Distributed Artificial Intelligence and Intelligent Agents (ID2209): Project assignment

Kim Hammar, Stockholm 16446

kimham@kth.se

I. Introduction

The requirements statement is essentially just a set of articulated requirements for the system/organization to be designed, for structural reasons the requirements are divided into various related models that use different levels of detail. The system in this context is a SmartMuseum Agent Framework, as of following the GAIA methodology [1] I will from here on frequently use the *organization* metaphor when referring to the system.

II. Task 1 - Modeling with GAIA Methdology

I. Analysis

I.1 Requirements Statement

I.1.1 Mission Statement

The SmartMuseum organization has the purpose of connecting different people and entities that are in some sense involved in consuming or providing services related to art. The goal of the organization is to improve the overall experience for everyone involved. The organization should make it easier for consumers to view and find interesting art, for art-curators to provide art and reach out to consumers, for tourguides to find interested consumers as well as building relevant tours and finally for artists to sell their work.

I.1.2 Organization Description

The activity of a consumer viewing an art-artifact involves atleast three, sometimes four, or five main divisions: *tour-guide division*, *art-curator division*, *artist-management division*, *user-service division* and *artist-division*. The activity is initiated by the consumer who contacts the user-service division and selects some type of art-service, the user-service divison support the consumer in requesting/retrieving the service from either the art-curator division or tour-guide-division. In parellel to managing consumer requests the tour-guide division browses art-artifacts that is curated by the art-curator division. Further more, the art-curator divison participates in auctions for obtaining art-artifacts from the artist-management division, in parallel to managing requests from consumers and tourguides. Finally, the artist-management division initiates auctions for art-artifacts on request from artists.

The activities described above can the be modelled as an organization in the following way. The organization consists of 7 roles. The ArtConsumer (AC) who consumes arts in different forms. The UserHandler (UH) which the consumer uses to purchase and browse services related

to art. The TourGuide (TA) which builds and offers virtual tours. The ArtBuyer (AB) who buys art to include in its gallery/museum, the ArtQuoter (AQ) who quotes the price for arts and sells it to consumers. The ArtSeller (AS) who is hired by artists to sell their work to art buyers. And finally the Artist (A) who produces art.

I.2 Roles Model

The following assumption is necessary to avoid making decisions about implementation details when doing the analysis/design.

Assumption 1-$\mathcal{A}$. *Roles can find each other in some way in order to communicate*

Role Schema: ArtConsumer (AC)

Description:
Initiates activity of consuming art, either buying artifact or downloading a virtual tour

Protocols and activities:
DownloadVirtualTour, BuyArt, ConsumeService

Permissions:

reads	supplied *availableServices*	//	*list of services*
	money	//	*money of the consumer*
	userProfile	//	*profile of the consumer*
	visitedArtifacts	//	*list of visited artifacts*
generates	*valuation*	//	*valuation of selected artifact*
	artifactTitle	//	*title of selected artifact*
	virtualTourTitle	//	*title of selected virtual-tour*
	moneyForArtifact	//	*money for selected artifact*
	supplied *virtualTour*	//	*downloaded virtual-tour*
	supplied *auctionResult*	//	*bought artifact or nil*

Responsibilities
Liveness:

$$\text{ArtConsumer} = (\text{GetService. ConsumeService})^{\omega}$$
$$\text{GetService} = (\text{DownloadVirtualTour} \mid \text{BuyArt})$$

Safety:

- $moneyForArtifact \leq money$
- $artifactTitle \in availableServices.artifacts$
- $virtualTourTitle \in availableServices.artifacts$

Figure 1: Schema for role ArtConsumer

Role Schema: UserHandler (UH)

Description:

Receives request to buy art-services from consumers and manages the process of the consumer purchasing and obtaining the service.

Protocols and activities:

GetArtifact, GetVirtualTour, GetArtifactsList, GetVirtualTourList, GenerateListOfArtServices

Permissions:

generates	*availableServices*	//	*list of services*
	strategy	//	*strategy for dutch auction*
reads	supplied *virtualTours*	//	*list of virtual tours*
	supplied *artifacts*	//	*list of art-artifacts*
	supplied *moneyForArtifact*	//	*consumer money to purchase artifact*
	supplied *valuation*	//	*consumer valuation of artifact*
	supplied *artifactTitle*	//	*title of artifact-purchase*
	supplied *virtualTourTitle*	//	*title of virtual-tour selection*
	supplied *virtualTour*	//	*virtual-tour downloaded by consumer*
	supplied *auctionResult*	//	*artifact bought by consumer or nil*

Responsibilities

Liveness:

$\text{UserHandler} = (All)^{\omega}$

$\text{All} = (\text{PresentServices} \mid\mid \text{HandleConsumerRequest})^{\omega}$

$\text{PresentServices} = \text{GetServices}.\ \text{GenerateListOfArtServices}$

$\text{GetServices} = \text{GetArtifactsList}.\ \text{GetVirtualToursList}$

$\text{HandleConsumerRequest} = \text{GetArtifact} \mid \text{GetVirtualTour}$

Safety:

- $availableServices = artifacts \cup virtualTours$
- $auctionResult \neq nil \implies auctionResult \in artifacts$
- $virtualTour \in virtualTours$

Figure 2: Schema for role UserHandler

Role Schema: TourGuide (TG)

Description:

Responsible for constructing virtual tours of art-artifacts. Looks up available artifacts at curators and then builds different types of tours.
Sends tours to user-handlers.

Protocols and activities:

SendVirtualTours, SendVirtualTour, GetArtifactList, BuildVirtualTour

Permissions:

generates	*virtualTour*	//	*virtual tour of art-artifacts*
	virtualTours	//	*list of virtual-tours*
reads	supplied *artifacts*	//	*list of artifacts*
	supplied *virtualTourTitle*	//	*specific virtual-tour title*

Responsibilities

Liveness:

$\text{TourGuideBuilder} = (\text{ConstructTour} \parallel [\text{Send}])^{\omega}$
$\text{ConstructTour} = (\text{GetArtifactList}.\ \underline{\text{BuildVirtualTour}})^{\omega}$
$\text{Send} = \text{SendVirtualTours} \mid \text{SendVirtualTour}$

Safety:

- $\forall virtualTour.artifact \quad virtualTour.artifact \in artifacts$

Figure 3: Schema for role TourGuide

Role Schema: ArtBuyer (AB)

Description:
Buys art-artifacts from art-sellers.

Protocols and activities:
BuyArt, SendArtifacts

Permissions:

generates	$artifacts$	//	*list of purchased artifacts*
	$strategy$	//	*strategy for dutch auction*
	$valuation$	//	*valuation for artifact*
	$moneyForArtifact$	//	*money for artifact*
reads	$money$	//	*the buyer's money*
	$artifactTitle$	//	*title for a specific artifact*
	supplied $artifactResult$	//	*bought artifact or nil*

Responsibilities
Liveness:

$$\text{ArtBuyer} = (\text{[BuyArt]} \mid\mid \text{[SendArtifacts]})^{\omega}$$

Safety:

- $moneyForArtifact \leq money$
- $artifactTitle \in artifacts$

Figure 4: Schema for role ArtBuyer

Role Schema: ArtQuoter (AQ)

Description:
Quotes art and resells it to consumers

Protocols and activities:
QuoteArt, SellArt, GetArtifacts, SendArtifacts

Permissions:

reads	supplied *artifacts*	//	*list of artifacts*
	supplied *artifact*	//	*artifact for auction*
generates	*quote*	//	*quote of artifact*
	rateOfReduction	//	*rate of reduction for dutch auction*
	initialPrice	//	*initial price for auction*
	reservePrice	//	*reserved price for auction*
	price	//	*price auction ended at*
	winner	//	*winner of auction or nil*
	artifactResult	//	*result of auction*
	bidders	//	*bidders of auction*

Responsibilities

Liveness:

$$\text{ArtQuoter} = ((\text{GetArtifacts}.\ \underline{\text{QuoteArt}}.\ \text{SellArt}) \parallel \text{SendArtifacts})^{\omega}$$

Safety:

- $winner \in bidders$
- $reservePrice \leq price \leq initialPrice$

Figure 5: Schema for role ArtQuoter

Role Schema: ArtSeller (AS)

Description:
Sells art to art-traders/curators.

Protocols and activities:
SellArt, GetArtifact

Permissions:

reads	supplied *artifact*	//	*artifact to be sold*
generates	*rateOfReduction*	//	*rate of reduction for dutch auction*
	initialPrice	//	*initial price for auction*
	reservePrice	//	*reserved price for auction*
	price	//	*price auction ended at*
	winner	//	*winner of auction or nil*
	artifactResult	//	*result of auction*
	bidders	//	*bidders of auction*

Responsibilities
Liveness:

$$\textsc{ArtSeller} = (\mathsf{GetArtifact}.\ \mathsf{SellArt})^{\omega}$$

Safety:

- $winner \in bidders$
- $reservePrice \leq price \leq initialPrice$

Figure 6: Schema for role ArtSeller

Role Schema: ARTIST (A)

Description:
Sells art to art-traders/curators.

Protocols and activities:
ProduceArt, SendArtifact

Permissions:
generates *artifact* // *produced artifact*

Responsibilities
Liveness:
$\text{ARTIST} = (\underline{\text{ProduceArt}}.\ \text{SendArtifact})^{\omega}$
Safety:

- **true**

Figure 7: Schema for role ARTIST

I.3 Interaction Model

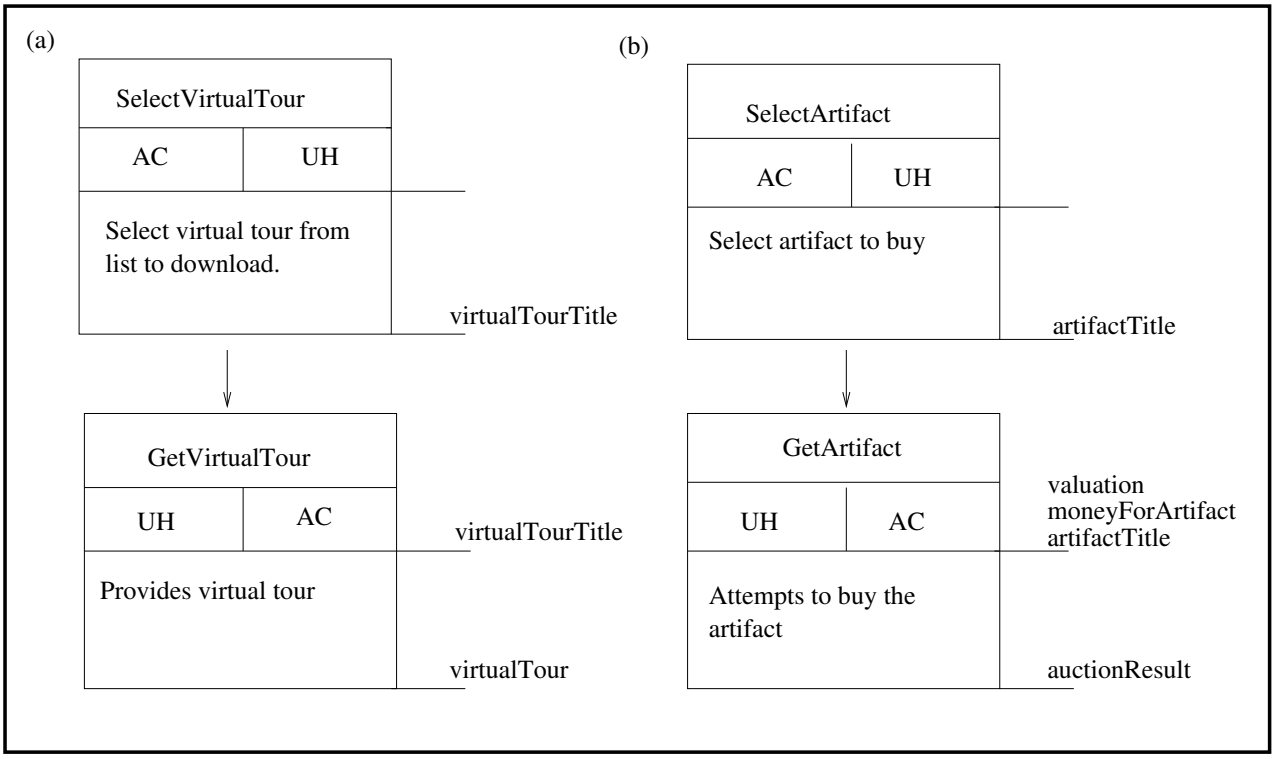

Figure 8: Definition of protocols associated with the ArtConsumer role: (a) DownloadVirtualTour, (b) BuyArt

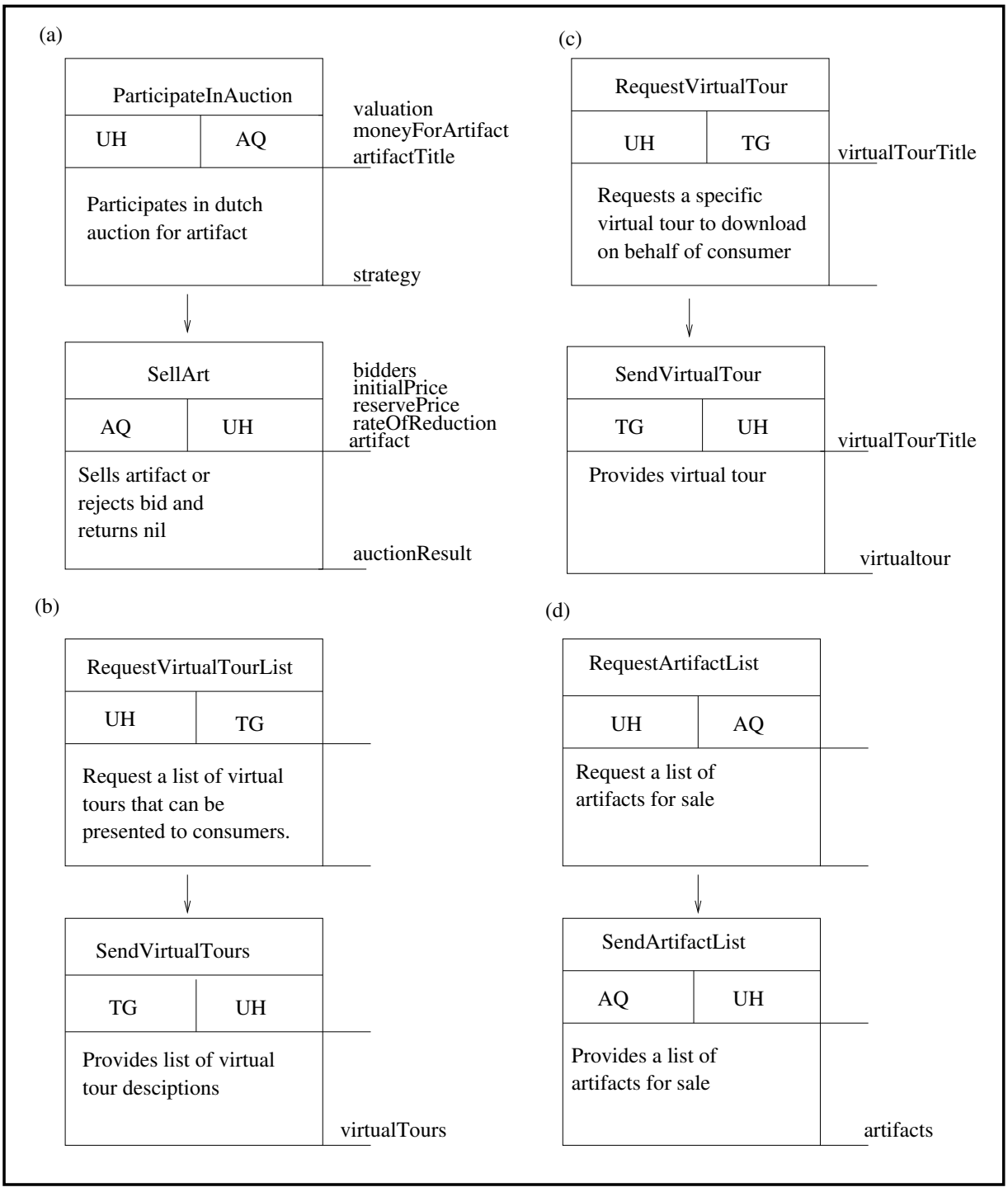

Figure 9: Definition of protocols associated with the UserHandler role: (a) GetArtifact, (b) GetVirtualTourList, (c) GetVirtualTour, (d) GetArtifactsList

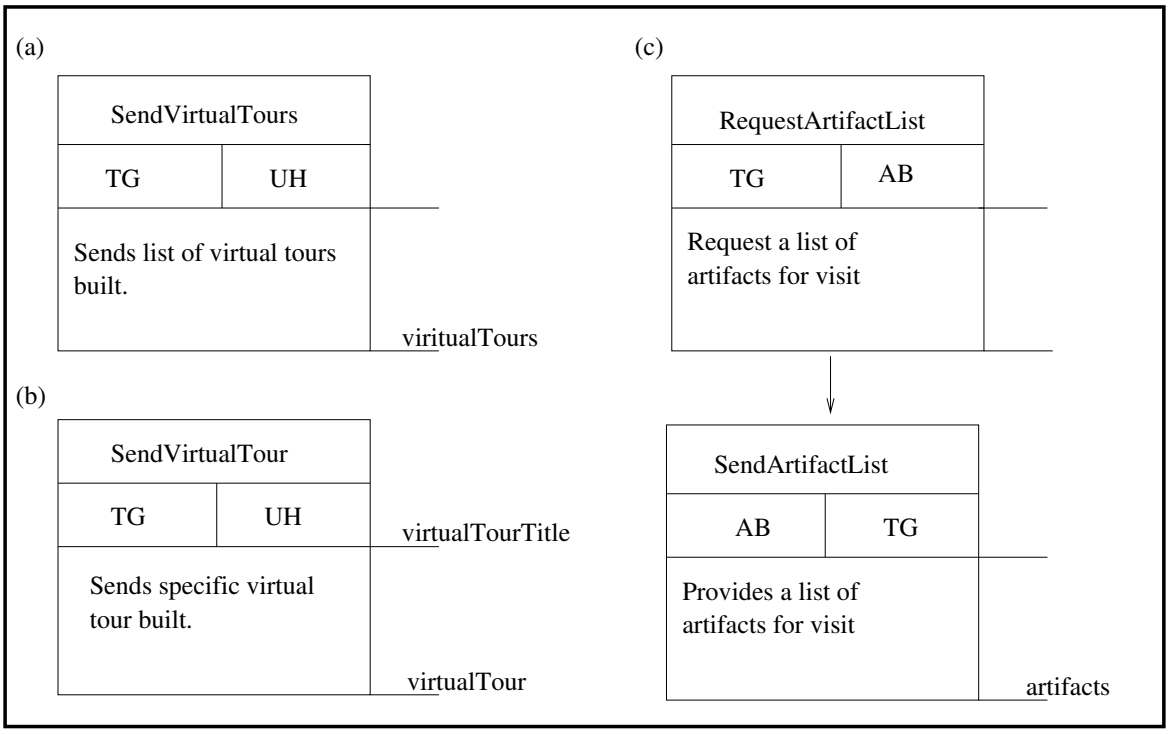

Figure 10: Definition of protocols associated with the TourGuide role: (a) SendVirtualTours, (b) SendVirtualTour, (c) GetArtifactList

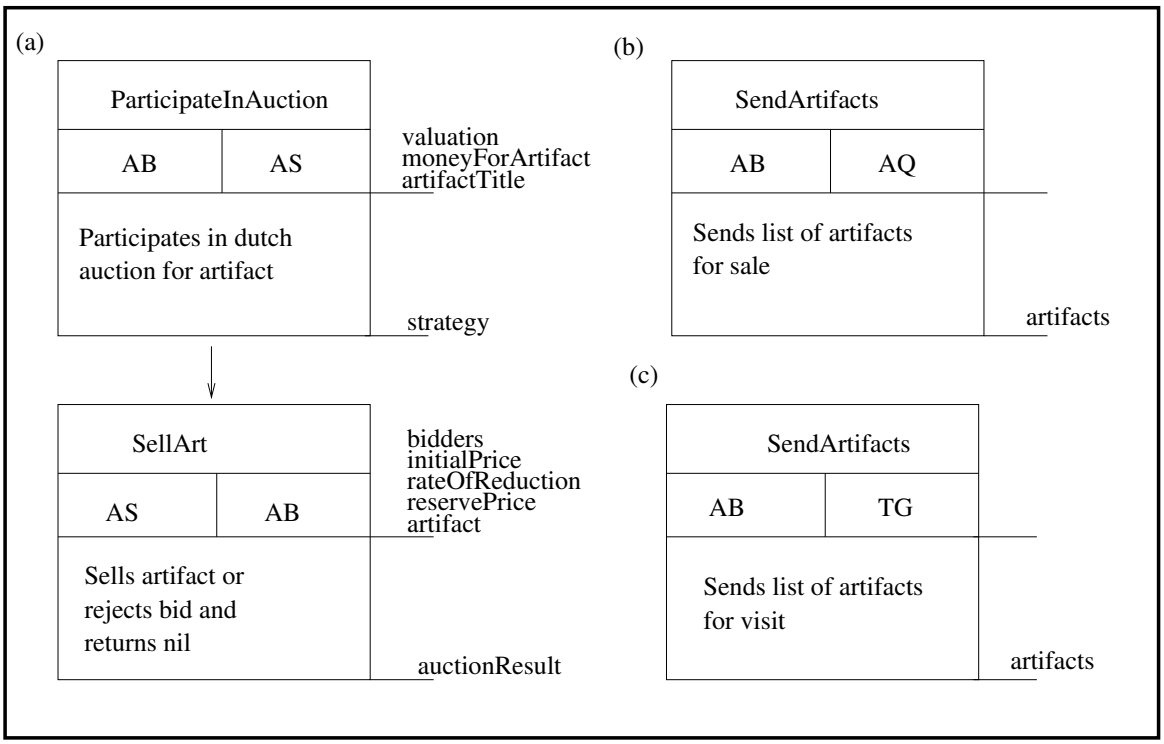

Figure 11: Definition of protocols associated with the ArtBuyer role: (a) BuyArt, (b) SendArtifacts (1), (c) SendArtifacts (2)

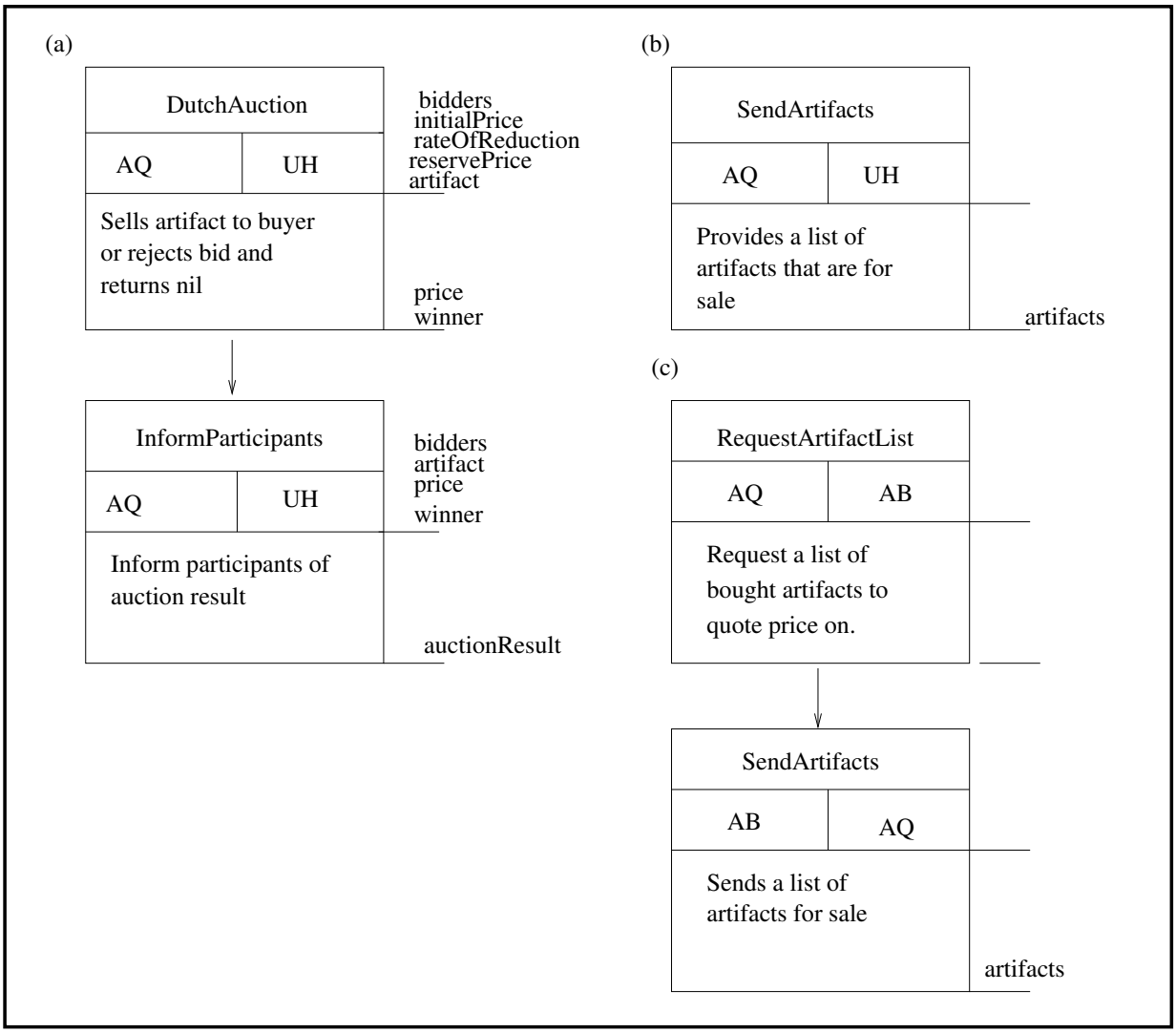

Figure 12: Definition of protocols associated with the ArtQuoter role: (a) SellArt, (b) SendArtifacts, (c) GetArtifacts

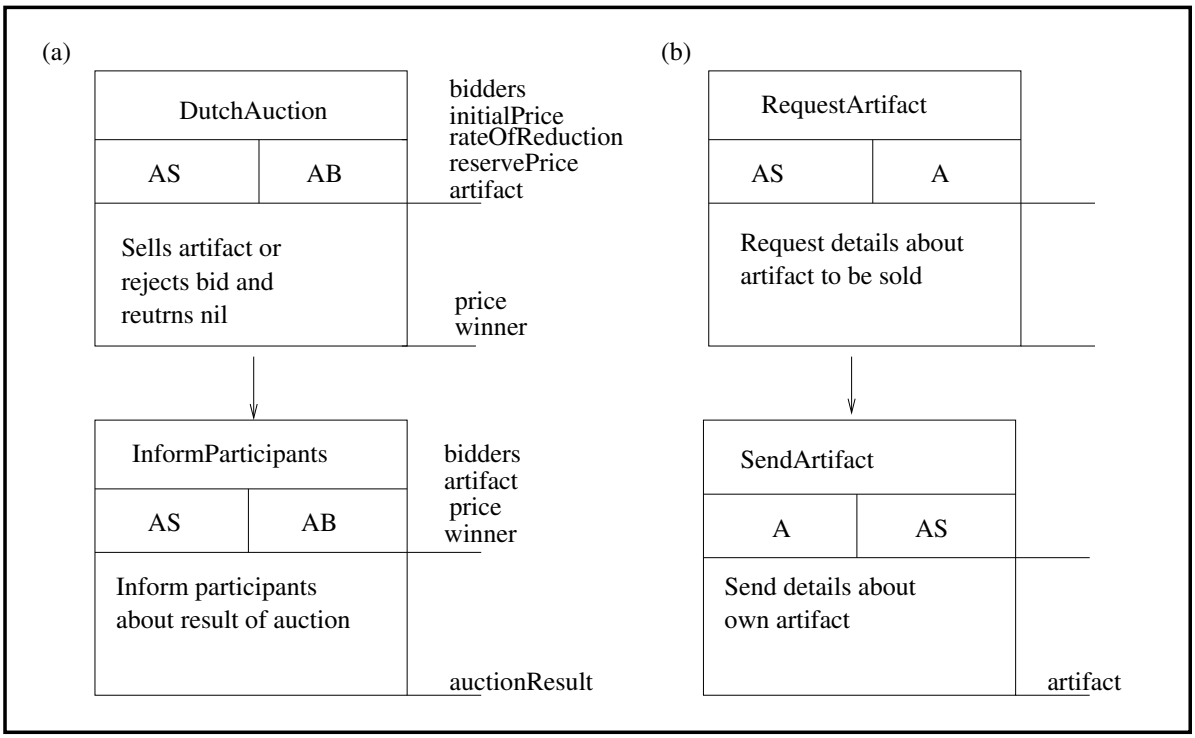

Figure 13: Definition of protocols associated with the ArtSeller role: (a) SellArt, (b) GetArtifact

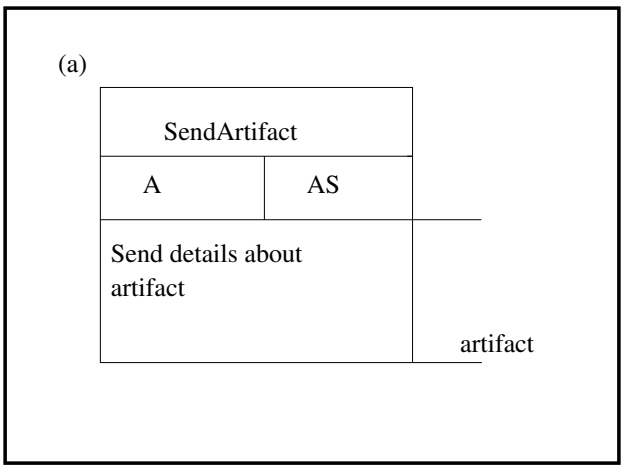

Figure 14: Definition of protocols associated with the Artist role: (a) SendArtifact

II. Design

II.1 Agent Model

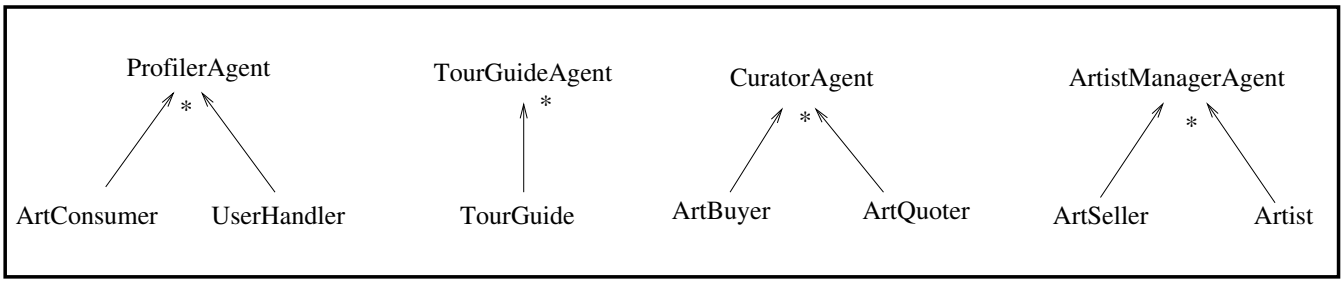

Figure 15: The agent model

II.2 Services Model

Table 1: Services model for agent ProfilerAgent

Service	Inputs	Outputs	Pre-condition	Post-condition
obtain virtual-tour list		$virtualTours$	**true**	$virtualTours \neq nil$
obtain artifact list		$artifacts$	**true**	$artifacts \neq nil$
generate list of services	$virtualTours, artifacts$	$availableServices$	$\exists virtualTours, artifacts$	created list of available services
register as bidder for auction	$auctioneer, artifact$		auction exists	$self \in auctioneer.bidders \wedge strategy \neq nil$
receive CFP	currentPrice		is participating in the auction	**true**
place bid	currentPrice		$currentPrice \leq moneyForArtifact$	bid sent to auctioneer
receive bid result	$accept \vee reject$		have bidded	bid accepted or rejected
informed auction ended	$artifact \vee nil$		participated in auction	informed auction ended and received result
download virtual tour	$tourguide, virtualtour$		$\exists tourguide, virtualtour$	downloaded virtual tour
visitArtifact	$curator, artifactTitle$		$artifactTitle \in curator.gallery.titles$	$artifactTitle \in visitedArtifacts$

Table 2: Services model for agent TourGuideAgent

Service	Inputs	Outputs	Pre-condition	Post-condition
obtain artifact list		$artifacts$	**true**	$artifacts \neq nil$
manage virtual-tour request	$virtualTourTitle$	$virtualTour \vee nil$	**true**	**true**
manage list of virtual-tours-request		$virtualTours$	**true**	**true**
build virtual tour	$artifacts$	$virtualTour$	$artifacts.size > 0$	$virtualTour \neq nil$

Table 3: Services model for agent ArtistManagerAgent

Service	Inputs	Outputs	Pre-condition	Post-condition
get registered bidders		$bidders$	**true**	**true**
send inform-start-of-auction	$bidders$	$informMessage$	bidders are registered	bidders informed about start of auction
send CFP	$bidders$	CFP	bidders are registered and auction ongoing	bidders informed about current price and encouraged to bid
receive bid	bid	$bids$	bidder registered	$bid \in bids$
manage bids	$bids$	$bidResponses$	$bids > 0$	one bid was accepted and the bidder received the good, the rest was rejected and the bidders were informed
modify price	$reservePrice$, $rateOfReduction$, $currentPrice$	$newPrice$	no bids was received	$reservePrice \leq newPrice \leq currentPrice$
send inform-auction-closed	$bidders$, $auctionResult$	$informMessage$	bidders are registered	bidders informed about close of auction

Table 4: Services model for agent CuratorAgent

Service	Inputs	Outputs	Pre-condition	Post-condition
get registered bidders		*bidders*	**true**	**true**
register as bidder for auction	*auctioneer*, *artifact*		auction exists	$self \in auctioneer.bidders \land strategy \neq nil$
receive CFP	currentPrice		is participating in the auction	**true**
place bid	currentPrice		$currentPrice \leq moneyForArtifact$	bid sent to auctioneer
receive bid result	$accept \lor reject$		have bidded	bid accepted or rejected
informed auction ended	$artifact \lor nil$		participated in auction	curator were informed auction ended and received result
manage artifact-list request		*artifacts*	**true**	**true**
manage visit-artifact request	*artifactTitle*	*artifact*	**true**	provided artifact for visit only
quote art	*artifact*	*quote*	**true**	**true**
send inform-start-of-auction	*bidders*	*informMessage*	bidders are registered	bidders informed about start of auction
send CFP	*bidders*	*CFP*	bidders are registered and auction ongoing	bidders informed about current price and encouraged to bid
receive bid	*bid*	*bids*	bidder registered	$bid \in bids$
manage bids	*bids*	*bidResponses*	$bids > 0$	one bid was accepted and the bidder received the good, the rest was rejected and the bidders were informed
modify price	*reservePrice*, *rateOfReduction*, *currentPrice*	*newPrice*	no bids was received	$reservePrice \leq newPrice \leq currentPrice$
send inform-auction-closed	*bidders*, *auctionResult*	*informMessage*	bidders are registered	bidders informed about close of auction

II.3 Acquaintance Model

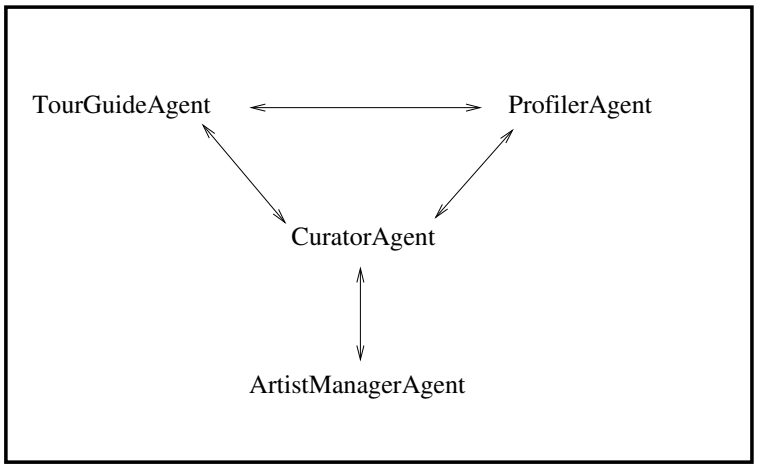

Figure 16: Acquaintance model

II.4 Mobility Model

Assumption 2-$\mathcal{A}$. *I've assumed the mobile architecture that I used for homework 3, i.e that only artist-manager agents and curator agents are mobile and can close themself. Further more the cardinality of agents and places also follow from this assumption.*

Table 5: Place Types

Place Types	Description	Instances
Heritage Malta Container	Container where art-curators can reside and perform their services and where artistman-ager agents can reside temporarily to perform auctions	1
Museo Galileo Container	Container where art-curators can reside and perform their services and where artistman-ager agents can reside temporarily to perform auctions	1
ArtistManager Container	Container where artist-manager agents reside and where they come back to after performing auctions	*
ProfilerAgent Container	Container where profiler agents reside	*
TourGuideAgent Con-tainer	Container where tour-guide agents reside	*

Table 6: Agents and Places Specification

Agent Type	Mobile	Place Type	Constraints
ProfilerAgent	No	ProfilerAgent Container	
TourGuideAgent	No	TourGuideAgent Con-tainer	
CuratorAgent	Yes	Museo Galileo Container, Heritage Malta Container	
ArtistManagerAgent	Yes	ArtistManagerAgentContainer, Museo Galileo Container, Heritage Malta Container	

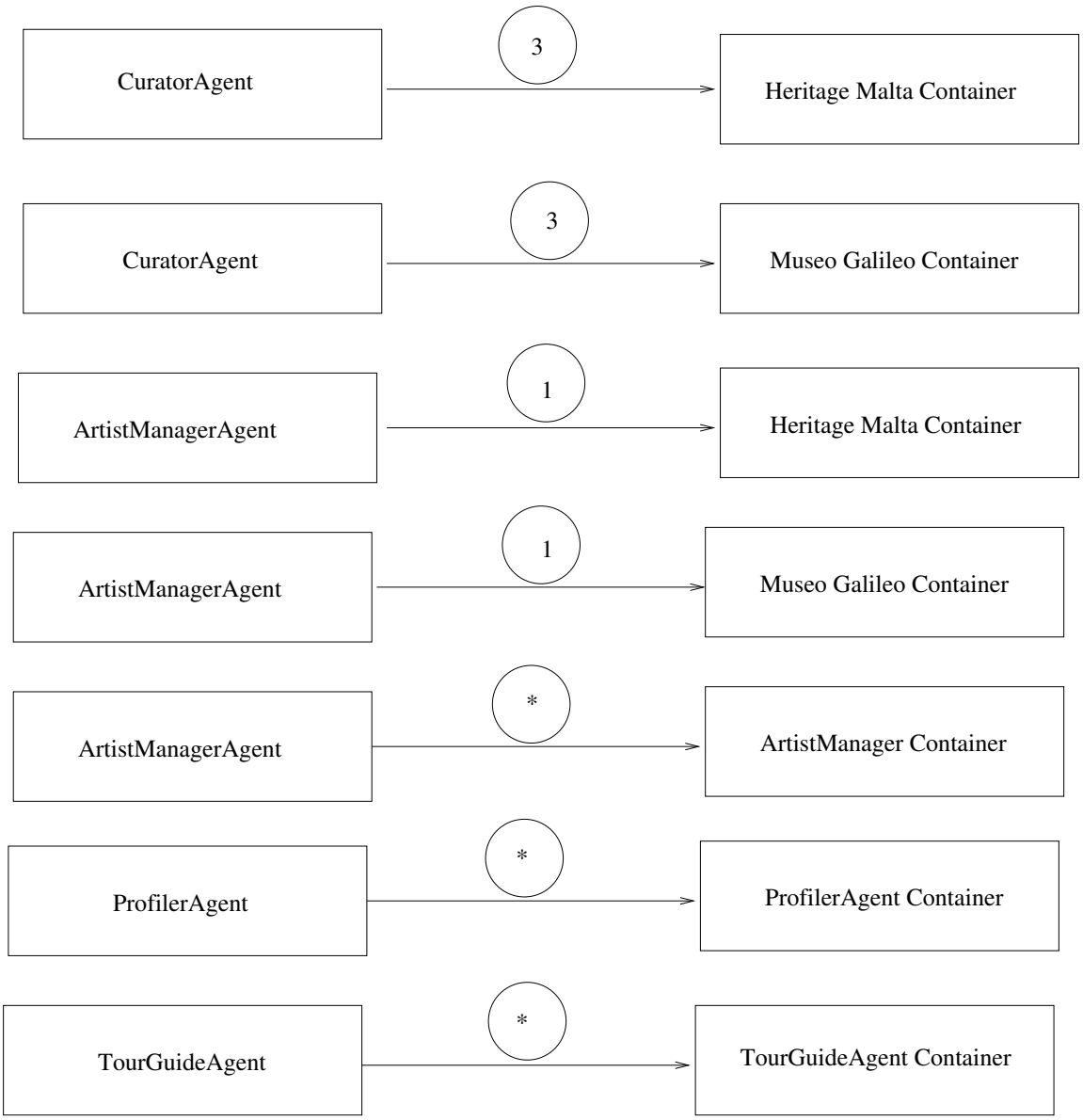

Figure 17: Cardinality of Agents and Places

Agent Type: CuratorAgent

Description: Can be cloned in current container to participate in auctions.
Origin: Heritage Malta Container or Museo Galileo Container.
Final Destination: Same as its origin container.
List of atomic movements:

1	Cloned in Heritage Malta Container
2	Cloned in Museo Galileo Container

Paths:

Cloned in the same container, no paths.

Figure 18: Travel schema for agentCuratorAgent

Agent Type: ArtistManagerAgent

Description: Can move between Museo Galileo Container, Heritage Malta Container and ArtistManager Container.
Origin: ArtistManager Container
Final Destination: ArtistManager Container
List of atomic movements:

1	Move from ArtistManager to Heritage Malta Container.
2	Move from Heritage Malta to ArtistManager Container.
3	Move from ArtistManager to Museo Galieo Container.
4	Move from Museo Galieo to ArtistManager Container.
5	Move from Museo Galieo to Heritage Malta Container.
6	Move from Heritage Malta to Museo Galieo Container.

Paths:

1	1.2
1	3.4
1	1.6
1	3.5
1	3.5.2
1	1.6.4

Figure 19: Travel schema for agentArtistManagerAgent

Agent Type: ProfilerAgent

Description:
Origin:
Final Destination:
List of atomic movements:

1	Description

Paths:

1	Description

Figure 20: Travel schema for agentProfilerAgent

III. Task 2 - Modeling with AgentUML

References

[1] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, September 2000.