## Task 1. Model your system via GAIA AOSE Methodology (tricky part would be modeling mobility)

|  |
| --- |
| 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 |

#### 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 |

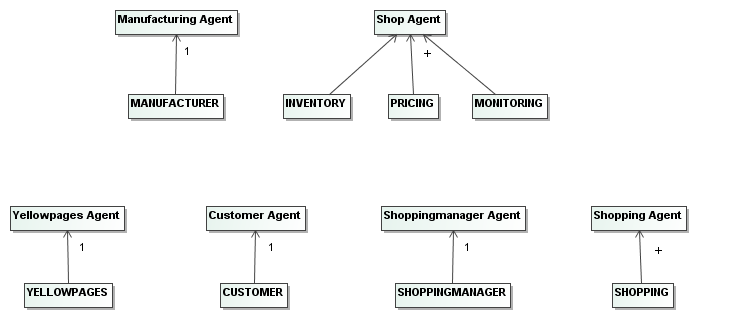
#### Shopping

##### Price Negotiation

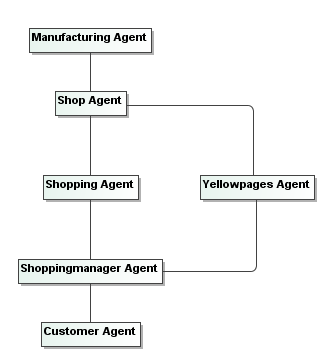
|  |  |  |  |
| --- | --- | --- | --- |
| ICNPInitiate | |  |  |
| SHOPPING | PRICING |  |  |
| SHOPPING sends cfp message with maxPrice and receives responses | |  | ProposedPrice |

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

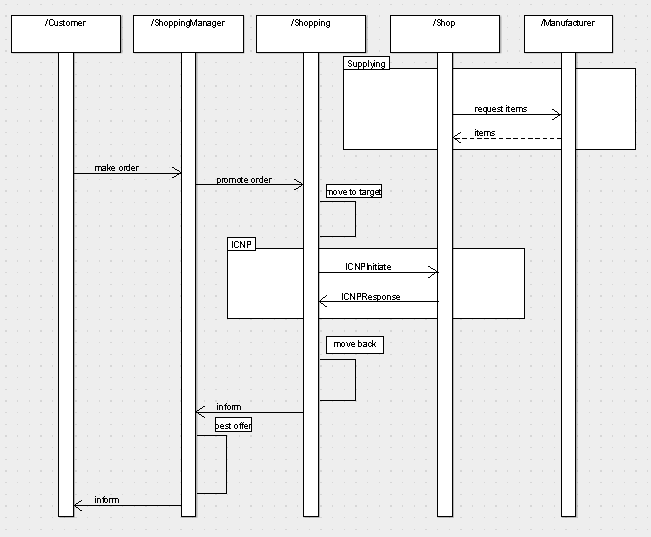
|  |  |  |  |
| --- | --- | --- | --- |
| ICNPAcceptProposal | |  |  |
| SHOPPING | PRICING |  | Responses |
| SHOPPING decide to accept proposal and informs PRICING | |  | AcceptProposal |

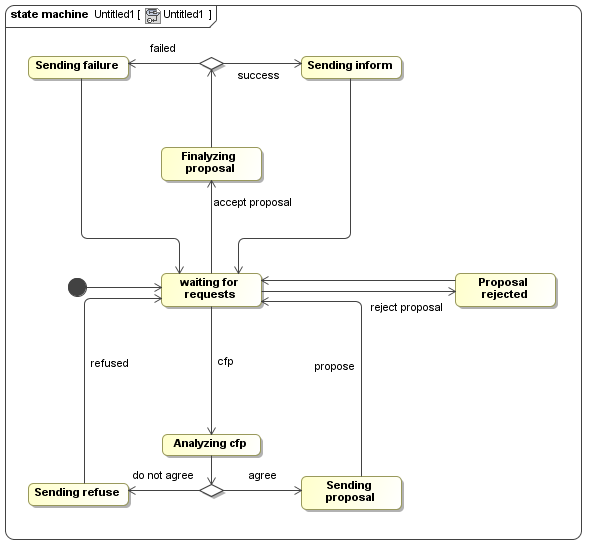


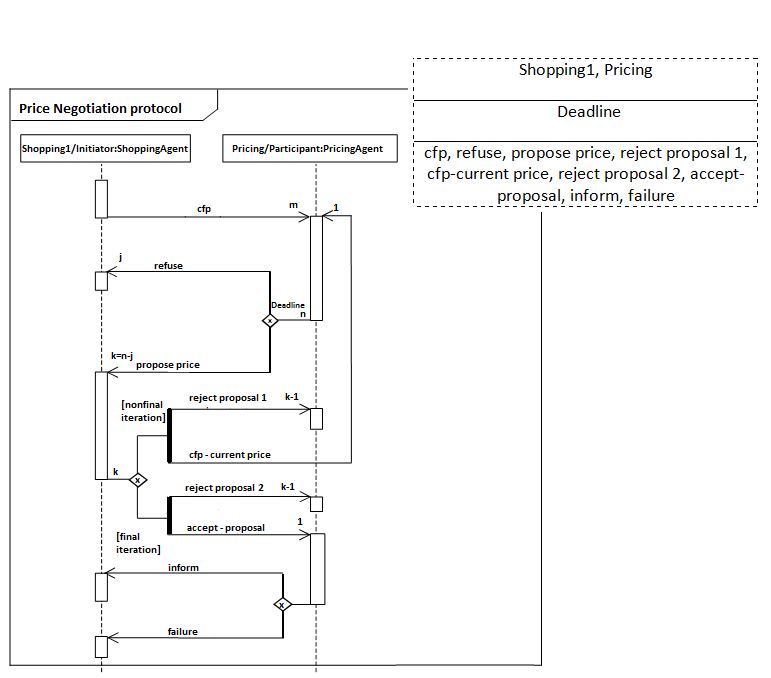
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Agent** | **Service** | **Inputs** | **Outputs** | **Pre-condition** | | **Post-condition** |
| Customer Agent | Place an order | customerRequirements | result | customerRequirements != null | | true |
| Shoppingmanager Agent | Accept 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 | |



## Task 2. Model interactions among agents in UML

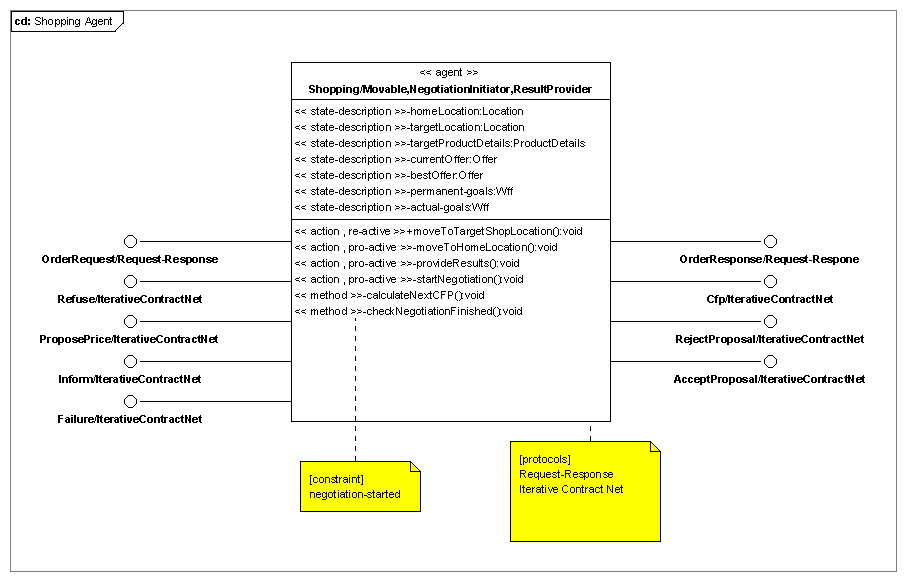


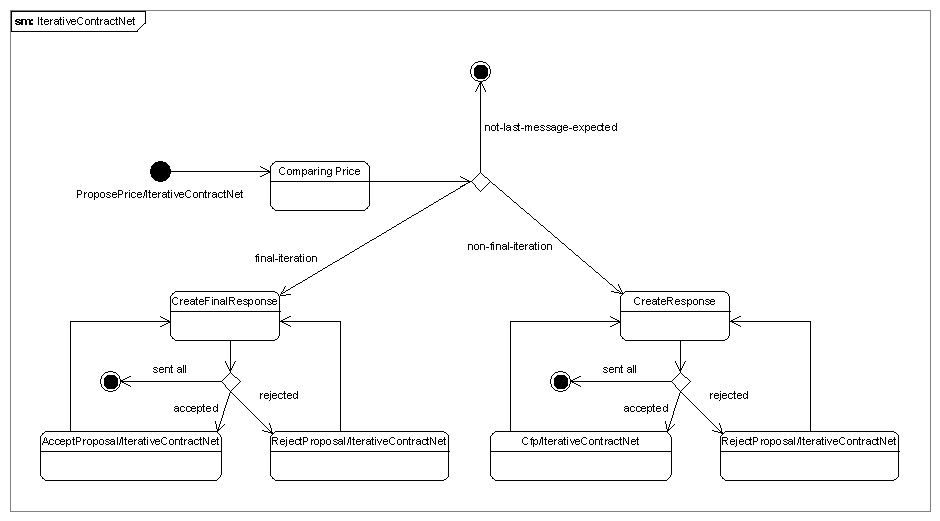


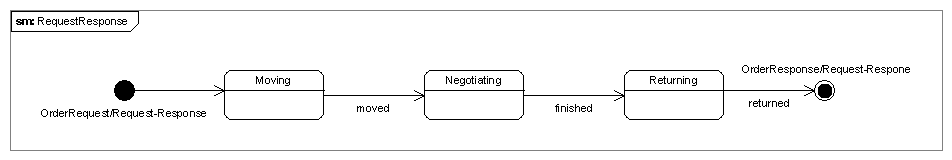
****

|  |
| --- |
| Shopping1, Pricing |
| Deadline |
| cfp, refuse, propose price, reject proposal 1, cfp-current price, reject proposal 2, accept-proposal, inform, failure |

## Task 3. Use UML Class diagrams to design behavior of your agents.

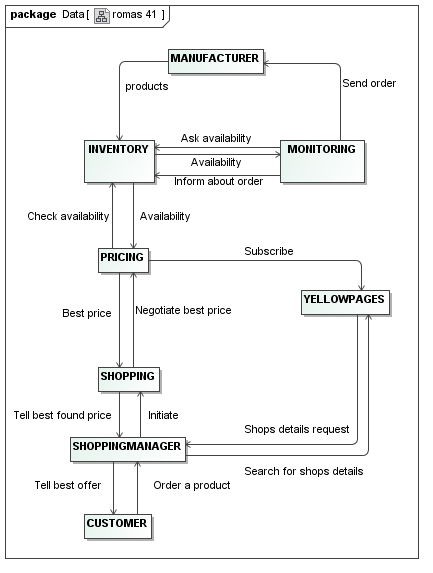


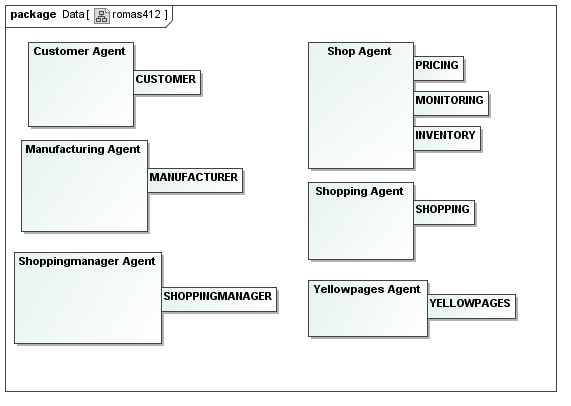


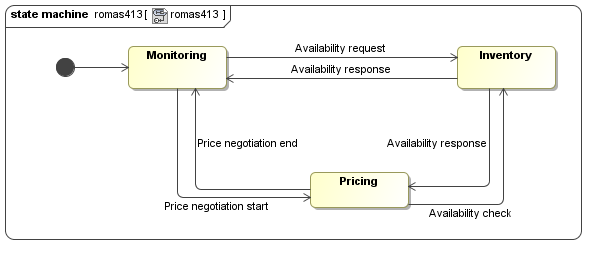


## Task 4. Model your system using Role based modeling approach

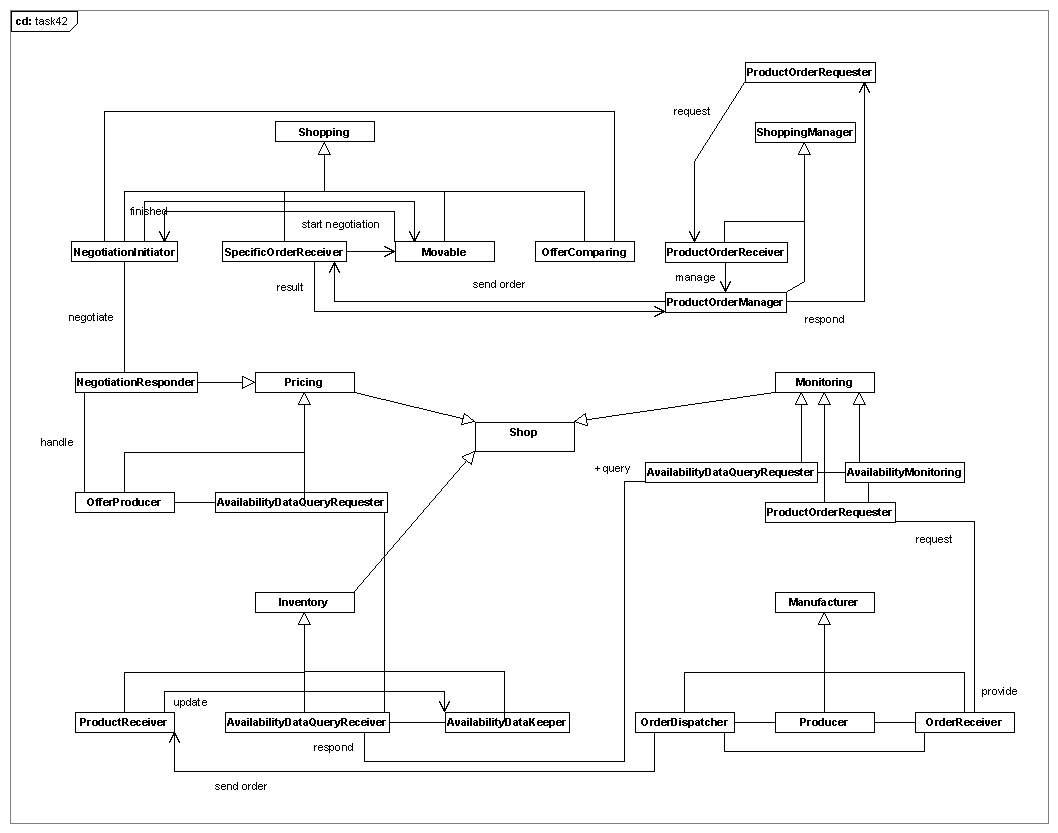
### Task 4.1 Augment the Analysis phase of GAIA with role binding







### 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