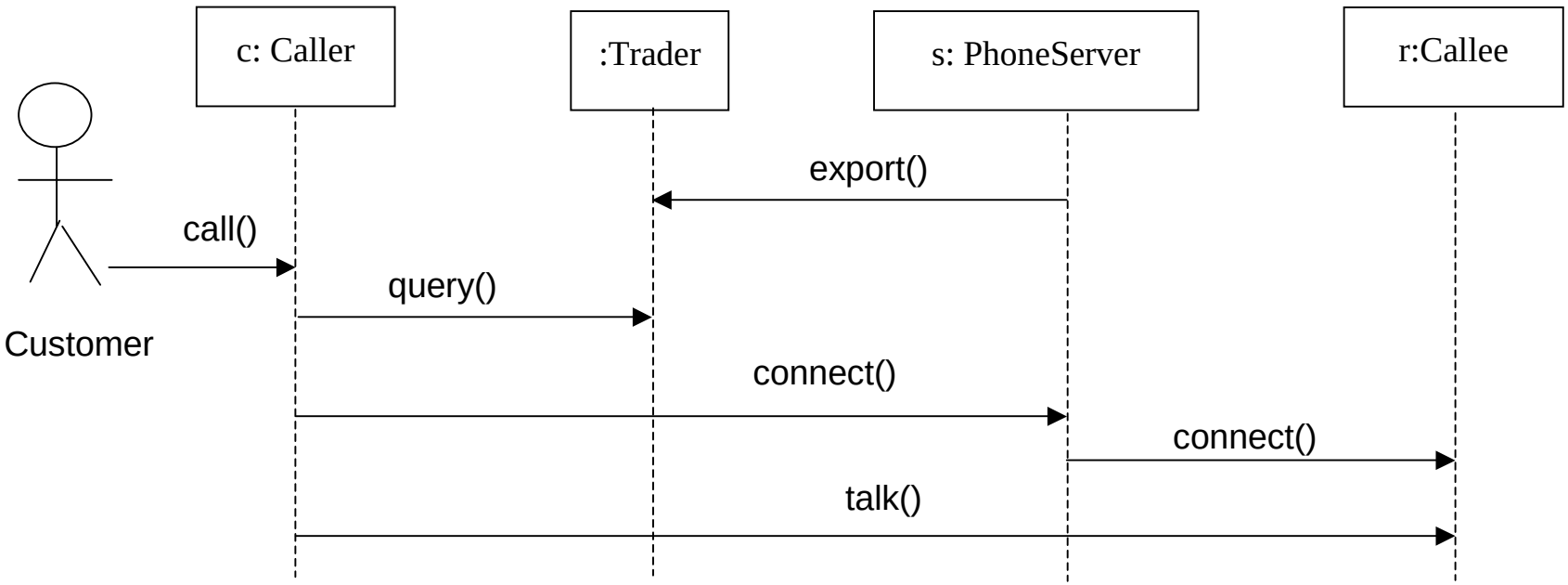
-Phone servers register with Trader by specifying their service type
-Client query the Trader by specifying their needs
-Trader matches clients needs with available services

Phone service: location, VoIP protocol,Price, Link quality

```
typedef enum {MGCP, H323, SIP} Protocol;
interface PhoneServer {…}
service internet_phone {
        interface PhoneServer;
        readonly mandatory property long location;
        readonly mandatory property Protocol voip_prot;
         readonly mandatory property double price;
        readonly mandatory property LinkQuality quality;
}
```

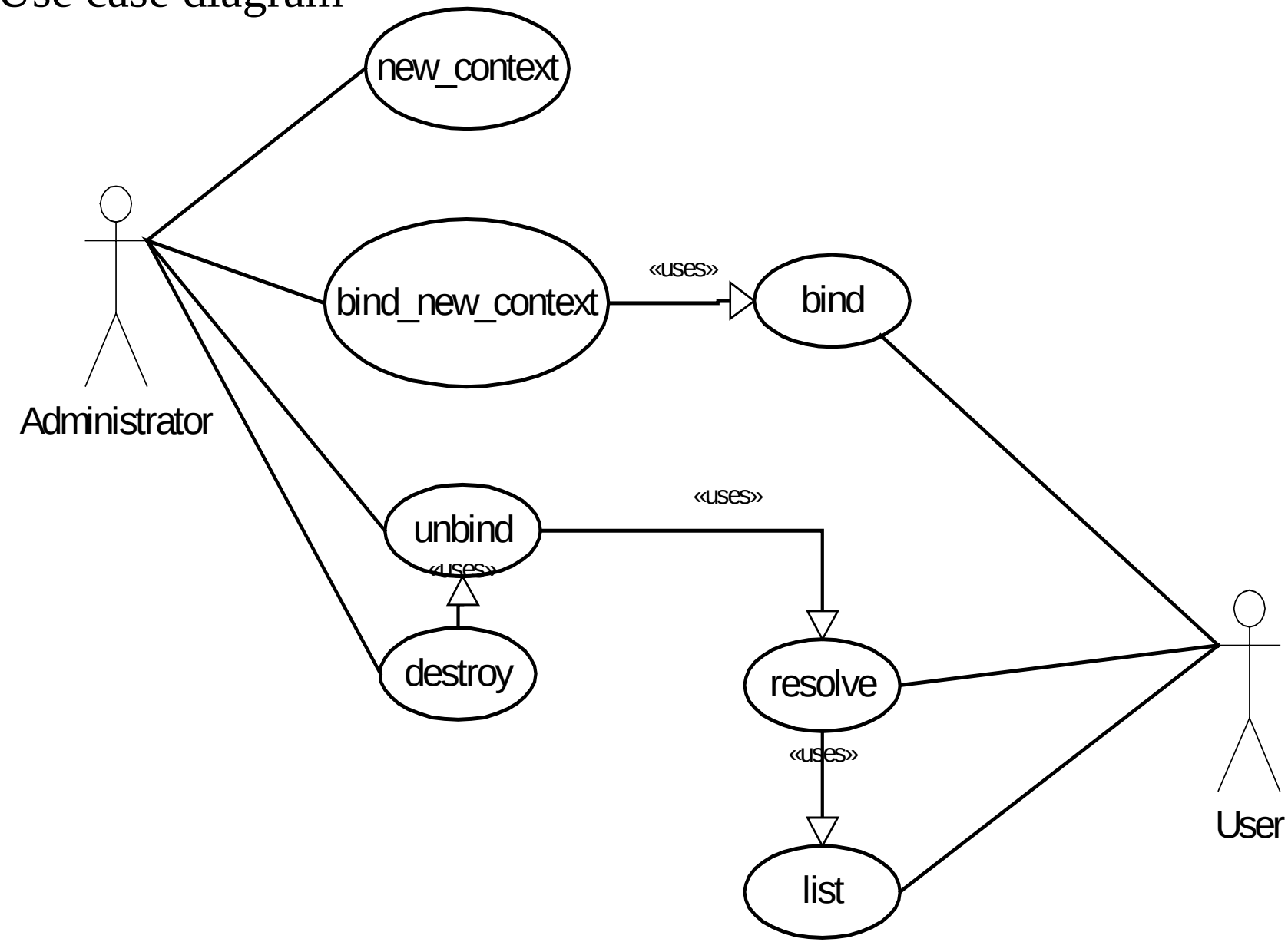
Client: area code, destination number, VoIP protocol, service plan

# Call Scenario



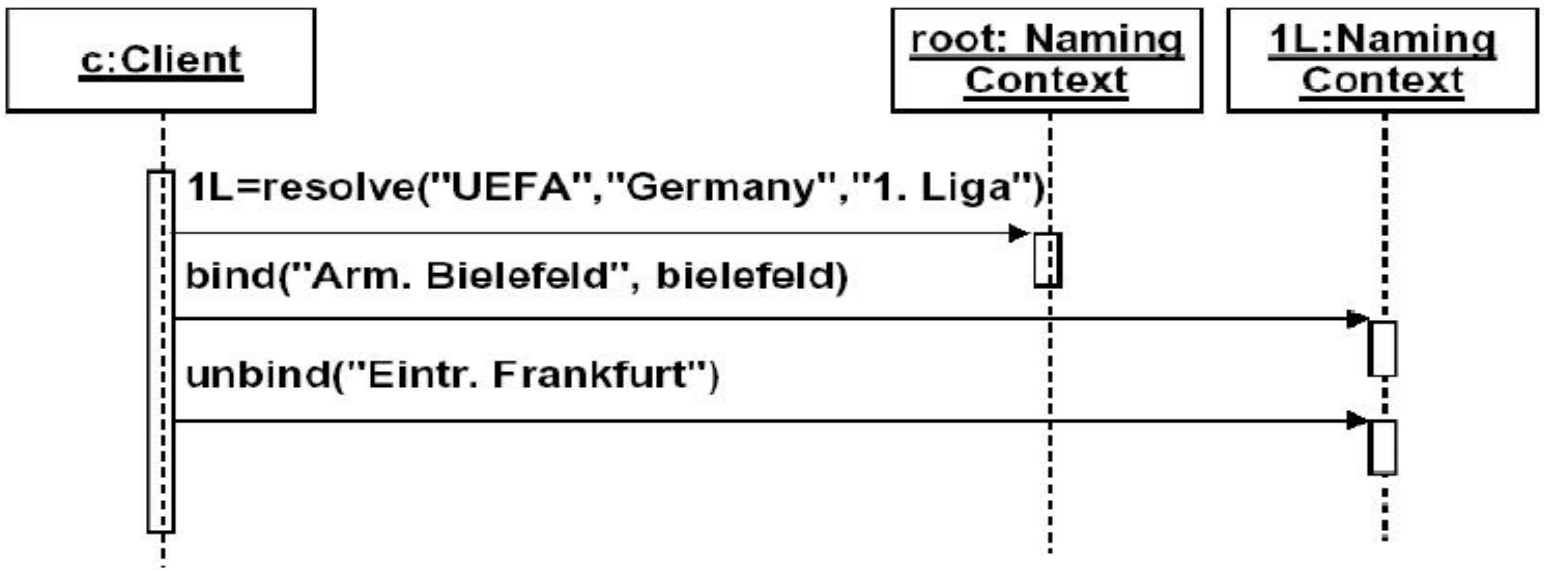
# Exercise 6: CORBA Naming Scenarios

Use case diagram



# Naming Scenario: Binding

**Promote Bielefeld to German '1. Liga' and relegate Frankfurt**



# Naming Scenario: Resolving

**Print squad of Borussia Dortmund**