

Hovering Information: a MAS implementation

Daniele Bellavista

November 26, 2013

1 Introduction

1.1 Vision

The hovering information is an information dissemination service working in an dynamic infrastructure-free environment with a self-organizing behaviour; a MAS[3, 5] approach may offer a sound paradigm for both hovering information implementation and simulation. The simulation implies the design of a *social* system, where people - hovering information users - move in an environment with different behaviours.

In section 2 the system is designed using the *SODA*[2] methodology. Section 3 describes the implementation techniques and strategies. Finally, section 4 shows how to configure the system and some screenshot.

1.2 Hovering Information System

Hovering Information is a geo-localized information dissemination service, proposed in [4], able to work without a centralized infrastructure. The service works with mobile devices capable of peer-to-peer communication and geo-localization. The hovering information system is composed by two main components: mobile nodes and pieces of hovering information.

Mobile nodes are components moving into the environment with a limited communication range, capable of communicate to peers, discover neighbors, access and store (inside a limited buffer) pieces of hovering information. A mobile node is assumed able to determinate its geographic position, speed and direction.

Pieces of hovering information are data that have to *survive* inside a circular area centered at a location called *anchor location* and having a radius called *anchor radius*. The survivability goal of a piece of hovering information is achieved moving or replicating the piece itself through the mobile

nodes. A piece of hovering information may have some policies controlling the movement between nodes.

In an hovering information system, three main requirements may be defined for each piece of hovering information [4]:

Survivability: a piece of hovering information is alive at some time t , if there is at least one node hosting a replica of this information.

$$survivability = \frac{alive_time}{total_time}$$

Availability: a piece of hovering information is available at some time t , if there is at least one node in its anchor area hosting a replica of this information.

$$availability = \frac{available_time}{total_time}$$

Accessibility: a piece of hovering information is accessible by a node at some time t , if the node is able to get this information; therefore, a replica exists in the node communication range.

$$accessibility = \frac{replica_covered_area}{anchor_area}$$

2 Hovering Information design

This project aims to implement the hovering information system working inside a social environment. Mobile nodes are owned by people, moving inside an environment composed by anchors, that is, locations where pieces of hovering information are present. In a real context, anchors are bound to points of interest, such as shops, information centers, and so on.

The system simulates an hovering information usage, inside an environment (although “*a simulator without the reproducibility ability is no more than little game*”[cit. Pianini], so from now on I may use the word “simulator” as a synonymous for “little game”).

People, carrying mobile nodes, have to walk with different behaviour, emulating movements inside an area composed by points of interest. The simulation should gather periodic data about hovering information status and properties (i.e. availability, survivability and accessibility), nodes position and communications link.

Note on the implemented system: the people move only using some sort of random walk, nothing more. And the simulation system only shows the actors, without computing the system properties.

2.1 Preliminary analysis

From requirements a three sub-system design emerges:

Simulator: graphic interface.

Hovering information: composed by pieces of hovering information and mobile nodes.

Social system: a group of hovering information users moving into the environment.

2.2 Requirements Analysis

From the requirements and the preliminary analysis considerations, the *SODA* process focuses on analysis and design of the *Hovering information system* and the *Social system*, treating them as a single system. The sub-system *Simulator* doesn't require independent control or intelligence, so the simulator interface can be assumed as a *Legacy System*.

2.2.1 Requirements Tables

Actor	Description
Simulation Manager	The simulation manager, which renders the world and analyze its data.
Person	User of a mobile node.
Mobile node	Hovering information low-level user: discover, access and create the infrastructure for the pieces of hovering information.
Piece of Hovering Information	Hovering information instance.

Table 2.2.1: Actor table $(C)Ac_t$

Requirement	Description
Access Hovering Information	Access the hovering information available in the current position.
Show Simulation	Show the simulation.
Move	Move within the social environment.
Survive	Stay alive.

Table 2.2.2: Requirement table $(C)Re_t$

Actor	Requirement
Person	Access Hovering Information, Move.
Simulation Manager	Show Simulation.
Mobile node	Access Hovering Information.
Piece of Hovering Information	Survive, Access Hovering Information.

Table 2.2.3: Actor-Requirement table $(C)AR_t$

2.2.2 Domain Tables

External Environment	Legacy system
External	Simulator UI.

Table 2.2.4: External Environment-Legacy System table $(C)EELS_t$

Legacy System	Description
Simulator UI	Simulation output interface that shows simulation data.

Table 2.2.5: Legacy System table $(C)LS_t$

2.2.3 Relations Tables

Relation	Description
Inquisition	The system should permit inquisition.
GUI	The system should send the information needed by the GUI.
Neighborhood	Two mobile nodes are able to communicate if they are neighbors.
HostToPiece Exchange	A piece of hovering information and its host are able to communicate.
Node Interface	A person needs a node interface in order to use it for accessing or creating information.

Table 2.2.6: Relation table $(C)Rel_t$

Requirement	Relation
Access Hovering Information	HostToPiece Exchange, Node Interface, Neighborhood.
Survive	HostToPiece Exchange, Neighborhood.
Be available	HostToPiece Exchange, Neighborhood.
Move	
Show Simulation	GUI.

Table 2.2.7: Requirement-Relation table $(C)RR_t$

Legacy-System	Relation
Simulator UI	GUI.

Table 2.2.8: Relation-LegacySystem table $(C)RLS_t$

2.3 Analysis

By analyzing the requirements, the basilar and necessary tasks are defined. Those are the tasks for landing and cloning hovering information, displaying information on the mobile device and get information hosted locally or remotely.

I decided to create functions for getting the position and discovering the neighbors, because those operation are available to each subject using the mobile node.

2.3.1 References Tables

Requirement	Task
Access Hovering Information	display_information, access_remote_information, access_local_information
Show Simulation	inquire_world, show_simulation
Survive	land_task, clone_task
Move	sense_task, move_task

Table 2.3.1: Reference Requirement-Task table $(C)RRT_t$

Requirement	Function
Access Hovering Information	get_position, discover_neighbors
Show Simulation	render

Table 2.3.2: Reference Requirement-Function table $(C)RRF_t$

Requirement	Topology
Access Hovering Information	Anchor Area, Communication Range, Node Topology.
Survive	Anchor Area, Communication Range, Node Topology.
Show Simulation	World.
Move	World.

Table 2.3.3: Reference Requirement-Topology table $(C)RRT_o_t$

Requirement	Dependency
Move	BoundedWorldDep

Table 2.3.4: Reference Requirement-Dependency table $(C)RReqD_t$

Legacy System	Function
Simulator UI	render.

Table 2.3.5: Reference Legacy System-Function table $(C)RLSF_t$

Legacy System	Topology
Simulator UI	World.

Table 2.3.6: Reference Legacy System-Topology table $(C)RLST_t$

Relation	Dependency
GUI	GUIDep
Neighborhood	NeighbourDep, PositionDep.
Node Interface	NodeIfDep.
HostToPiece Exchange	HostDep.
Show Simulation	HostDep.

Table 2.3.7: Reference Relation-Dependency table $(C)RRD_t$

2.3.2 Responsibilities Tables

Task	Description
display_information	Show the information on the mobile node UI.
access_local_information	Access the needed information available in the current position from an hosted piece of hovering information.
access_remote_information	Access the needed information available in the current position from other nodes.
inquire_world	Get the data needed by the show simulation process.
show_simulation	Create a displayable representation of the world.
move_task	Move with a certain behaviour inside the environment.
land_task	Jump between nodes in order to keep an high survivability.
clone_task	Replicate between nodes in order to keep an high survivability.

Table 2.3.8: Task table $(C)T_t$

Function	Description
render	Show a graphical representation of the simulated system.
get_position	Return the position of a mobile node.
discover_neighbours	Discover the neighbours of a mobile nodes.

Table 2.3.9: Function table $(C)F_t$

2.3.3 Topologies Tables

Topology	Description
Node Topology	The mobile node, positioned inside the world, seen as a topological constraint.
Anchor Area	Area associated to each hovering information, defined as a circular area with center into the <i>anchor location</i> and radius the <i>anchor radius</i> .
Communication Range	The maximum effective distance of a <i>p2p</i> mobile node communication.
World	The environment.

Table 2.3.10: Topology table $(C)Top_t$

Task	Topology
display_information	Node Topology
access_local_information	Anchor Area, Node Topology
access_remote_information	Anchor Area, Communication Range
inquire_world	World
show_simulation	World
move_task	World
land_task	Communication Range, Anchor Area
clone_task	Communication Range, Anchor Area

Table 2.3.11: Task-Topology table $(C)TTop_t$

Function	Topology
render	World.
discover_neighbour	Communication Range, World, Node Topology.
get_position	World, Node Topology.

Table 2.3.12: Function-Topology table $(C)FTop_t$

2.3.4 Dependency Tables

Dependency	Description
GUIDep	Make data available for GUI purposes.
NeighbourDep	A mobile node is a neighbour of another if both are in communication range.
NodeIfDep	A person needs a node interface in order to access hovering information.
PositionDep	Many operations (almost all) depend on position.
InquireDep	Get system information for simulation purposes.
BoundedWorldDep	A person cannot move outside the world.
HostDep	Pieces can performs certain operations only with its host and vice-versa.
RemoteNodeDep	Pieces and nodes can perform certain operations only with remote host.

Table 2.3.13: Dependency table $(C)D_t$

Task	Dependency
display_information	NodeIfDep
access_local_information	HostDep, NodeIfDep, PositionDep
access_remote_information	NeighbourDep, RemoteNodeDep, NodeIfDep, PositionDep
inquire_world	InquireDep
show_simulation	GUIDep, InquireDep
move_task	BoundedWorldDep
land_task	NeighbourDep, RemoteNodeDep, HostDep, PositionDep
clone_task	NeighbourDep, RemoteNodeDep, HostDep, PositionDep

Table 2.3.14: Task-Dependency table $(C)TD_t$

Function	Dependency
render	GUIDep.
discover_neighbours	NeighbourDep.
get_position	PositionDep.

Table 2.3.15: Function-Dependency table $(C)FD_t$

Topology	Dependency
Node Topology	HostDep.
Anchor Area	PositionDep.
Communication Range	PositionDep, NeighbourDep.
World	InquireDep, GUIDep, BoundedWorldDep

Table 2.3.16: Topology-Dependency table $(C)TopD_t$

2.4 Architectural Design

I re-iterated many times this phases. At last I have decided to not report the communication methods, but to assume that interaction are possible, regardless of the topology: if two mobile nodes are in rage, they can communicate.

Pieces of hovering information interact with remote mobile nodes, for asking information (probe action), asking permission for landing or cloning and to perform the true land or clone.

2.4.1 Transition Tables

Role	Task
Simulation Manager	inquire__world, show__simulation.
Person	move__task
Mobile node	display__information, access__local__information, access__remote__information, land__task, clone__task.
Piece of Hovering Information	land__task, clone__task, access__local__information.

Table 2.4.1: Transition Role-Task table $(C)TRT_t$

Task	Action
inquire__world	InquireNodes, InquireHoveringInformations, InquireEnvironment.
show__simulation	ShowSimulation.
access__local__information	NotifyInformation, GetInformation.
access__remote__information	AskInformation, GetInformation.
move__task	SenseAction, MoveAction.
land__task	Probe, AskPermission, Land.
clone__task	Probe, AskPermission, Clone.

Table 2.4.2: Transition Task-Action table $(C)TTA_t$

Resource	Function
SimulatorUIRes	render.
MobileResource	discover__neighbour, get__position.

Table 2.4.3: Transition Resource-Function table $(C)TRF_t$

Function	Operation
render	RenderWorld.
discover_neighbors	DiscoverNeighbors.
get_position	ObtainPosition.

Table 2.4.4: Transition Function-Operation table $(C)TFO_t$

Dependency	Interaction
GUIDep	Render Interaction
NeighbourDep	DiscoverNeighbor Interaction
PositionDep	ObtainPosition Interaction
NodeIfDep	DisplayInfoInteraction
InquireDep	InquireNodeInteraction, InquireHoveringInteraction
BoundedWorldDep	
HostDep	GetDataInteraction, ArakiriInteraction, PleaseDieInteraction
RemoteNodeDep	GetRemoteDataInteraction, ProbingInteraction, PermissionToLandInteraction, LandInteraction, CloneInteraction

Table 2.4.5: Transition Dependency-Interaction table $(C)TDI_t$

Dependency	Rule
NeighbourDep	Neighbour Rule
PositionDep	Accessible Rule, OkZone Rule, WarningZone Rule, ViolentZone Rule, DeathZone Rule
NodeIfDep	
InquireDep	
RemoteNodeDep	NoDuplication Rule
HostDep	Allocation Rule, Hosting Rule, Hosting Resource Rule
BoundedWorldDep	Bounded World Rule

Table 2.4.6: Transition Dependency-Rule table $(C)TDRu_t$

Topology	Space
Anchor Area	Anchor Area Space.
Communication Range	Communication Range Space.
World	World Space.
Node Topology	Node Space.

Table 2.4.7: Transition Topology-Space table $(C)TTopS_t$

2.4.2 Entities Tables

Action	Description
InquireNodes	Obtain information on mobile nodes, for showing the simulation.
InquireHoveringInformations	Obtain informatio on pieces of hovering information, for showing the simulation
InquireEnvironment	Obtain information about the world, for showing the simulation.
ShowSimulation	Show the simulation
NotifyInformation	Show on the mobile node ui interface the information.
GetInformation	Obtain the information from a currently hosted mobile node.
AskInformation	Ask information to a neighbor.
SenseAction	Obtain information from the environment, such as other people and points of interest.
MoveAction	Move inside the world.
Probe	Obtain information such as pieces, space and position from a neighbor mobile node.
AskPermission	Ask permission for landing or cloning to a mobile node.
Land	Perfom a landing operation toward a neighbour mobile node.
Clone	Perform a clone operation toward a neighbour mobile node.

Table 2.4.8: Action table $(C)A_t$

Operation	Description
RenderWorld	Render a representation of the world.
DiscoverNeighbour	Discover peers node inside the communication range.
ObtainPosition	Obtain the current node position using some location device.

Table 2.4.9: Operation table $(C)O_t$

Role	Action
Simulation Manager	InquireNodes, InquireHoveringInformations, InquireEnvironment, ShowSimulation.
Person	SenseAction, MoveAction.
Mobile node	NotifyInformation, GetInformation, AskInformation.
Piece of Hovering Information	Probe, AskPermission, Land, Clone.

Table 2.4.10: Role-Action table $(C)RA_t$

Resource	Operation
SimulatorUIRes	RenderWorld.
MobileResource	DiscoverNeighbour, ObtainPosition.

Table 2.4.11: Resource-Operation table $(C)RO_t$

2.4.3 Interactions Tables

Interaction (Social & Simulation)	Description
InquireNode Interaction	Mobile node inquisition for simulation analysis and management purposes.
InquireHovering Interaction	Pieces of hovering information inquisition for simulation analysis and management purposes.
Render Interaction	The system should interact with the GUI.

Table 2.4.12: Interaction table $(C)Isa_t$

Interaction (Hovering System)	Description
DisplayInfo Interaction	Show the mobile user the available information.
GetData Interaction	Mobile nodes ask hosted pieces of hovering information for their data.
GetRemoteData Interaction	Mobile nodes ask their neighbors their hovering information data.
PleaseDie Interaction	Mobile nodes asks hovering information to die, releasing resources.
Arakiri Interaction	Pieces of hovering information warn their host they are about to commit suicide.
Probing Interaction	Pieces of hovering information ask their host's neighbors information on their available space, their hosted pieces and their position.
PermissionToLand Interaction	Pieces of hovering information ask mobile nodes the permission to perform a transfer (land or clone).
Land Interaction	Pieces of hovering information perform a landing toward a remote mobile node.
Clone Interaction	Pieces of hovering information perform a clone toward a remote mobile node.
DiscoverNeighbor Interaction	Mobile nodes and pieces of hovering information use the mobile resource to get a list of neighbors.
ObtainPosition Interaction	Mobile nodes and pieces of hovering information use the mobile resource to get the current position.

Table 2.4.13: Interaction table $(C)Ihov_t$

Action	Interaction
InquireNodes	InquireNode Interaction
InquireHoveringInformations	InquireHovering Interaction
InquireEnvironment	
ShowSimulation	
NotifyInformation	
GetInformation	GetData Interaction
AskInformation	GetRemoteData Interaction
Probe	Probing Interaction
AskPermission	PermissionToLand Interaction
Land	Land Interaction
Clone	Clone Interaction

Table 2.4.14: Action-Interaction table $(C)AcI_t$

Operation	Interaction
RenderWorld	Render Interaction.
DiscoverNeighbour	DiscoverNeighbor Interaction.
ObtainPosition	ObtainPosition Interaction.

Table 2.4.15: Operation-Interaction table $(C)OpI_t$

2.4.4 Constraints Tables

Given A the anchor radius of an hovering information and D the distance of a piece of hovering information from its anchor center, the space is divided in four zones:

Ok Zone: $D \leq A$

Warning Zone: $A < D \leq A * 2$

Violent Zone: $A * 2 < D \leq A * 3$

Death Zone: $D < A * 3$

Rule	Description
Accessible Rule	A mobile node can access an hovering information, if it resides in its anchor area.
Allocation Rule	A mobile node cannot allocate more information than its buffer permits.
Neighbour Rule	A mobile node is reachable (is a neighbour) from another if both are in communication range.
No Duplication Rule	If a neighbour mobile node doesn't contain the same hovering information, landing or cloning is allowed.
Bounded World Rule	People cannot move outside the world.
Hosting Resource Rule	An hovering information can have only one host and can use only its host mobile resource.
Hosting Rule	A mobile node can communicate directly only with its hosted pieces of hovering information and viceversa.
OkZone Rule	A piece of hovering information is in its <i>ok zone</i> it's safe and may not perform cloning, landing or probing actions.
WarningZone Rule	A piece of hovering information is in its <i>warning zone</i> may begin to landing toward closer nodes, or perform a moderate cloning.
ViolentZone Rule	A piece of hovering information is in its <i>violent zone</i> is in danger and may try to survive at any cost.
DeathZone Rule	A piece of hovering information is in its <i>death zone</i> must commit suicide.

Table 2.4.16: Rule table $(C)Ru_t$

Interaction	Rule
InquireNode Interaction	
InquireHovering Interaction	
Render Interaction	
DisplayInfo Interaction	
GetData Interaction	Hosting Rule, Accessible Rule
GetRemoteData Interaction	Neighbour Rule, Accessible Rule
PleaseDie Interaction	Hosting Rule
Arakiri Interaction	Hosting Rule
Probing Interaction	Neighbour Rule, Hosting Resource Rule
PermissionToLand Interaction	Neighbour Rule, Hosting Resource Rule
Land Interaction	Allocation Rule, No Duplicate Rule, Neighbour Rule, OkZone Rule, WarningZone Rule, ViolentZone Rule, DeathZone Rule
Clone Interaction	Allocation Rule, No Duplicate Rule, Neighbour Rule, OkZone Rule, WarningZone Rule, ViolentZone Rule, DeathZone Rule
DiscoverNeighbor Interaction	Neighbour Rule, Hosting Resource Rule
ObtainPosition Interaction	Hosting Resource Rule

Table 2.4.17: Interaction-Rule table $(C)IRu_t$

Resource	Rule
MobileResource	Neighbour Rule, Hosting Resource Rule, Allocation Rule.

Table 2.4.18: Resource-Rule table $(C)ReRu_t$

Role	Rule
Simulation Manager	
Person	Bounded World Rule.
Piece of Hovering Information	Neighbour Rule, Accessible Rule, No Duplicate Rule, Hosting Resource Rule, Hosting Rule, OkZone Rule, WarningZone Rule, ViolentZone Rule, DeathZone Rule.
Mobile Node	Accessible Rule, Allocation Rule, Neighbour Rule, No Duplication Rule

Table 2.4.19: Role-Rule table $(C)RoRu_t$

Space	Rule
Anchor Area Space	Accessible Rule, OkZone Rule, WarningZone Rule, ViolentZone Rule, DeathZone Rule
Communication Range Space	Neighbour Rule
Node Space	Hosting Rule
World Space	Bounded World Rule

Table 2.4.20: Space-Rule table $(C)SRu_t$

2.4.5 Topological Tables

Space	Description
Node Space	Mobile node space: located in a specific position, holds a mobile resource.
Anchor Area Space	Area associated to each hovering information, defined as a circular area with center into the <i>anchor location</i> and radius the <i>anchor radius</i> .
Communication Range Space	The maximum effective distance of a <i>p2p</i> mobile node communication.
World Space	The environment.

Table 2.4.21: Space table $(C)S_t$

Space	Connection
Communication Range Space	Centered, Inside.
Node Space	Centered, Inside.
Anchor Area Space	Inside.
World Space	Inside.

Table 2.4.22: Space-Connection table $(C)SC_t$

Resource	Space
SimulatorUIRes	World Space
Node Resource	Node Space

Table 2.4.23: Resource-Space table $(C)ReS_t$

Role	Space
Simulation Manager	World Space.
Person	World Space, Anchor Space.
Mobile node	Node Space, Anchor Space.
Piece of Hovering Information	Anchor Space, Node Space.

Table 2.4.24: Role-Space table $(C)RoS_t$

2.5 Detailed Design

I haven't used the layering feature of SODA, so the design has only one possible architecture.

2.5.1 Carving Diagram

2.5.2 Mapping Tables

Agent	Role
Person	Person.
MobileNode	Mobile Node.
Hovering	Piece of Hovering Information.
Simulator	Simulation Manager.

Table 2.5.1: Mapping Agent-Role table $(C)MAR_t$

Society	Role

Table 2.5.2: Mapping Society-Role table $(C)MSR_t$

Artifact	Action
Body Artifact	MoveAction, SenseAction.
MobileUI Artifact	NotifyInformation.
MobileUIInterface Arti- fact	NotifyInformation.
Simulator Artifact	ShowSimulation, InquireNodes, Inquire- HoveringInformation, InquireEnvironment.
HoveringInformation Artifact	

Table 2.5.3: Mapping Artifact-Action table $(C)MAAc_t$

Artifact	Resource
UI Artifact	SimulatorUIRes.
MobileResource Arti- fact	MobileResource.

Table 2.5.4: Mapping Artifact-Resource table $(C)MArR_t$

Aggregate	Resource

Table 2.5.5: Mapping Aggregate-Resource table $(C)MAggR_t$

Artifact	Operation
UI Artifact	RenderWorld.
MobileResource Arti- fact	DiscoveNeighbour, ObtainPosition.

Table 2.5.6: Mapping Artifact-Operation table $(C)MArOp_t$

Artifact	Rule
Body Artifact	Bounded World Rule.
MobileResource Arti- fact	Allocation Rule, Neighbour Rule, Hosting Re- source Rule.

Table 2.5.7: Mapping Artifact-Rule table $(C)MArRu_t$

Space	Workspace
Node Space	Node Workspace.
Anchor Area Space	Anchor Area.
Communication Range Space	Communication Range.
World Space	World.

Table 2.5.8: Mapping Space-Workspace table $(C)MSW_t$

Interaction	Use
Render Interaction	Render Use
DisplayInfo Interaction	DisplayInfo Use
DiscoverNeighbor Interaction	Discover Use
ObtainPosition Interac- tion	ObtainPosition Use

Table 2.5.9: Mapping Interaction-Use table $(C)MIU_t$

Interaction	Manifest
DisplayInfo Interaction	NewInfo Manifest

Table 2.5.10: Mapping Interaction-Manifest table $(C)MIM_t$

Interaction	SpeakTo
InquireNode Interaction	NodeInquisition Speech
InquireHovering Interaction	HoveringInquisition Speech
GetData Interaction	GetData Speech
GetRemoteData Interaction	GetRemoteData Speech
PleaseDie Interaction	PleaseDie Speech
Arakiri Interaction	Arakiri Speech
Probing Interaction	Probe Speech
PermissionToLand Interaction	Permission to Land Speech
Land Interaction	Landing Speech
Clone Interaction	Clone Speech

Table 2.5.11: Mapping Interaction-SpeakTo table $(C)MISp_t$

Interaction	LinkedTo
Render Interaction	Render Link
DisplayInfo Interaction	DisplayInfo Link

Table 2.5.12: Mapping Interaction-LinkedTo table $(C)MIL_t$

2.5.3 Agent/Society Design Tables

Agent	Artifact
Simulator	Simulator Artifact.
Person	Body Artifact, MobileUI Artifact.
MobileNode	MobileUI Artifact, MobileUIInterface Artifact, MobileResource Artifact
Hovering	MobileResource Artifact, HoveringInformation Artifact.

Table 2.5.13: Agent-Artifact table $(C)AA_t$

Society	Agent

Table 2.5.14: Society-Agent table $(C)SA_t$

Society	Artifact

Table 2.5.15: Society-Artifact table $(C)SAr_t$

2.5.4 Environment Design Tables

Artifact	UsageInterface
Body Artifact	Sense, Move.
MobileUI Artifact	
MobileUIInterface Artifact	ShowInformation.
MobileResource Artifact	DiscoveNeighbour, ObtainPosition.
Simulator Artifact	ShowSimulation, InquireEnvironment
HoveringInformation Artifact	

Table 2.5.16: Artifact-UsageInterface table $(C)AUI_t$

Aggregate	Artifact

Table 2.5.17: Aggregate-Artifact table $(C)AggArt_t$

Aggregate	Agent

Table 2.5.18: Aggregate-Agent table $(C)AggAge_t$

2.5.5 Interaction Detailed Design Tables

Use	Protocol
Movement Use	world = Sense() choose_destination Move(Destination)
Render Use	ShowSimulation(World)
DisplayInfo Use	ShowInformation(Data)
DiscoverNeighbor Use	neigh_list = DiscoverNeighbor()
ObtainPosition Use	position = ObtainPosition()
InquireEnvironment Use	env_info = InquireEnvironment()

Table 2.5.19: Use-Protocol table $(C)UP_t$

Agent	Use
Person	Movement Use
MobileNode	ObtainPosition Use, DiscoverNeighbour Use, DisplayInfo Use.
Hovering	ObtainPosition Use, DiscoverNeighbour Use
Simulator	InquireEnvironment Use, Render Use

Table 2.5.20: Agent-Use table $(C)AgeU_t$

Artifact	Use
Body Artifact	Movement Use.
MobileUI Artifact	
MobileResource Artifact	DiscoverNeighbour Use, ObtainPosition Use
MobileUIInterface Artifact	DisplayInfo Use.
HoveringInformation Artifact	
Simulator Artifact	InquireEnvironment, Render Use

Table 2.5.21: Artifact-Use table $(C)ArtU_t$

SpeakTo	Protocol
InquireNode Speech	inquire_node reply node data
InquireHovering Speech	inquire_hoveringinformation reply hovering information data
GetData Speech	give_me_your_data reply my data
GetRemoteData Speech	have_you_this_data reply yes or no get_data reply data
PleaseDie Speech	please_die ack
Arakiri Speech	performing_arakiri ack
Probing Speech	have_you_space reply yes, no, position and if a equal piece is already present
PermissionToLand Speech	can_i_land reply yes or no
Land Speech	land wait for data receiving resume
Clone Speech	clone wait for data receiving start

Table 2.5.22: SpeakTo-Protocol table $(C)SP_t$

Agent	SpeakTo
MobileNode	InquireNode Speech, GetData Speech, GetRemoteData Speech, PleaseDie Speech, Arakiri Speech, Probe Speech, PermissionToLand Speech, Landing Speech, Clone Speech.
Piece of Hovering Information	InquireHovering Speech, GetData Speech, PleaseDie Speech, Arakiri Speech, Probe Speech, PermissionToLand Speech, Landing Speech, Clone Speech.
Simulator	InquireNode Speech, InquireHovering Speech

Table 2.5.23: Agent-SpeakTo table $(C)AgeSp_t$

Manifest	Protocol
DisplayInfo Manifest	new_info

Table 2.5.24: Manifest-Protocol table $(C)MP_t$

Agent	Manifest
Person	DisplayInfo Manifest

Table 2.5.25: Agent-Manifest table $(C)AgeM_t$

Artifact	Manifest
MobileUIInterface Artifact	ShowInfo Manifest

Table 2.5.26: Artifact-Manifest table $(C)ArtM_t$

LinkedTo	Protocol
Render Link	render(world)
DisplayInfo Link	show_info(info)

Table 2.5.27: LinkedTo-Protocol table $(C)LP_t$

Artifact	LinkedTo
MobileUIInterface Artifact	DisplayInfo Link.
MobileUI Artifact	DisplayInfo Link.
UI Artifact	Render Link
Simulator Artifact	Render Link

Table 2.5.28: Artifact-LinkedTo table $(C)ArtL_t$

2.5.6 Workspace Design Tables

Workspace	Description
Node Workspace	A mobile node workspace.
Anchor Area	Area bound to an hovering information.
Communication Range	Node communication range.
World	Simulation world.

Table 2.5.29: Workspace table $(C)W_t$

Workspace	Connection
Communication Range	Centered, Inside.
Node Workspace	Centered, Inside.
Anchor Area	Inside.
World	Inside.

Table 2.5.30: Workspace-Connection table $(C)WC_t$

Workspace	Artifact
Node Workspace	HoveringInformation Artifact, MobileResource Artifact, MobileUIInterface Artifact.
World	Simulator Artifact, Body Artifact, MobileUI Artifact.
Anchor Area	

Table 2.5.31: Workspace-Artifact table $(C)WArt_t$

Workspace	Agent
Node Workspace	MobileNode, Hovering
World	Person, Simulator, Simulator Manager, Hovering, MobileNode.
Anchor Area	Hovering.

Table 2.5.32: Workspace-Agent table $(C)WA_t$

3 MAS Implementation

The goal of this project is to simulate and analyze the hovering system used by the social system. In order to reduce the technological gap between the SODA process and the implementation, I have chosen to use *Jason+CArtAgO*[6, 7, 1] infrastructure.

3.1 Initialization

Prior to simulation, the sistem must be initialized. I couldn't find the required flexibility in the `mas2j` files, so I decided to create a new Agent, called *initiator*, which reads a configuration file and creates agents, workspaces and common artifacts specifying the needed configuration. Artifacts are configured by setting the proper initialization parameters, while agents are configured by telling them what they have to believe (using the *AgentSpeak* `tell` primitive).

The *initiator* creates the person agents and inserts them into the environment. For each person, a new mobile agent is created. Each person searches the *MobileUI Artifact*, which represent the mobile device itself. This artifact is created by the mobile node agent, thus, when the person agent finds it, both the mobile node and the person are ready to begin.

When the system is about to start, the *initiator* sends an `achieve(start)` to all the person agents. Each person agent turns on its mobile nodes using a new operation of *MobileUI Artifact*: `StartDevice`, which is linked to the *MobileUIInterface Artifact* and manifests itself to the mobile agent.

The last initialization operation is the *dissemination*, that is, the creation of the initial hovering agent inside some mobile nodes. For now the only dissemination method is the random dissemination, which assigns to each mobile node an hovering agent with a certain probability.

3.2 Environment Artifact

The environment artifact represents the world. Every Body Artifact, MobileResource and the SimulationArtifact is linked to it. The Environment Artifact provides operation for insert, sense and move people into the environment. The *MobileResource* is linked to the Environment Artifact for discovering neighbors, obtaining the position and sending and receiving messages.

3.3 Messaging System

For the communication, I have used the *AgentSpeak* primitives for communication between hovering agents and their host and to use a custom message-passing architecture for communicating with neighbors. Doing this, I assumed a simple dispatch-based architecture, easily implementable in real mobile devices. The messaging system is realized with two new operation `SendMessage` and `ReceiveMessage` into the *MobileResource Artifact*, usable by both the mobile node and the hosted hovering information.

Thanks to the `await` method of *CARTAgO*, the receive message operation blocks the execution flow, completing only when a new message is available.

3.4 Obtaining the position

Obtaining the position is an important task and must be performed continuously. I have defined a new operation `StartObtainingPosition` in *MobileResource Artifact*: using the `await` method, the artifact polls the position, manifesting it to the focused agents.

3.5 Social System

People have to move around the world, basing on their environment perception. In the real world, a person has eyes, through which can recognize other people and point of interest and approximate their distance.

With the `Sense` operation, each person obtain a list of other people and points of interest. Behaviors can be build from here.

For now, the only implemented behavior is a sort of random walk: a person chose a random position and reach it at a random speed.

3.6 Hovering Survive Strategy

The behavior of hovering agents depends on various factor. I have decided to rise the abstraction level by creating the concept of *Defcon*. At defcon

5 and 4, an hovering agent thinks it's safe and just monitor it's position, speed and so on. At defcon 3, the hovering starts probing the neighbors, creating a list of potential destination for landing or cloning operation. At defcon 2, it tries to land on a mobile node closer to the anchor area. On defcon 1 it clone itself on the best neighbor.

For now the defcon level is computed by a linear combination of two exponential function with values about between 0 and 1. With values closer to 1, the defcon level is low, while with values closer to 0, the defcon is high. The function may depend on various factors. I have implemented:

$$v = e^{-\frac{Distance}{DeathZoneLimit} * ZoneFactor} * e^{\frac{NumNeighbors}{10}}$$

$$ZoneFactor = \begin{cases} 1, & \text{if ok zone} \\ 2, & \text{if warning zone} \\ 3, & \text{if violent zone} \\ 4, & \text{if death zone} \end{cases}$$

$$DefconLevel = \begin{cases} 1, & \text{if } 0.8 \leq v \\ 2, & \text{if } 0.68 \leq v < 0.8 \\ 3, & \text{if } 0.42 \leq v < 0.68 \\ 4, & \text{if } 0.35 \leq v < 0.42 \\ 5, & \text{if } 0.0 \leq v < 0.35 \end{cases}$$

Using the defcon level, the hovering agent can adjust its interactions basing on multiple factors.

3.7 Hovering Landing and Cloning operations

In the real world, a landing operation may consist in the agent serialization. The agent packs and transfers itself into the new mobile node, which will resume it. On the other hand, the clone operation should send the agent source and an initial sets of believes to the target mobile node. The new host will create a new agent and *tell* him the initial believes.

I have implemented a similar mechanism, but of course the hover agent doesn't send code or the serialized itself, because they are in the same local system.

When an hovering information wants to land on another mobile node, after asking the permission, it sends a message, tells the old host that it's dead (using the Arakiri Speech) and waits for the resume of the new host.

It may happen that the communication fails during the landing. In this case the hovering agent accepts it's fate and dies alone.

3.8 Odds and Ends

The JASON complexity raised, and I have faced some problems, without knowing the exact cause. They are surely due to a some implementation errors, but I wasn't able to spot them and the debugging wasn't so helpful.

Some random error and crashes of the JASON parser still occurs and I have no idea why. I usually just restart the application.

This and that leads me to change, restore and reedit some SODA interaction. For instance the hovering information tells (and untells) its data to the mobile node (it's the reverse situation of the *GetData Speech*).

Another example is the *GetRemoteData Speech*. The mobile node doesn't asks every mobile node which data they have, but instead it sends to every neighbor its information everytime something changes. This lead to the construction of a database, which it's easily queried by the mobile node.

Moreover I haven't implemented some features, such as smarter movement for people, different and better dissemination process, landing and cloning basing on availability and accessibility of the hovering information and system analysis. All those features (except mayhaps for people movement) are easily implementable using the data available to each agent, but I have give up due to some random and frustrating errors.

There is also a *bug*: the communication range is not well managed, so for a message is succesfully sent it's necessary that only the sender has the receiver in range. In a real context, they must be both in range.

4 System Usage

The `InitiatorArtifact` loads the configuration contained in `simulation.yaml`, which must me located in the current working directory. The file is a configuration file using the `yaml` syntax and contains information about the world, people and hovering information that must be present in the system. An example is:

```
1 World:
2   width: 800
3   height: 800
4
5 Simulation:
```

```

6   gui_width: 800
7   gui_height: 800
8   gui_refresh_rate: 500
9   dissemination: 'random'
10  dissemination_param: 0.8
11
12  Analysis:
13    analysis_rate: 1000
14
15  People:
16    - name: 'pippo'
17      behaviour: 'random'
18      xpos: 10
19      ypos: 20
20      device_range: 200
21      device_storage: 100
22    - name: 'pluto'
23      behaviour: 'random'
24      xpos: 20
25      ypos: 10
26      device_range: 200
27      device_storage: 100
28
29  Hovering:
30    - name: 'gentoo_store'
31      xanchor: 20
32      yanchor: 70
33      anchor_radius: 150
34      data_size: 10
35      data: "If it moves, compile it!"
36    - name: 'google_store'
37      xanchor: 400
38      yanchor: 400
39      anchor_radius: 150
40      data_size: 20
41      data: "Don't be evil!"
42    - name: 'apple_store'
43      xanchor: 100
44      yanchor: 400
45      anchor_radius: 100
46      data_size: 40
47      data: "Think Different!"

```

By starting the JASON system, the initiator agent creates the other agents and starts the system.

Figure 4.0.1, 4.0.2, 4.0.3 and 4.0.4 show screenshots of the running system. The purple lines that connect people with the hovering information anchors, assert that the mobile node contains an piece of the hovering in-

formation. The size of the hovering information, related to the mobile node capacity, is shown as a colorful bar inside the mini-buffer at each people side.

A grey dotted line joining two peoples, indicates that the respective mobile nodes communicated in the last two seconds.

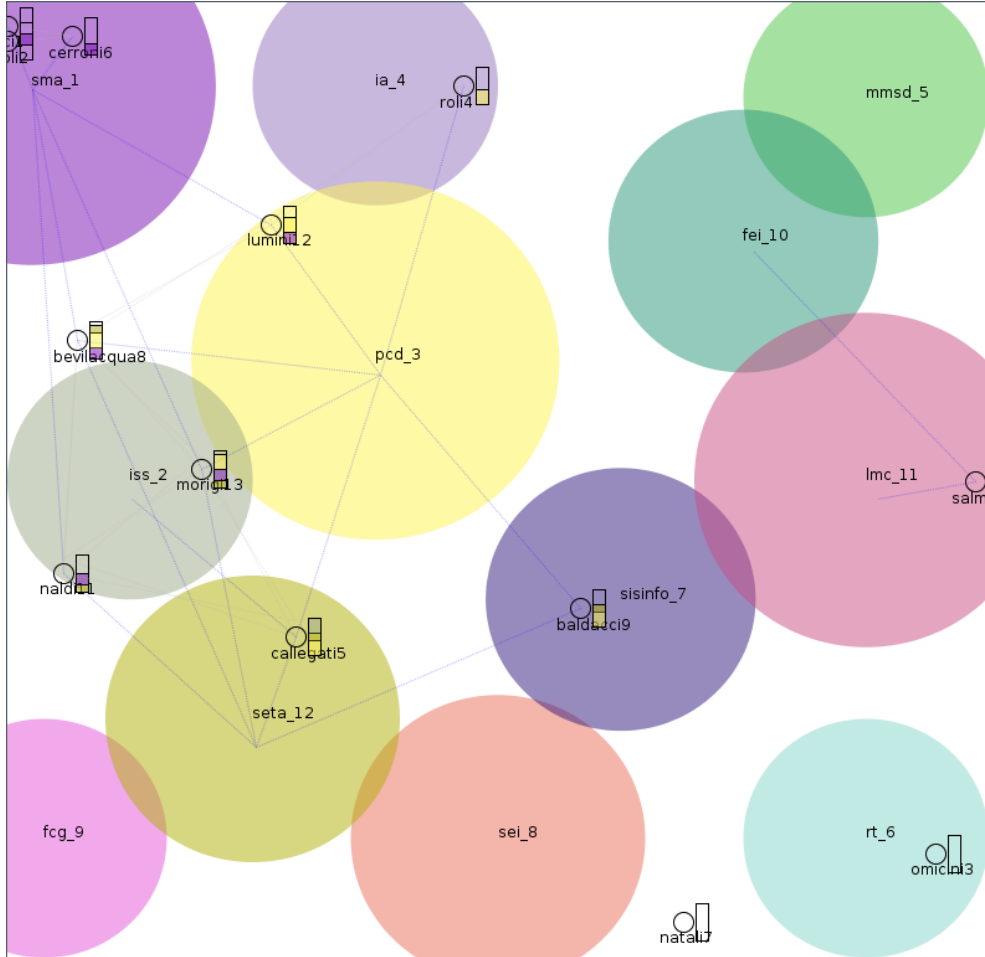


Figure 4.0.1: Screenshot 1



Figure 4.0.2: Screenshot 2

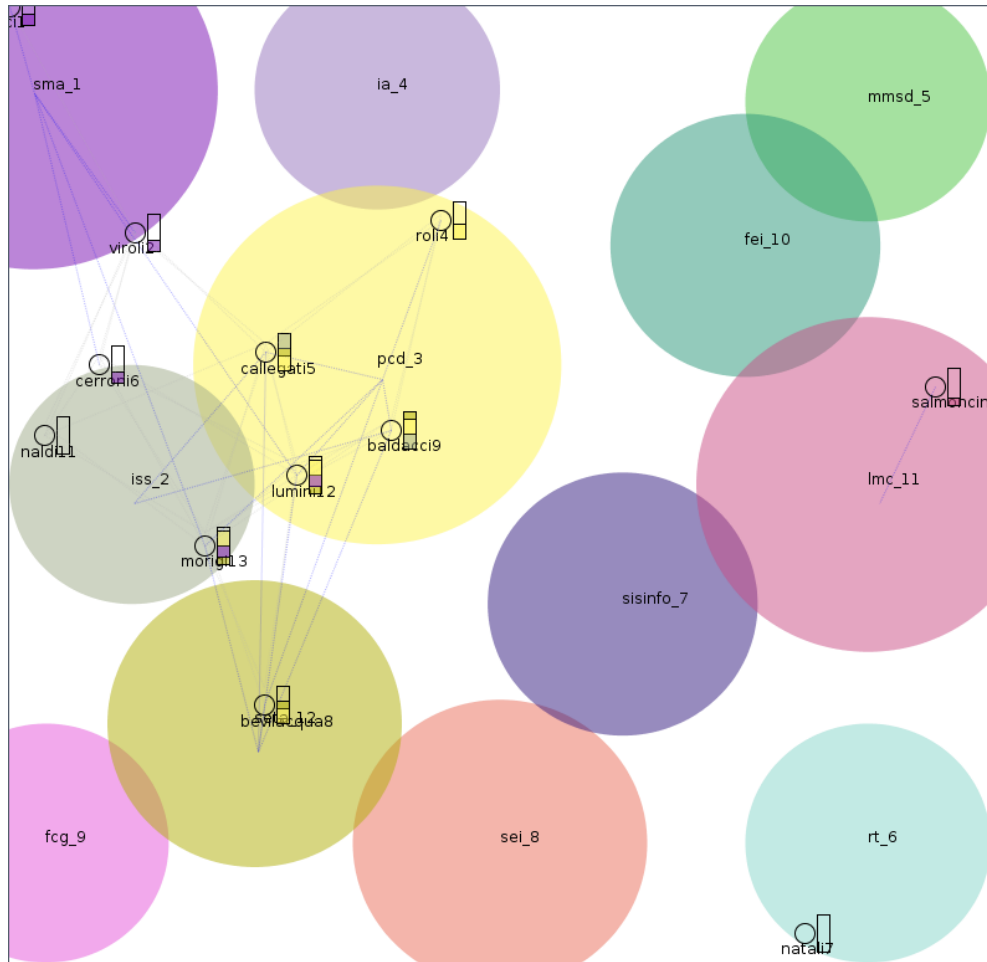


Figure 4.0.3: Screenshot 3



Figure 4.0.4: Screenshot 4

References

- [1] Andrea Santi Alessandro Ricci. *CARTAgO By Example*. DEIS, Università di Bologna, Italy, 2.0.1 edition, October 2010.
- [2] Andrea Omicini Ambra Molesini. *SODA - Societies in Open Distributed Agent spaces*. ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA, October 2012. <http://soda.alice.unibo.it>.
- [3] Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, and Alessandro Ricci. Organisation and environment oriented programming. In *13th European Agent Systems Summer School*, July 2011.
- [4] Alfredo A. Villalba Castro, Giovanna Di Marzo Serugendo, and Dimitri Konstantas. Hovering information - infrastructure-free self-organising location-aware information dissemination service. In *2nd ERCIM Workshop on eMobility*, 2008.
- [5] Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems: and artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192, September 2011.
- [6] Alessandro Ricci, Andrea Santi, Michele Piunti, Mirko Viroli, Andrea Omicini, Marco Guidi, and Mattia Minotti. Cartago website. <http://cartago.sourceforge.net/>.
- [7] John Wiley & Sons. Programming multi-agent systems in agentspeak using jason. <http://jason.sf.net/jBook/>, October 2007.