|  |  |
| --- | --- |
|  | **Royal Institute of Technology**  **MSc. Software Engineering of Distributed Systems** |

ID2209 Distributed Artificial Intelligence and Intelligent Agents

Project

|  |
| --- |
|  |
|  |
| Andrei Shumanski | Trigonakis Vasileios |
| andreish@kth.se | vtri@kth.se |
| 00460707761992 | 00460707694420 |

*Stockholm 2009*

Table of Contents

[Task 1 4](#_Toc247653194)

[Answer 4](#_Toc247653195)

[Task 2 19](#_Toc247653196)

[Answer 19](#_Toc247653197)

[Level 1 representation for overall System 19](#_Toc247653198)

[Level 2 representation of Agent interactions using Sequence Diagrams 20](#_Toc247653199)

[Level 3 representation of Agent behaviors using State-chart diagrams 21](#_Toc247653200)

[Task 3. 22](#_Toc247653201)

[Answer 22](#_Toc247653202)

[Task 4. 24](#_Toc247653203)

[Answer 24](#_Toc247653204)

[Task 4.1 Augment the Analysis phase of GAIA with role binding 24](#_Toc247653205)

[Task 4.2 In this case, perform role-based modeling first and then proceed to GAIA analysis phase. 26](#_Toc247653206)

[Task 4.3 Commend on differences in resultant design of 4.1 and 4.2. 27](#_Toc247653207)

[Task 5. 28](#_Toc247653208)

[Answer 28](#_Toc247653209)

[Level 0 Analysis 28](#_Toc247653210)

[Level 1 Analysis 32](#_Toc247653211)

[Task 6 43](#_Toc247653212)

[Answer 43](#_Toc247653213)

[Compare MESSGE UML and GAIA without Role based modeling 43](#_Toc247653214)

[Compare MESSGE UML and GAIA with Role based modeling 44](#_Toc247653215)

# Task 1

Model your system via GAIA AOSE Methodology (tricky part would be modeling mobility)

## Answer

The following tables describe roles in the system:

|  |
| --- |
| Role Schema:  MANUFACTURER |
| Description:  Manufacturer role creates products and provides products. |
| Protocols and Activities:  CreateProduct, ProvideProducts, AwaitRequest |
| Permissions:  **reads supplied** orderDetails //reads request for availableproducts  **geneates** productsList **//** generates list of available products |
| Responsibilities:  Liveness:  Manufacturer=(CreateProduct)ω||(AwaitRequest.ProvideProducts )ω  Safety:   * **true** |

|  |
| --- |
| Role Schema:  INVENTORY |
| Description:  Inventory keeps list of available products. |
| Protocols and Activities:  AvailabilityRequest, ProvideProductsAvailability, ReadProductsOrder, GetProducts, UpdateProductsList |
| Permissions:  **reads supplied** productsOrder //reads products  **supplied** newProductsList //reads new products list  availableProducts //reads currently available items  **changes** availableProducts //updates list of available items |
| Responsibilities:  Liveness:  Inventory = (AvailabilityRequest . ProvideProductsAvailability )ω ||[ReadProductsOrder.GetProducts .UpdateProductsList] ω  Safety:   * infoAvailable(availableProducts) |

|  |
| --- |
| Role Schema:  PRICING |
| Description:  Pricing role tells a price of a selected product. |
| Protocols and Activities:  ICNPResponce, checkProductAvailability, calculatePrice |
| Permissions:  **reads supplied** productAvailability //checks product availabilty  **geneates** productPrice **//** generates price for a given product |
| Responsibilities:  Liveness:  Pricing = (checkProductAvailability. calculatePrice. ICNPResponce)ω  Safety:   * productsAvailable=0→ price=null |

|  |
| --- |
| Role Schema:  MONITORING |
| Description:  Monitors the quantity of the products and order from manufacturers when needed. |
| Protocols and Activities:  MonitorAvailability, GenerateProductOrder, RequestProducts, InformIncomingProducts |
| Permissions:  **reads** availabilityLimit //the lower availability for the product  **supplied** currentAvailability //the availability that the inventory has  **generate** productOrder //the order that will be sent to manufacturer |
| Responsibilities:  Liveness:  Monitoring = ((MonitorAvailability . [SendOrder])ω  SendOrder = GenerateProductOrder, RequestProducts . InformIncomingProducts  Safety:   * availabilityLimit < currentAvailability → productOrder = null |

|  |
| --- |
| Role Schema:  YELLOWPAGES |
| Description:  Accepts subscriptions of “shops”, in order to be able to provide information about the available “shops” and their location. |
| Protocols and Activities:  SubsctriptionRequest, SubscribeShop, AvailableShopsRequest, AvailableShopsResponse |
| Permissions:  **reads** **supplied**  shopDetails //the details of a shop that wants to subscribe  **supplied**  CustomerRequirements //the requirement for selecting shops  **update** shopsList //the availability that the inventory has  **generates** specificShopList //the shop list for this case |
| Responsibilities:  Liveness:  YellowPages = ((SubsctriptionRequest . SubscribeShop) ||  (AvailableShopsRequest . AvailableShopsResponse))ω  Safety:   * infoAvailable(shopDetails) |

|  |
| --- |
| Role Schema:  CUSTOMER |
| Description:  The one that wants to buy a product, initializes the whole buy process |
| Protocols and Activities:  MakeCall, GiveRequirements, ReceiveBestOffer |
| Permissions:  **reads** **supplied**  result //the results for the shopping request  **generates** customerRequirements //the requirements for the request |
| Responsibilities:  Liveness:  Customer = (MakeCall . GiveRequirements .ReceiveBestOffer)+  Safety:   * infoAvailable(customerRequirements) * customerRequirements != null |

|  |
| --- |
| Role Schema:  SHOPPINGMANAGER |
| Description:  Accept the request form a client and starts the buying process by finding the possible shops and take care that the protocol will run “on” them |
| Protocols and Activities:  AwaitsRequest, RequestRequirements, FindShopsDetails, InitiateShopperModules, AwaisForResponses, CalculateBestOffer, InformClient |
| Permissions:  **reads** **supplied**  customerRequirements //the requirements for the request  **supplied**  shopsDetails //the details of all available shops that can be //explored  **supplied**  shoppingResults //the results from all shops  **generates** specificRequest //the specific data that it will provide to the //agents //that will search for the price  **generates** bestOffer //the best offer found |
| Responsibilities:  Liveness:  ShoppingManager = (AwaitsRequest . RequestRequirements . FindShopsDetails . [((InitiateShopperModule)+) . ((AwaisForResponses)+) . CalculateBestOffer . InformClient])ω  Safety:   * customerRequirements != null * shoppingResults != null → bestOffer != null * shopsDetails = null → bestOffer = null |

|  |
| --- |
| Role Schema:  SHOPPING |
| Description:  Moves to the shop place, gets the best price possible, returns back to the starting location announcing the best price found |
| Protocols and Activities:  Initialize, MoveToLocation, ICNPInitiate, ICNPAcceptProposal, MoveBack, AnnounceResults |
| Permissions:  **reads** **supplied**  specificRequest //the requirements for the request and //specifically the target price and location  **updates** currentLocation //the location in which it resides  currentOffer //the offer that is currently negotiating  **generates** bestOffer //the best offer found |
| Responsibilities:  Liveness:  Shopping = (Initialize . MoveToLocation . ICNPInitiate . MoveBack . AnnounceResults)ω  Safety:   * infoAvailable(specificRequest) * currentLocation != null |

After defining roles we can describe associated interaction models for these roles:

#### SHOPPINGMANAGER

##### RequestRequirements

|  |  |  |  |
| --- | --- | --- | --- |
| RequestRequirements | |  |  |
| SHOPPINGMANAGER | CUSTOMER |  |  |
| Ask for the requirements of the customer | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| GiveRequirements | |  |  |
| CUSTOMER | SHOPPINGMANAGER |  |  |
| Provide the requirements | |  | customerRequirements |

##### InitiateShopperModule

|  |  |  |  |
| --- | --- | --- | --- |
| SelectShopper | |  |  |
| SHOPPINGMANAGER | SHOPPING |  |  |
| Select the agent that will send to a specific shop | |  | customerRequirements  shopsDetails |

|  |  |  |  |
| --- | --- | --- | --- |
| Initialize | |  |  |
| SHOPPINGMANAGER | SHOPPING |  | specificRequest |
| Initialize the agent | |  |  |

##### FindShopsDetails

|  |  |  |  |
| --- | --- | --- | --- |
| AvailableShopsRequest | |  |  |
| SHOPPINGMANAGER | YELLOPAGES |  |  |
| Ask for the available shop agents | |  | customerRequirements |

|  |  |  |  |
| --- | --- | --- | --- |
| AvailableShopsResponse | |  |  |
| YELLOPAGES | SHOPPINGMANAGER |  | customerRequirements  shopsList |
| Provide the available shop agents | |  | specificShopList |

#### MONITORING

##### MonitorAvailability

|  |  |  |  |
| --- | --- | --- | --- |
| AvailabilityRequest | |  |  |
| MONITORING | INVENTORY |  |  |
| Ask for the availability of the product | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| ProvideProductsAvailability | |  |  |
| INVENTORY | MONITORING |  | availableProducts |
| Provide the availability | |  | currentAvailability |

##### RequestProducts

|  |  |  |  |
| --- | --- | --- | --- |
| AwaitRequest | |  |  |
| MONITORING | MANUFACTURER |  |  |
| Request to the manufacturer to provide products | |  | productOrder |

|  |  |  |  |
| --- | --- | --- | --- |
| ReadProductsOrder | |  |  |
| MONITORING | INVENTORY |  | productOrder |
| Informs the inventory that it will receive products from manufacturer | |  | productsOrder |

|  |  |  |  |
| --- | --- | --- | --- |
| GetProducts | |  |  |
| MANUFACTURER | INVENTORY |  | orderDetails |
| It sends the products | |  | newProductsList |

#### Shopping

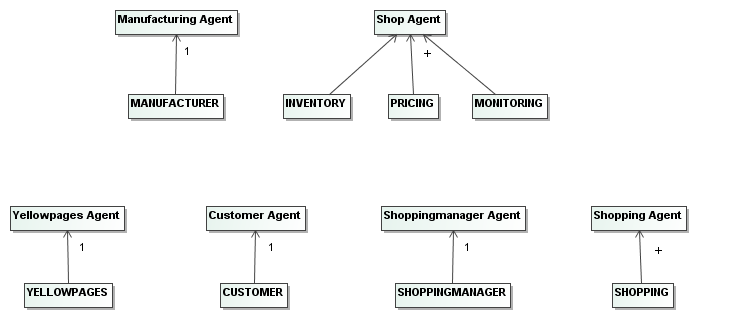
##### Price Negotiation

|  |  |  |  |
| --- | --- | --- | --- |
| ICNPInitiate | |  |  |
| SHOPPING | PRICING |  |  |
| SHOPPING sends cfp message with maxPrice and receives responses | |  | ProposedPrice |

|  |  |  |  |
| --- | --- | --- | --- |
| ICNPResponse | |  |  |
| PRICING | SHOPPING |  | ProposedPrice |
| PRICING sends response on the given price | |  | Responses |

|  |  |  |  |
| --- | --- | --- | --- |
| ICNPAcceptProposal | |  |  |
| SHOPPING | PRICING |  | Responses |
| SHOPPING decide to accept proposal and informs PRICING | |  | AcceptProposal |

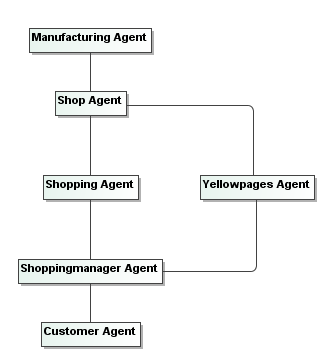
The agent model:



The following table describes identified services:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Agent | Service | Inputs | Outputs | Pre-condition | | Post-condition |
| Customer Agent | Place an order | customerRequirements | result | customerRequirements != null | | true |
| Shoppingmanager Agent | Accept request | null | customerRequirements | true | | customerRequirements != null |
| Shoppingmanager Agent | Find available shops | customerRequirements | shopsDetails | customerRequirements != null | | true |
| Shoppingmanager Agent | Promote the order | shopsDetails | shoppingResults | shopsDetails != null | | true |
| Shoppingmanager Agent | Calculate the best offer | shoppingResults | bestOffer | shoppingResults != null | | bestOffer != null |
| Shoppingmanager Agent | Inform the user about the result | bestOffer | null | bestOffer!= null | | true |
| Shopping Agent | Prepare for negotiation | specificRequest | currentLocation, currentOffer | specificRequest != null | | currentLocation != null, currentOffer != null |
| Shopping Agent | Negotiate | specificRequest | bestOffer | specificRequest != null | | true |
| Shopping Agent | Finish negotiation | bestOffer | null | true | | true |
| Manufacturing Agent | Product creation | null | Product | True | Product ≠null | |
| Manufacturing Agent | Provide products list | Avilability request | Products List | Avilability request ≠ null | Products List ≠ null ∧ Product ∈ Products List | |
| Shop Agent | Price calculation | Product Availability | Product Price | Product Availability≠null ∧  Product availability>0 | Product Price≠null | |
| Shop Agent | Check of product availability | Product Name | Product Availabiliy | Product Name≠null | Product Availabiliy ≠null | |
| Shop Agent | ICNPResponce | Proposed Price | Response to proposed price | Product availability>0 | Response to proposed price≠null | |
| Shop Agent | Update Products List | null | Products List | True | Products List ≠ null | |

The acquaintance model diagram:



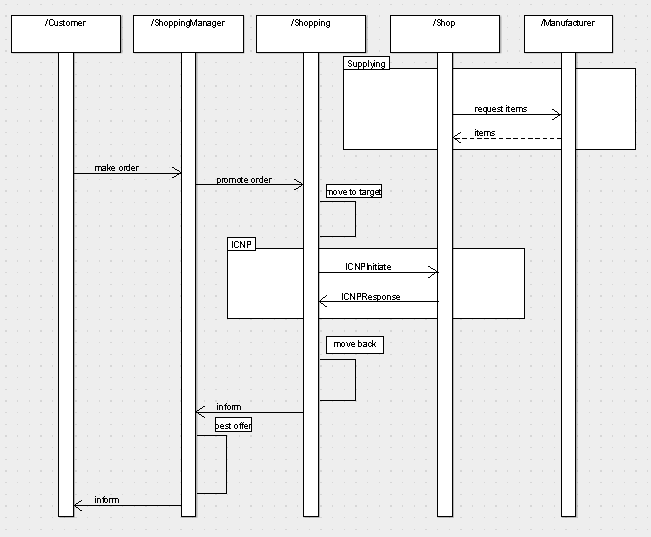
# Task 2

Model interactions among agents in UML

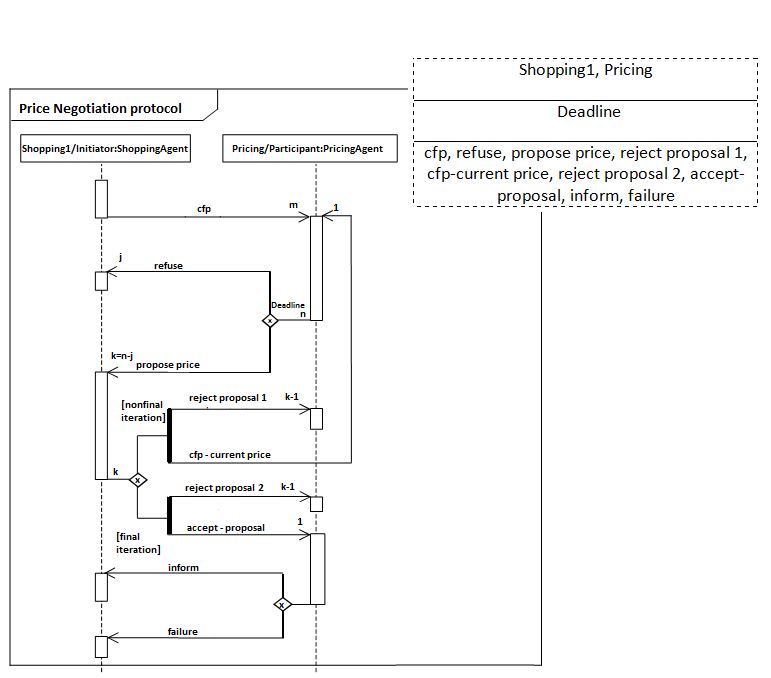
## Answer

### Level 1 representation for overall System

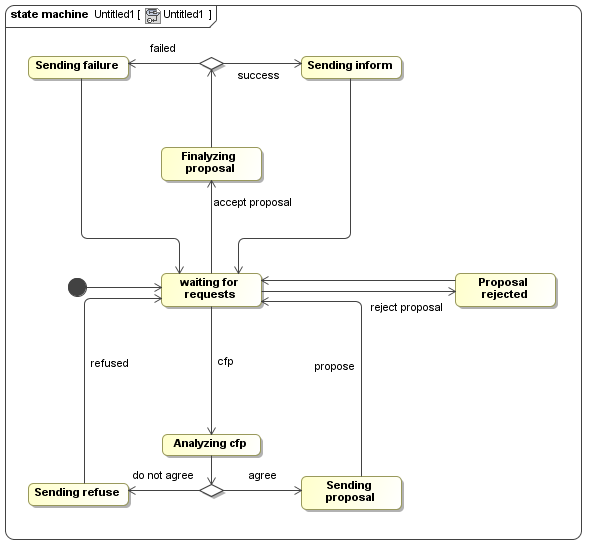
The following diagram shows level 1 representation of the system:



### Level 2 representation of Agent interactions using Sequence Diagrams



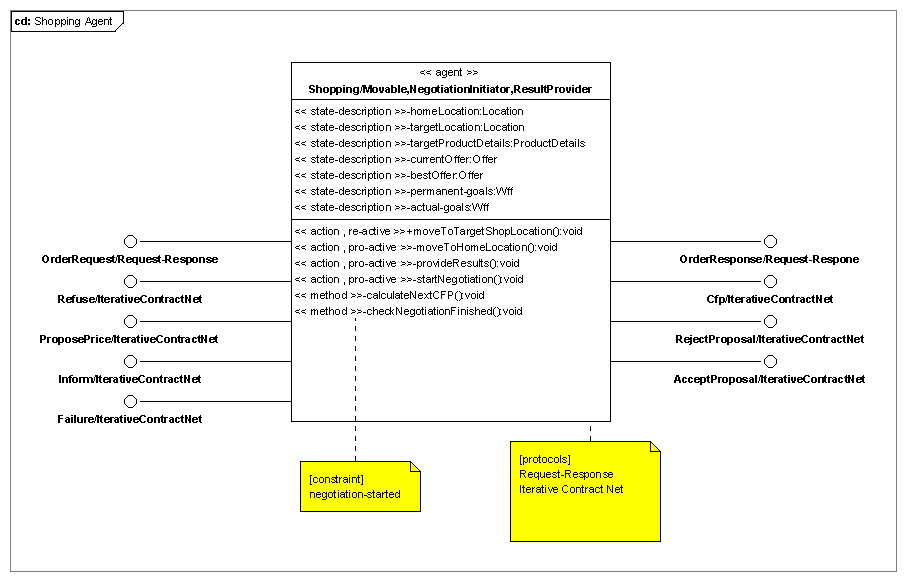
### Level 3 representation of Agent behaviors using State-chart diagrams

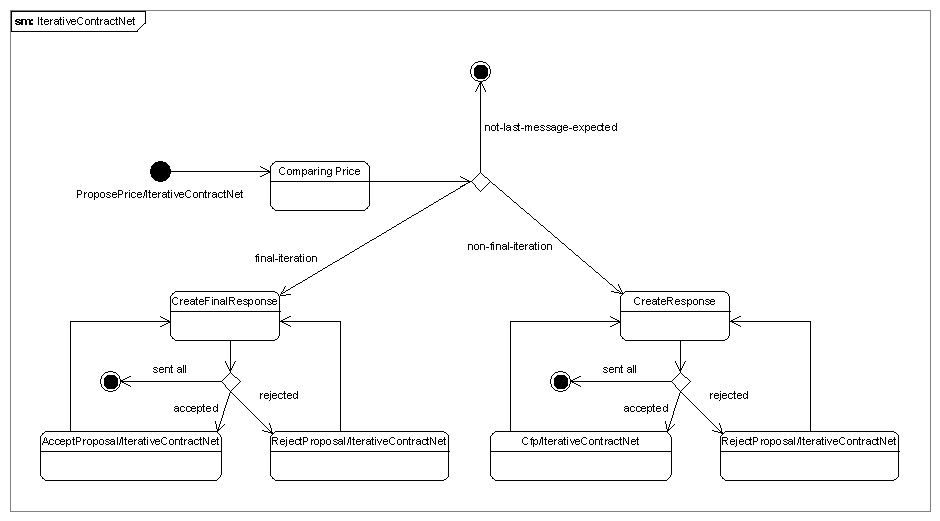


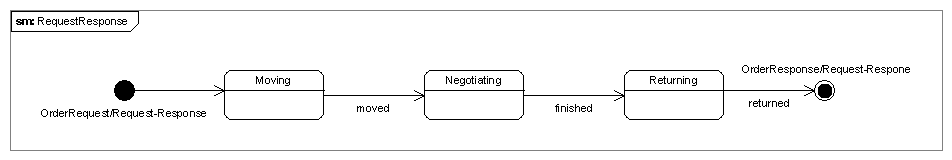
# Task 3.

Use UML Class diagrams to design behavior of your agents.

## Answer





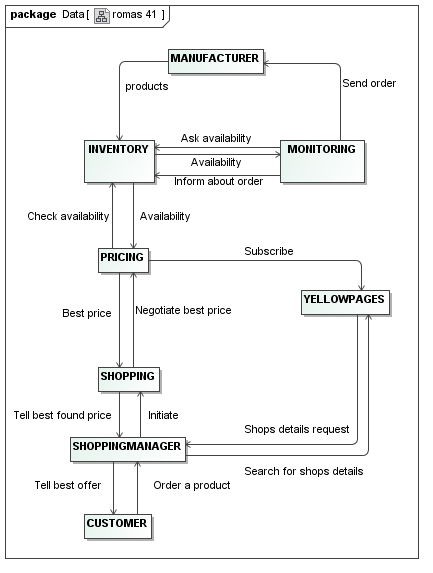


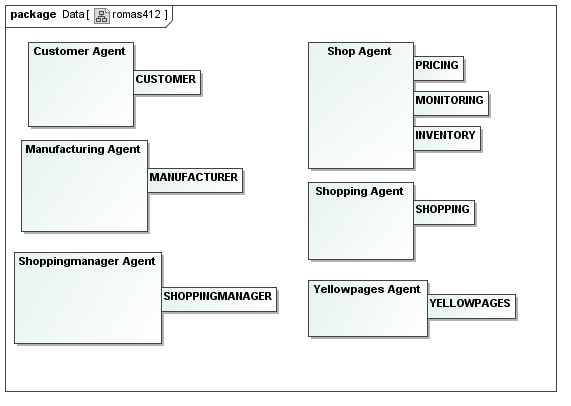
# Task 4.

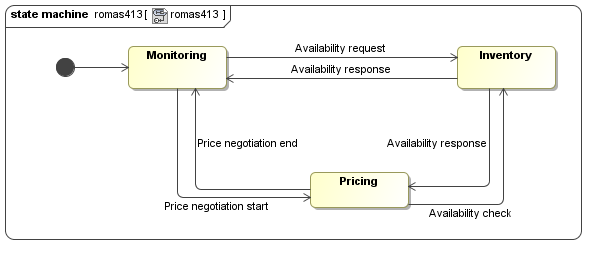
Model your system using Role based modeling approach

## Answer

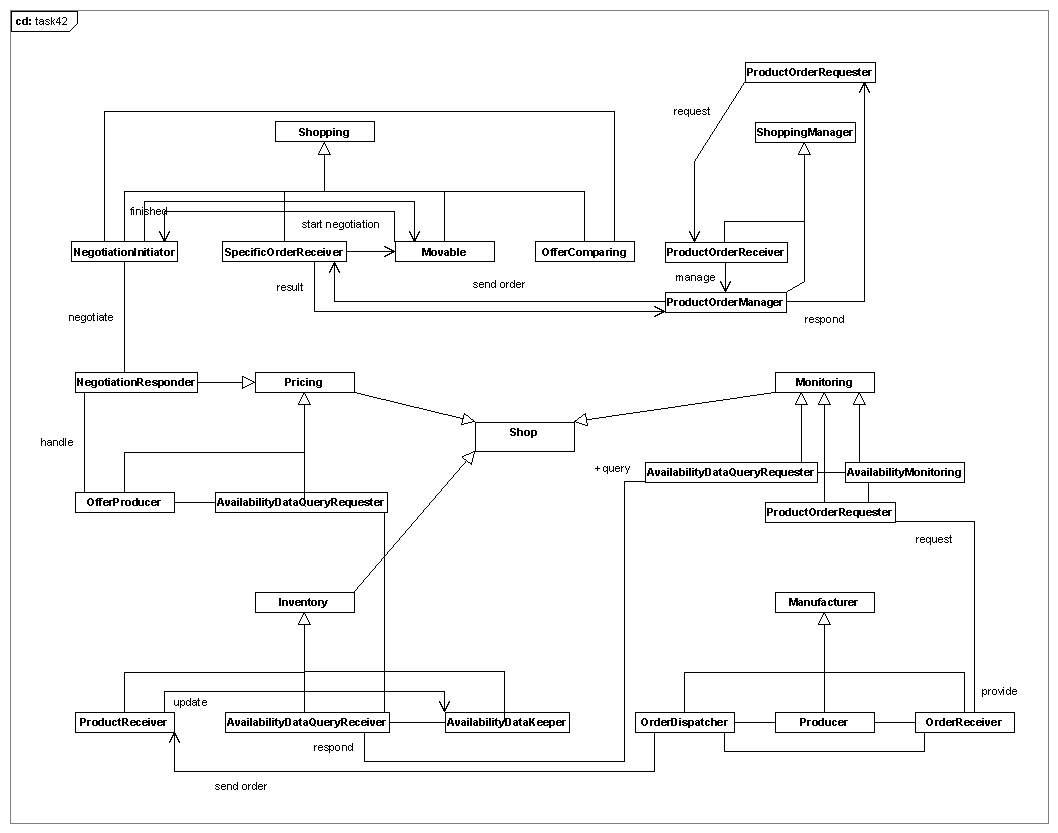
### Task 4.1 Augment the Analysis phase of GAIA with role binding







### Task 4.2 In this case, perform role-based modeling first and then proceed to GAIA analysis phase.



### Task 4.3 Commend on differences in resultant design of 4.1 and 4.2.

Both approaches, provided a more detailed view about the Roles that we created during using the GAIA methodology. As expected, the resultant design of task 4.2 was much more detailed comparing to the one at 4.1.

On task 4.1 we used the already generated Roles from GAIA methodology and we created the diagrams that role-based modeling proposes. This process, of course, helped to increase the details level and to make the usability and of the roles more clear, but did not lead to creating more roles.

On the other hand, on task 4.2, we first used the role-base modeling approach and then we considered how the GAIA methodology would be. In this case, since the first step in our process was extracting roles from use cases, make role hierarchy and find the roles' relationship the result was more fine grained roles, that also means more and more well defined roles. Of course, after role based methodology, almost the whole analysis part for GAIA methodology was ready.

Concluding, we thought that following the process of task 4.1 is almost pointless, because the role-based approach is not that suitable if you have your roles already defined. On the other hand, the process of the task 4.2 looks more rational, because it helped us create more precose models in the GAIA methodology.

# Task 5.

Re-model the entire system using MESSAGE UML

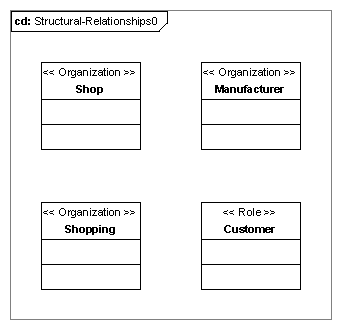
## Answer

### Level 0 Analysis

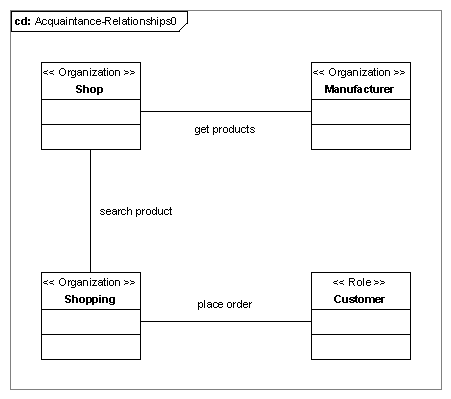
We start from level 0 analysis of the system.

#### Organization view

Structural relationships in level 0 organization diagram:

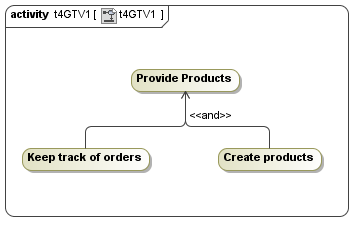


Acquaintance relationships in level 0 organization diagram:

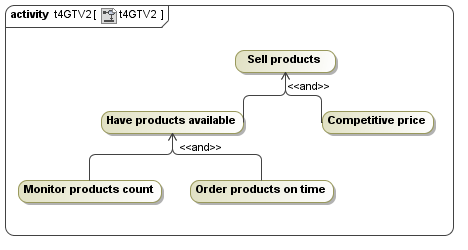


#### Goal/Task view

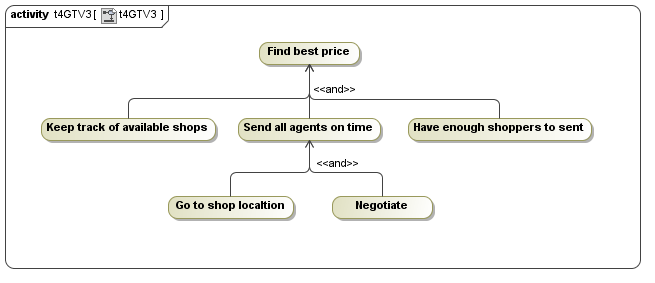
Goal decomposition diagram for Manufacturer:



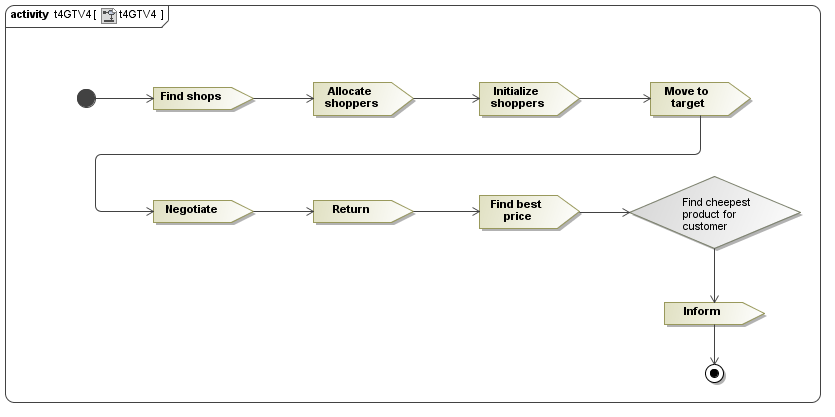
Goal decomposition diagram for Shop:



Goal decomposition diagram for Shopper:



Workflow of task implementing Find-Cheapest-Product service:

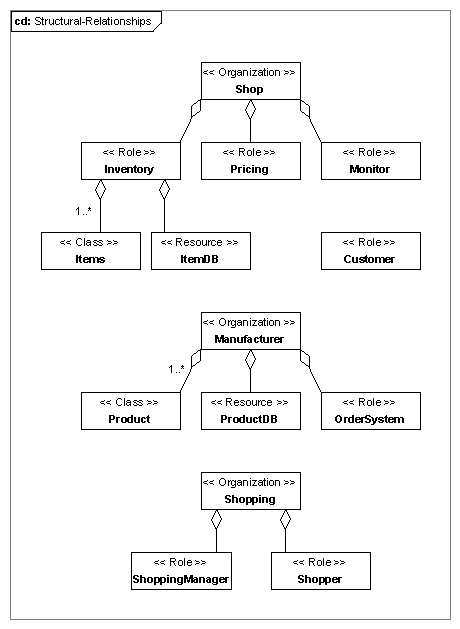


### Level 1 Analysis

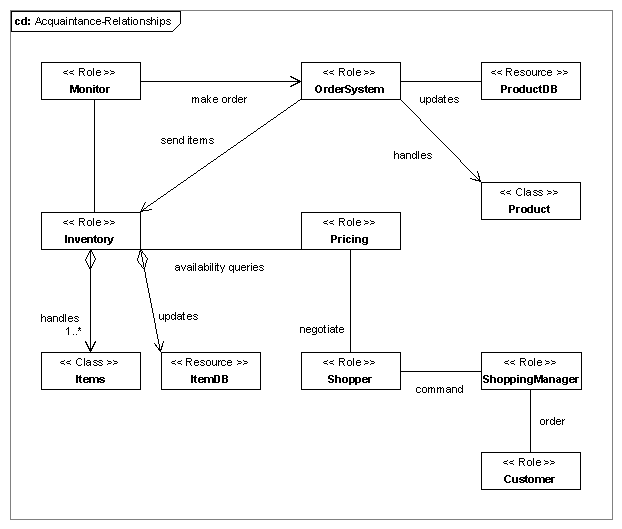
Level 1 analysis focuses on the system itself identifying at a glance the main pieces of functionality.

#### Organization view

Structural relationships in level 1 organization diagram:

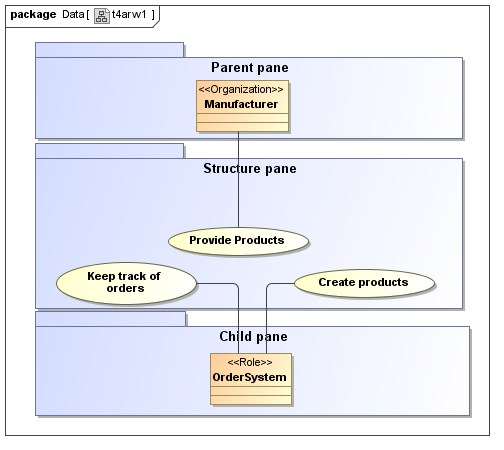


Acquaintance relationships in level 0 organization diagram:

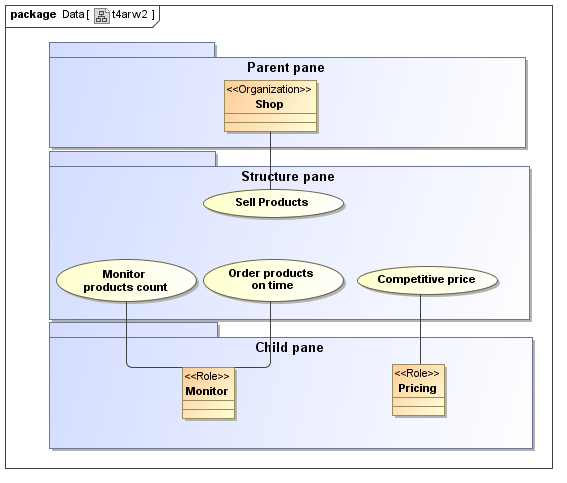


#### Agent/Role view

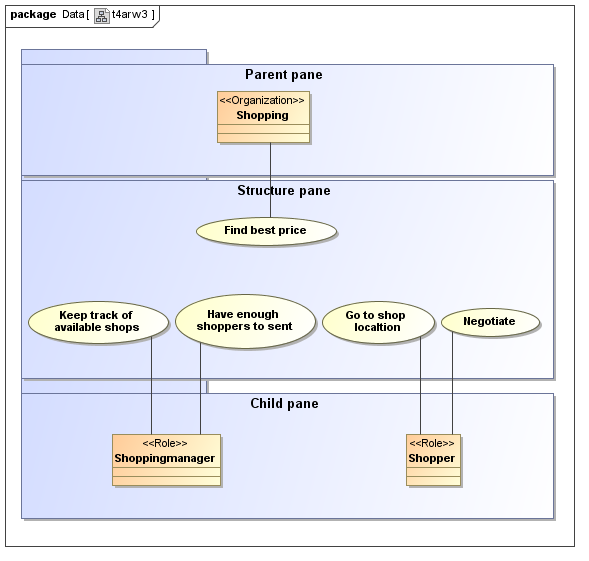
Delegation structure diagram for Manufacturer:



Delegation structure diagram for Shop:



Delegation structure diagram for Shopping:



##### Agent/Role schemas:

|  |
| --- |
| Role Schema:  Inventory |
| GOALS:  - |
| CAPABILITIES:  Organizing capabilities required. |
| KNOWLEDGE, BELIEFS:  The items that the shop currently has is kept by the inventory. It is a passive role, so it does not have any beliefs. |
| AGENT REQUIREMENTS:  This role will be played by the shop agent, which requires to keep track of the products that it currently has. |

|  |
| --- |
| Role Schema:  Pricing |
| GOALS:  Competitive price |
| CAPABILITIES:  Ability to research the market and to make good price selection choices for the product, according to this research and to the availability of the product. |
| KNOWLEDGE, BELIEFS:  Information about the prices that the products are bought by the manufacturers and how to get the availability of the products are needed. Also, has some strategy in order to be able to select a price for the products. |
| AGENT REQUIREMENTS:  This role will be played by the shop agent, which requires selecting a price for the products that it wants to sell. |

|  |
| --- |
| Role Schema:  Monitor |
| GOALS:  Monitors product count, Place order on time |
| CAPABILITIES:  The agent should be able to monitor the stocks of the products, to sense when they are low and to place an order to the manufacturer. |
| KNOWLEDGE, BELIEFS:  It should know how often to check the availability of the products and when it should consider that the shop is low in stock, so it should made an order. For the aforementioned it should have a strategy. |
| AGENT REQUIREMENTS:  This role will be played by the shop agent, which requires keeping the stock of the products that it sells in a “good” level. |

|  |
| --- |
| Role Schema:  Customer |
| GOALS:  Buy product |
| CAPABILITIES:  It should be able to place an order. |
| KNOWLEDGE, BELIEFS:  It should know and believe that it needs or wants a product and how much it is willing to pay. |
| AGENT REQUIREMENTS:  This role will be played by the customer agent, who wants to buy a product. |

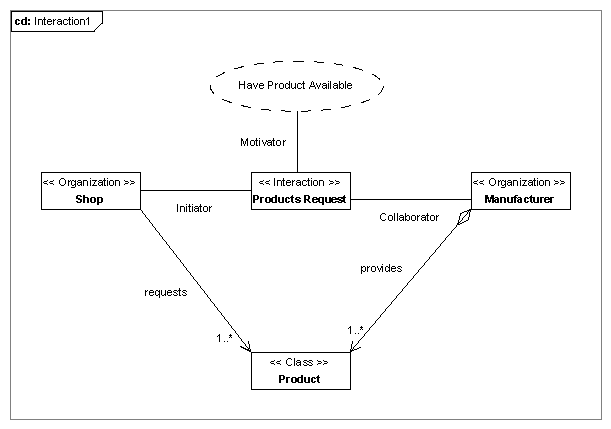
|  |
| --- |
| Role Schema:  OrderSystem |
| GOALS:  Keep track of orders, Create Products |
| CAPABILITIES:  It is required to accept orders and handle them and also to order the creation of products. |
| KNOWLEDGE, BELIEFS:  It has to know the price in which it sells each specific product in a specific quantity and to a specific customer. Also, it should keep a strategy of the quantities of product that it should create. |
| AGENT REQUIREMENTS:  This role will be played by the manufacturer agent, who needs to accept orders and to create products. |

|  |
| --- |
| Role Schema:  ShoppingManager |
| GOALS:  Keep truck of available shops, Have enough shoppers to send |
| CAPABILITIES:  It is required to choose what shoppers to use for the specific request and to handle this specific request to the shopper that will go to the location of the shop to negotiate. Also, it has to gather the responses from the shoppers find the best one and send it to the customer. |
| KNOWLEDGE, BELIEFS:  It should know as many shops as possible. Also, because it has a limited amount of shoppers that it can use, it should have a policy when and how to send shoppers to the shops. |
| AGENT REQUIREMENTS:  This role will be played by the shopping agent, which wants to provide the ability to customer to find products. |

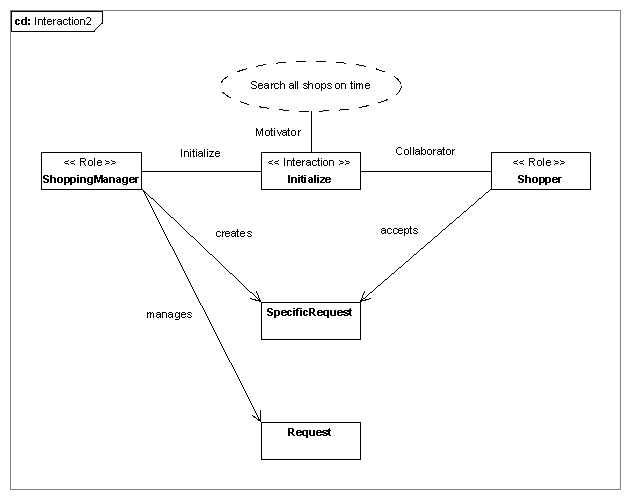
|  |
| --- |
| Role Schema:  Shopper |
| GOALS:  Go to shop location, Negotiate |
| CAPABILITIES:  It is required to have the ability to move to shop’s location and back to the shopping agent location. Also, it should be able to use the iterative contract net protocol to negotiate with the shop. |
| KNOWLEDGE, BELIEFS:  It should know how to move from one location to another. Also, how to negotiate with the shop in order to achieve the best price possible. |
| AGENT REQUIREMENTS:  This role will be played by the shopping agent, which wants to provide the ability to customer to find products in the best price. |

#### Interaction view

Interaction diagram describing Products Request interaction:



Interaction diagram describing Initialize interaction:



# Task 6

Compare MESSAGE UML and GAIA

## Answer

### Compare MESSGE UML and GAIA without Role based modeling

Both of the methods suggest a similar process on how to specify an agent based software system. The agent engineer should start by finding the “roles” that will exist in the system and continue the specification on this first step. Though, even in this step, the two methodologies differ.

In GAIA, the first step was to create the role schema, that means that during this phase we can only create roles. On the other hand, the MESSAGE UML methodology introduces Level 0, Organizational view which uses Roles, Organization of roles, Classes and Resources. So, with MESSAGE we can model more than just the Roles of the system.

Moreover, the approach for going into more detailed diagrams for the two methodologies is different. GAIA, has an analysis and a design phase, while MESSAGE UML introduces different level of analysis, that mean different level of details. GAIA’s approach is more precise, and also if needed, it is possible to repeat the same process again in order to make a refined model.

Overall, from our experience with these two agent oriented engineering approaches we concluded that we “prefer” GAIA over MESSAGE UML. Firstly, it is a more strict approach, since the elements and diagrams that you can use are less and more simple from MESSAGE UML and also, by only introducing Roles and Agents, the engineering avoids thinking about the specific implementation of the System. MESSAGE UML gives you the ability to model simple classes and resources, which leads the engineer to think about business logic implementation details.

### Compare MESSGE UML and GAIA with Role based modeling

First of all, this answer includes the previous answer also. The main difference that we faced was that after using the Role based approach before using GAIA, the Role model that we produced was much more sophisticated. This can be characterized both as a benefit, or a problem of this approach.

We say benefit, because our roles after role based approach were very well defined and so, the GAIA methodology should have faster and better results. On the other hand, the number of roles that we created for the role based methodology was very big, so the GAIA approach afterwards, would be time consuming, tiring and the result more difficult to understand.

So, if we compare them in this level, the MESSAGE UML approach could go to the desired level of details, just by doing one or more levels of analysis. We consider that this would be faster than implementing the whole role based and GAIA approach. The results are expected to be more or less the same, since with MESSAGE UML, on each iteration you can create more specific roles.