# Task1

## Virtual Tour

### 1. Requirements Statement

The Virtual Tour is a scenario that involves three different roles:

* the Profiler Agent, which represents a user, with personal information and interests
* the Tour Guide Agent, which represents a guide in a museum
* the Curator Agent, which represents the museum

In this scenario, we suppose that a user is interested in making a virtual tour, that is, viewing some artifacts, according to his own interests. To obtain the virtual tour, a Profiler Agent, which represents the user, will interact with two other agents, a Tour Guide Agent and a Curator Agent, sending and receiving messages.

The Tour Guide Agent is an intermediary between the Profiler and the Curator. Indeed, once it receives the request from the Profiler for getting a tour, it simply contacts the Curator, sending the same message that it received, concerning the interests of the user. Once it obtains the list of artifacts representing the tour, it sends it to the Profiler Agent.

The Curator Agent represents the museum, so it has a complete knowledge of the available artifacts. Once it receives a request for a virtual tour, it looks for the artifacts which best suit the interests of the user, and informs the Tour Guide about those items.

The last interaction is between the Profiler Agent and the Curator Agent. Since a virtual tour is given by a list of items’ names, the Profile must contact the Curator to get more details about the artifacts. Also in this case the interaction consists in sending and receiving messages.

### 2. Roles Model

|  |
| --- |
| **Role Schema**: PROFILER AGENT |
| **Description**:  Responsible for representing a user, generating with his/her interests and characteristics.  Responsible for looking for a Tour Guide Agent and a Curator Agent; sending requests to get a virtual tour from a Tour Guide Agent; receiving and handling the answer and sending another request to get more detailed information, about the artifacts of the virtual tour received, from a Curator Agent. |
| **Protocols and Activities**:  GenerateUser, LookForATourGuide, LookForACurator, RequestArtifactsIDs, GetArtifactsIDs, GetArtifactsDetails |
| **Permissions**:  **generates** UserDetails  **reads** **supplied** VirtualTour  **reads** **supplied** ArtifactsDetails |
| **Responsibilities**:  **Liveness**:  PROFILER AGENT = GenerateUser.  (LookForATourGuide.LookForACurator.Ids.GetArtifactsDetails)ω  IDS = RequestArtifactsIDs.GetArtifactsIDs  **Safety**: |

Schema for role PROFILER AGENT

Schema for role TOUR GUIDE AGENT

|  |
| --- |
| **Role Schema**: TOUR GUIDE AGENT |
| **Description**:  Responsible for looking for a Curator Agent.  Responsible for providing a virtual tour to a Profiler Agent, upon request. This is done requesting the tour to a Curator Agent. Once the answer is received, the Tour Guide Agent sends the tour to the Profiler Agent. |
| **Protocols and Activities**:  LookForACurator, RequestForTour, GetTour |
| **Permissions**:  **reads** **supplied** UserDetails  **reads** **supplied** VirtualTour |
| **Responsibilities**:  **Liveness**:  LookForACurator.(RequestForTour.GetTour)ω  **Safety**: |

Schema for role CURATOR AGENT

|  |
| --- |
| **Role Schema**: CURATOR AGENT |
| **Description**:  Responsible for generating a list of items, that represents the galleria of a museum; providing a virtual tour to a Tour Guide Agent, upon request, given the user’s interests. This is done comparing the user’s interests with the available artifacts. Once the list of matching items is constructed, it sends the virtual tour. It is also responsible for providing more information about the artifacts to a Profiler Agent, upon request. |
| **Protocols and Activities**:  GenerateGallery, ReceiveTourRequest, ReceiveDetailsRequest |
| **Permissions**:  **generates** Gallery  **generates** VirtualTour  **reads** **supplied** UserDetails  **reads** **supplied** VirtualTour |
| **Responsibilities**:  **Liveness**:  GenerateGallery.(ReceiveTourRequest.ReceiveDetailsRequest)ω  **Safety**: |

### 3. Interaction Model

|  |  |  |
| --- | --- | --- |
| RequestArtifactsIDs | | |
| Profiler Agent | Tour Guide Agent | Guide address  User information |
| Request for a virtual tour, with artifacts that match the interests of the user. | | Request message containing the user’s information |

|  |  |  |
| --- | --- | --- |
| RequestForTour | | |
| Tour Guide Agent | Profiler Agent | Request message containing the user’s information |
| Receiving a request for a tour from the Profiler.  Preparing a new message, to be sent to the Curator, containing the user’s information, to ask for a suitable tour. | | Request message containing the user’s information |

|  |  |  |
| --- | --- | --- |
| ReceiveTourRequest | | |
| Curator Agent | Tour Guide Agent | Request message containing the user’s information |
| Receiving the request for a tour from the Tour Guide.  Comparing the user’s interests with the available artifacts, to find the ones to put in the tour.  Preparing a new message containing the virtual tour to answer to the Tour Guide. | | Informative message containing a list of artifacts IDs |

|  |  |  |
| --- | --- | --- |
| GetTour | | |
| Tour Guide Agent | Curator Agent | Informative message containing a list of artifacts IDs |
| Receiving a virtual tour from the Curator.  Preparing a new message, to be sent to the Profiler, containing the list of artifacts IDs. | | Informative message containing a list of artifacts IDs |

|  |  |  |
| --- | --- | --- |
| GetArtifactsIDs | | |
| Profiler Agent | Tour Guide Agent | Informative message containing list of artifacts IDs |
| Receiving the tour, that is given by a list of artifacts IDs.  Preparing a new message, to be sent to the Curator, containing the list of artifacts IDs, to ask for more details. | | Request message containing a list of artifacts IDs |

|  |  |  |
| --- | --- | --- |
| ReceiveDetailsRequest | | |
| Curator Agent | Profiler Agent | Request message containing a list of artifacts IDs |
| Receiving a list of artifacts IDs from the Profiler.  Comparing the IDs with the IDs of the available items to find the artifacts to be sent to the Profiler.  Preparing a new message, to be sent to the Profiler, containing the artifacts, with all their information. | | Informative message containing a list of artifacts |

|  |  |  |
| --- | --- | --- |
| GetArtifactsDetails | | |
| Profiler Agent | Curator Agent | Informative message containing a list of artifacts |
| Receiving the complete information about the artifacts of the tour. | | List of artifacts of the virtual tour |

### 4. Agent Model

Curator Agent

Tour Guide Agent

Profiler Agent

Museum

Guide

User

+

+

+

### 4. Service Model

Profiler Agent services:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Input** | **Output** | **Preconditions** | **Postconditions** |
| Send a request for a virtual tour | Guide address  User’s information | Request message containing the user information | Guide registered at the DF | Guide aware of the request |
| Get the virtual tour, i.e., the list of artifacts IDs, and send a new request for getting more details about the artifacts | Inform message containing a list of artifacts IDs | Request message containing a list of artifacts IDs | Guide responds with a message containing the list of artifacts IDs  Curator registered on the DF | Curator aware if the request |
| Get a list of artifacts | Inform message containing a list of artifacts | The list is stored | Curator responds with a message containing the list of artifacts | Profiler knows all the artifacts suggested |
| Generate user | // | User’s personal information and interests | // | Profiler represents a user |
| Look for a Tour Guide Agent and store the AID as the reference guide | AID of a Tour Guide Agent | AID of a guide | Guide registered on the DF | Profiler knows which agent to contact as a guide |
| Look for a Curator Agent and store the AID as the reference curator | AID of a Curator Agent | AID of a curator | Curator registered on the DF | Profiler knows which agent to contact as a curator |

Tour Guide Agent services:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Input** | **Output** | **Preconditions** | **Postconditions** |
| Send a request for a virtual tour | Request message containing the user’s information | Request message containing the user’s information | Curator registered on the DF  Profiler sends a message with the tour request | Curator aware of the request |
| Get a virtual tour | Inform message containing a list of artifacts IDs | Inform message containing a list of artifacts IDs | Curator responds to the tour request | Profiler informed about the virtual tour |
| Look for a Curator Agent and store the AID as the reference curator | AID of a Curator Agent | AID of a curator | Curator registered on the DF | Tour Guide knows which agent to contact as a curator |

Curator Agent services:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Input** | **Output** | **Preconditions** | **Postconditions** |
| Receive a tour request  Find the matching artifacts | Request message containing the user’s information | Inform message containing a list of artifacts IDs | Guide sends a request for a tour | Guide is informed of the list of artifacts IDs |
| Receive a request for more details about a list of artifacts  Find the correct artifacts among the available ones | Request message containing a list of artifacts IDs | Inform message containing a list of artifacts | Profiler sends a request for more details about artifacts | Profiler is informed of the list artifacts |
| Generate a gallery, that is a collection of artifacts representing the museum | // | List of artifacts | // | Curator represents a museum |

### 5. Acquaintance Model



## Dutch Auction

### 1. Requirements Statement

In the Dutch Auction scenario there are two types of agents that interact:

* the Artist Manager Agent, which plays the role of an auctioneer
* the Curator Agent, which plays the role of a bidder

In order to actually perform a Dutch auction, there should be one artist manager and at least two curators playing the roles of bidder.

The strategies of the participants are the following:

* the auctioneer sets a first price for the item and, at each round, lowers the proposed price, until either one bidder makes a good proposal and wins the item, or the reserve price is reached and the auction ends without selling the item;
* a bidder sets a starting bid and, at each round, increases his/her own bid, in order to win the auction before the other bidders but not paying too much.

Following these strategies, the participants will hopefully reach an agreement, that is the auctioneer will sell the item for a good price and a bidder will buy the item paying less than what the auctioneer asked at the beginning of the auction. But it may happen that an agreement is not reached: in this case the auction ends without the item being sold.

There are different exchanges of information between the auctioneer and the bidders. First, the participants must be informed of the start of the auction, and at each round they send and receive messages regarding the bids and the proposed price. When a bidder wants to actually buy the item, it will send a proposal, which will either accepted, if the bidder is the first one that tries to buy the item, or refused, if somebody else already got the item.

### 2. Roles Model

Schema for role Auctioneer

|  |
| --- |
| **Role Schema**: AUCTIONEER - **system** |
| **Description**:  Responsible for initiating an auction, processing proposals, modifying quote price, and terminating the auction either by accepting the best bid or by informing participants of no suitable bids when exceeding the reserve price. |
| **Protocols and Activities**:  GenerateFirstPrice, LookForBidders, InformStartAuction, SendQuotePrice, ReceiveProposals, InformNoBids, InformSuccessfulBid |
| **Permissions**:  **reads** **supplied** BidderProposal  **generates** QuotePrice |
| **Responsibilities**:  **Liveness**:  AUCTIONEER = AuctionProcess  AUCTIONPROCESS = GenerateFirstPrice.LookForBidders.InformStartAuction.  GenerateProposalResult  RUNPROPOSALCYCLE = (SendQuotePrice.ReceiveProposals)  GENERATEPROPOSALRESULT = (RunProposalCycle)+.InformSuccessfulBid |  (RunProposalCycle)+.InformNoBids  **Safety**:   * QuotePrice < ReservePrice 🡪 AuctionStatus = ended * ProposalPrice < QuotePrice 🡪 ProposalAcceptance = false |

Schema for role Bidder

|  |
| --- |
| **Role Schema**: BIDDER |
| **Description**:  Responsible for initiating an auction, processing proposals, modifying quote price, and terminating the auction either by accepting the best bid or by informing participants of no suitable bids when exceeding the reserve price. |
| **Protocols and Activities**:  GenerateStrategy, ReceiveActiveStatus, ReceiveEndStatus, ReceiveQuotePrice, MakeProposal, ReceiveAnswerProposal |
| **Permissions**:  **reads** **supplied** QuotePrice  **generates** ProposalPrice |
| **Responsibilities**:  **Liveness**:  GenerateStrategy.(ReceiveActiveStatus.ReceiveQuotePrice.|MakeProposal|.  ReceiveAnswerProposal) | ReceiveEndStatus  **Safety**:   * ProposalPrice < QuotePrice 🡪 SendNewProposal = false |

### 3. Interaction Model

|  |  |  |
| --- | --- | --- |
| InformStartAuction | | |
| Auctioneer | Bidder | Bidders’ addresses  Auction with status set to active |
| Inform the bidders participating to the auction that it is started | | Inform message containing the auction |

|  |  |  |
| --- | --- | --- |
| ReceiveActiveStatus | | |
| Bidder | Auctioneer | Inform message containing the auction |
| Receive a message containing the auction whose status is set to active | | Bidder is aware that the auction is active |

|  |  |  |
| --- | --- | --- |
| SendQuotePrice | | |
| Auctioneer | Bidder | QuotePrice |
| Inform the bidders participating to the auction about the new quote price | | CFP message containing the auction with the current quote price |

|  |  |  |
| --- | --- | --- |
| ReceiveQuotePrice | | |
| Bidder | Auctioneer | CFP message containing the auction with the current quote price |
| Receive an inform message containing the auction with the current quote price  If the quote price is less than the current bid,  then SendNewProposal = true | | SendNewProposal |

|  |  |  |
| --- | --- | --- |
| MakeProposal | | |
| Bidder | Auctioneer | SendNewProposal = true |
| Prepare a propose message to make a proposal to the auctioneer | | Propose message containing the auction with the current proposal of the bidder |

|  |  |  |
| --- | --- | --- |
| ReceiveProposal | | |
| Auctioneer | Biddder | Propose message containing the auction with the current proposal of the bidder |
| Receive the proposal from the bidder and analyse the bid. Then prepare an answer, that is either accepting the proposal (ProposalAcceptance = true) or refusing it because the auction already ended (ProposalAcceptance = false) | | ProposalAcceptance |

|  |  |  |
| --- | --- | --- |
| InformSuccessfulBid | | |
| Auctioneer | Biddder | ProposalAcceptance |
| Send an inform message about the end of the auction because a bidder has won | | Inform message concerning the acceptance/rejection of the proposal |

|  |  |  |
| --- | --- | --- |
| ReceiveAnswerProposal | | |
| Bidder | Auctioneer | Inform message concerning the acceptance/rejection of the proposal |
| Receive an inform message concerning the acceptance/rejection of the proposal | | The bidder is aware of winning/losing the auction |

|  |  |  |
| --- | --- | --- |
| ReceiveEndStatus | | |
| Bidder | Auctioneer | Inform message concerning the ending of the auction |
| Receive an inform message about the ending of the auction | | The bidder is aware of the ending of the auction |

|  |  |  |
| --- | --- | --- |
| InformNoBid | | |
| Auctioneer | Biddder | AuctionStatus = ended |
| Send an inform message to the participants to inform that the auction ended because the reserve price has been reached | | Inform message concerning the ending of the auction |

### 4. Agent Model

Bidder

Auctioneer

2… n

1

Buyer

Sellery

### 4. Service Model

Auctioneer services:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Input** | **Output** | **Preconditions** | **Postconditions** |
| Inform the bidders participating to the auction that it is started | Bidders’ addresses  Auction | Inform message containing the auction | Bidders registered at the DF | Bidders aware of the start of the auction |
| Inform the bidders participating to the auction about the new quote price | Bidders’ addresses  QuotePrice  Auction | CFP message containing the auction | Bidders registered at the DF  Auction is active  QuotePrice > ReservePrice | Bidders aware of the new QuotePrice |
| Receive the proposal from the bidder and analyze the bid. Then prepare an answer. | Propose message  Bid  Auction | ProposalAcceptance | QuotePrice = Proposal | Bidder either wins or loses |
| Inform the participants the auction ended | Bidders’addresses  Auction | Inform message containing the auction | StatusAuction = Ended and QuotePrice <= ReservePrice | Bidders aware of the end of the auction |
| Inform message about the end of the auction because a bidder has won | Bidder’s address  Auction | Inform message containing the auction | StatusAuction = FinishedItemSold  The bidder made a proposal | Bidder aware of having either won (accepted proposal) or lost (rejected proposal) the auction |
| Generate the auction and the first quote price | // | Auction | // | Auctioneer can start the auction |
| Look for Bidders and store the AIDs | AID of all the Bidders registered | AID of some Bidders | Bidders registered on the DF | Auctioneer knows which agents are participating to the auction |

Bidder services:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Input** | **Output** | **Preconditions** | **Postconditions** |
| Receive a message containing the auction whose status is set to active | Inform message containing the auction | Auction | Bidder registered on the DF | Bidder is aware that the auction is active |
| Receive an inform message about the ending of the auction | Inform message concerning the ending of the auction | AuctionStatus = Ended | Bidder was participating to the auction | The bidder is aware of the ending of the auction |
| Receive an inform message containing the auction with the current quote price | CFP message containing the auction with the current quote price | SendNewProposal = true if QuotePrice < Bid | Bidder is participating to the auction |  |
| Prepare and send a propose message to make a proposal to the auctioneer | SendNewProposal  QuotePrice | Propose message containing the auction with the current proposal of the bidder | SendNewProposal = true | Auctioneer is aware of the proposal of the Bidder |
| Receive an inform message concerning the acceptance/rejection of the proposal | Inform message concerning the acceptance/rejection of the proposal | AuctionStatus = FinishedItemSold | Auction ended because a bidder won | The bidder is aware of winning/losing the auction |
| Generate the strategy of the bidder | Name of the Bidder | Incremental factor | The name of the Bidder contains an integer number | Bidder participates to the auction with its own strategy |

### 5. Acquaintance Model



## Intra-Platform Auctions

### 1. Requirements Statement

The Intra-Platform Auctions scenario extends the Dutch Auction one for Intra-Platform mobility. An auctioneer will handle two different auctions in two containers; the final goal is to determine which auction ended with the best result, i.e., in which container the item was sold for a greater price.

To do that, the auctioneer must be able to create two clones, that will handle the two separate auctions, regarding the same artifact. In each container there must be multiple bidders: one original agent creates several clones that will represent the other participants to the auction. We can notice that on one hand the auctioneer creates the clones and moves them in different containers; while on the other hand the bidders create clones in the same container in which they are. At the end of the auction, the auctioneer’s clones move back to the main container to communicate the final prices so that the original agent can decide which auction ended in a better way.

The roles, interactions, services and protocols between the auctioneer and the bidders are the same as the ones presented in the Dutch Auction scenario. The additional part is represented by the mobility model.

### 2. The Mobility Model

#### Step 1: Places Types

Place types are locations that the mobile agent can visit or reside in; they define the working environment of mobile agents.

|  |  |  |
| --- | --- | --- |
| **Place Type** | **Description** | **Instances** |
| Main Container | A container where the original auctioneer is created and waits for messages from the cloned auctioneers | 1 mobile agent, the auctioneer |
| Heritage Malta Container | A container where bidders can reside and get cloned  A clone of the auctioneer will be moved here to start an auction | 4 mobile agents: a clone of the auctioneer, an original bidder which creates 2 clones in the same place |
| Museum Galileo Container | A container where bidders can reside and get cloned  A clone of the auctioneer will be moved here to start an auction | 4 mobile agents: a clone of the auctioneer, an original bidder which creates 2 clones in the same place |

#### Step 2: Agents and Places Specifications

We identify the relationship between agent types and place types.

|  |  |  |  |
| --- | --- | --- | --- |
| **Agent Type** | **Mobile** | **Place Type** | **Constraints** |
| Bidder | Yes | Heritage Malta Container  Museum Galileo Container |  |
| Auctioneer | Yes | Heritage Malta Container  Museum Galileo Container |  |

#### Step 3: Cardinality of Agents and Places

The cardinality between agent types and place types shows how many agents of an agent type can reside in a place of a place type.



#### Step 4: Travel Schema of Mobile Agent Types

|  |  |
| --- | --- |
| **Agent Type** | Bidder |
| **Description** | This agent will be cloned to participate in the auction |
| **Origin** | Heritage Malta Container  Or  Museum Galileo Container |
| **Final Destination** | Same as its origin |
| **List of atomic movements** | |  |  | | --- | --- | | Heritage Malta Container | Cloned in this container | | Museum Galileo Container | Cloned in this container | |
| **Paths** | Cloned in the same container, no paths |
| **Agent Type** | Auctioneer |
| **Description** | This agent will be created and cloned twice to set up two auctions and, in the end, deciding which was the best one |
| **Origin** | Main Container |
| **Final Destination** | Heritage Malta Container  Or  Museum Galileo Container |
| **List of atomic movements** | |  |  | | --- | --- | | Main Container | Cloned and moved in Heritage Malta Container | | Main Container | Cloned and moved in Museum Galileo Container | |
| **Paths** | From Main Container to Heritage Malta Container to Main Container  Or  From Main Container to Museum Galileo Container to Main Container |