Project: "El Farol Bar Problem and the Minority Game"

GROUP #4:

Juan Bazurto

Ehud Cohen

Richard Pellosie

Justin Phalon

Mike Puntolillo

Nicholas Tse

Table of Contents

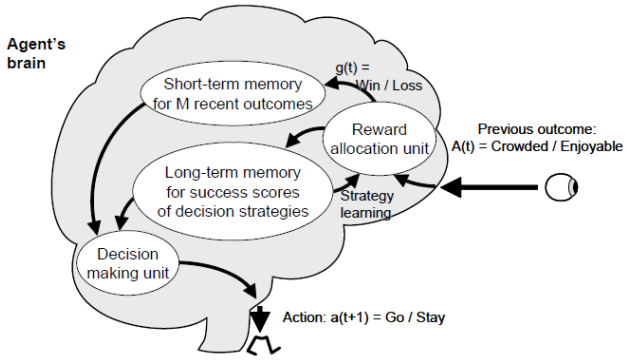
Individual Requirements Breakdown

	Juan Bazurto	Ehud Cohen	Richard Pellosie	Justin Phalon	Mike Puntolillo	Nicholas Tse
Project Management	16.00%	17.00%	16.00%	17.00%	16.00%	18.00%
Section 3	33.30%		50.00%			16.60%
Section 4	25.00%	75.00%				
Section 5	10.00%	30.00%	30.00%		30.00%	
Section 6	100.00%					
Section 7	10.00%		10.00%	10.00%	16.00%	54.00%
Section 8				100.00%		
Section 9				100.00%		
Section 10				100.00%		

Customer Statement of Requirements

The program designed needs to be able to simulate agents deciding whether or not to go to a venue based on attendance in previous weeks. The user should input a percentage P. If more than P of the total number of agents attended, then the experience was not enjoyable due to overcrowding. If less than P attended, then the experience was enjoyable.

Figure 1

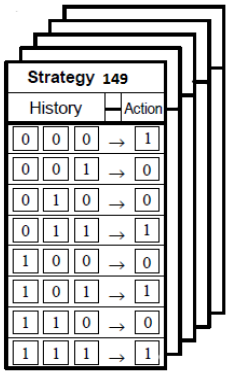


A visualization of the agent's brain for making decisions based on previous outcomes

The program should allow for the user to set the number of agents in the simulation. The simulation should be able to run for multiple rounds, the amount of which is chosen by the user. In addition the user should be able to vary the “memory” of each agent, how many previous rounds the agents consider when making a decision.

An agent should have multiple strategies when making a decision. Every round these different strategies should keep a score to monitor their success. At the end of each round, all strategies which predicted the correct outcome will gain a point. The agent will use these “scores” to decide which strategy will be used in the next round’s decision. At the end of the simulation the program should output the strategy with the highest score and the agent who had the highest success rate with his strategies.

Figure 2



A visualization of a strategy giving a decision based on every case with a memory of 3

The program should allow for agents to “die” or stop participating, simulating death, losing interest in the venue, or other factors. The rate at which agents die should be dependent on their age. A table provided by the United States Social Security Administration gives the probability of dying based on a person’s age. The agents will

keep track of their age and die in a manner statistically consistent with the projections of the United States Social Security Administration.

The program should allow for a human user to play in lieu of a simulated agent and decide when to attend or not attend the bar. The program should record the user's responses and relate them to the strategies in place. The strategies that match those of the user would be weighted since they are like real human decisions.

The simulation should also assign more weight to more recent outcomes. This rate should also be adjustable by the user defaulted to so no weight is assigned.

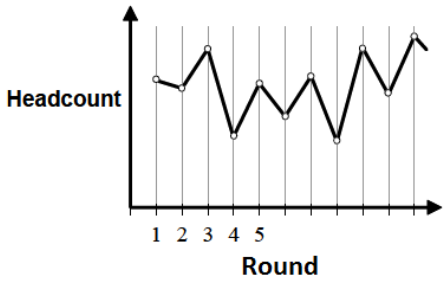
Figure 3

$$\sigma_{ij}(t) = \begin{cases} \alpha \cdot \sigma_{ij}(t-1) & , \text{ if } \left(a_{ij} \cdot A(t)\right) < 0 \\ \alpha \cdot \sigma_{ij}(t-1) + 1 & , \text{ if } \left(a_{ij} \cdot A(t)\right) > 0 \end{cases}$$

An example of a weighting outcomes based on how recent they are

The program should be able to plot the data collected. This should include the success rate of the highest rated strategy over the rounds and the amount of agents attending per round. 3-D surface plots with two variables (such as the number of rounds and number of players) may also be plotted at the user's request.

Figure 4



Example output plot

List of Requirements:

1. Agents deciding whether or not to go to a venue based on previous rounds

2. *Number of agents and rounds adjustable
3. Agents with the minority decision win
4. *Agents have a limited memory of previous rounds
5. *Multiple strategies per agent
6. Strategies keep score to determine most successful
7. *A variable β is used to set the mortality for the agents
8. Human participant resulting in weighted strategies
9. *Weight assigned based on how recent an outcome is
10. Plots success rate vs. rounds and number of agents in attendance vs. rounds

NOTE: Requirements denoted with * are parameter adjustable by the user

Glossary of Terms

Administrator- The program's main user. One who sets all parameters and interprets output.

Agent-A simulated player of the minority game.

Loss- An agent loses when it makes the same decision as the majority of the other agents.

Memory-The set of variables in which strategy score, agent score, and a record of wins and losses is stored.

Mortality-The ability of the system to age agents and simulate their “deaths” and replacement.

Strategy-A set of decisions that are to be made based on a given win-loss sequence.

Win- An agent wins when it makes the decision that the least number of other agents made.

Functional Requirements Specification

The target customers are any venues that deal with many patrons and limited space. The example given of a bar works very well, however many other examples apply. Gyms with many members and limited machines and weights, amusement parks dealing with increased ride wait times with more visitors, restaurants where wait times for tables discourages patrons, and department stores where many visitors results in messier displays and long checkouts. These are just few of the many examples of potential customers interested in predicting the attendance of their venues.

The program also could be marketed commercially to let the patrons of these venues predict when to visit. Obviously when a setting is overcrowded or uncomfortable the experience is negative. Anyone who has ever been to a bar too crowded to move around or an amusement park with 2+ hour wait times for rides can attest to this. Whether used by college students to figure out when a bar would be less crowded, a family with children trying to find the shortest wait times at a park, or a mother hoping to grocery shop with well stocked shelves and quick checkouts, the program has a market with the general public.

Companies that sell products at the venues listed above could be interested in sponsoring the program. In the case of the bar MillerCoors LLC, which produces Miller Lite, Coors Light, and several other types of beer, advertise having a good time and being with friends. A program that would allow predictions of patron behavior would let bars know when to stock up on product and potentially increase sales.

Actors	Actors Goals	Use Case Name
Administrator	To choose number of agents participating in game	NumAgent (UC-1)

Administrator	To choose the number of rounds the agents will participate in	NumRound (UC-2)
Administrator	To choose the number of strategies utilized by each player	NumStrat (UC-3)
Administrator	To choose the number of rounds saved to memory	MemSize (UC-4)
Administrator	To toggle mortality into and out of the game	Mortality (UC-5)
Administrator	To start a game in which the user can play as an agent	SubAgent (UC-6)
Player	To play as an agent	SubAgent (UC-6)
Administrator	To choose charts to print out, which can display results of all games for successful strategy, number of people in bar each round, etc	PrintOut (UC-7)
MATLAB	To print out the chosen charts	PrintOut(UC-7)
Administrator	To Start the game	Start(UC-8)

Use Case UC-1:	NumAgent
Related Requirements:	REQ2
Initiating Actor:	Administrator
Actor's Goal:	To choose number of agents participating in game
Participating Actors:	
Preconditions:	None
Postconditions:	Number of participating Agents is set
Flow of Events for Main Success Scenario:	
→	1. **Administrator** (a) selects the menu item "Number of Agents" (b) types in value
	4. **System** sets number of Agents as per entry

Use Case UC-2:	NumRound
Related Requirements:	REQ2
Initiating Actor:	Administrator
Actor's Goal:	To choose the number of rounds the Agents will participate in
Participating Actors:	
Preconditions:	None (though best to set amount of agents first)
Postconditions:	Number of rounds for Agents to participate in is set
Flow of Events for Main Success Scenario:	
→	1. **Administrator** (a) selects the menu item "Number of Rounds" and (b) types in value
	2. **System** sets number of Rounds as per entry

Use Case UC-3:	NumStrat
Related Requirements:	REQ5, REQ6
Initiating Actor:	Administrator
Actor's Goal:	To choose the number of strategies utilized by each Agent
Participating Actors:	
Preconditions:	None (though best to set amount of agents first)
Postconditions:	Number of strategies for each Agent to use is set
Flow of Events for Main Success Scenario:	
→	1. **Administrator** (a) selects the menu item "Number of Strategies" and (b) types in value
	2. **System** sets number of Strategies as per entry

Use Case UC-4:	MemSize
Related Requirements:	REQ4
Initiating Actor:	Administrator
Actor's Goal:	To choose the number of Rounds saved to Memory
Participating Actors:	
Preconditions:	Need total number of Rounds that will be played
Postconditions:	Number of Rounds saved to Memory is set
Flow of Events for Main Success Scenario:	
→	2. **Administrator** (a) selects menu item "Size of Memory" and (b) types in value
	3. **System** sets number of Rounds saved to Memory as per User entry

Use Case UC-5:	Mortality
Related Requirements:	REQ7
Initiating Actor:	Administrator
Actor's Goal:	To toggle mortality in and out of the game
Participating Actors:	
Preconditions:	None
Postconditions:	Mortality is set to be included in the game or not
Flow of Events for Main Success Scenario:	
→	1. **Administrator** (a) selects the pushbutton item "Mortality" (a) selects Yes to turn Mortality on, No to turn Mortality off
	2. **System** records whether or not to include Mortality in the Game

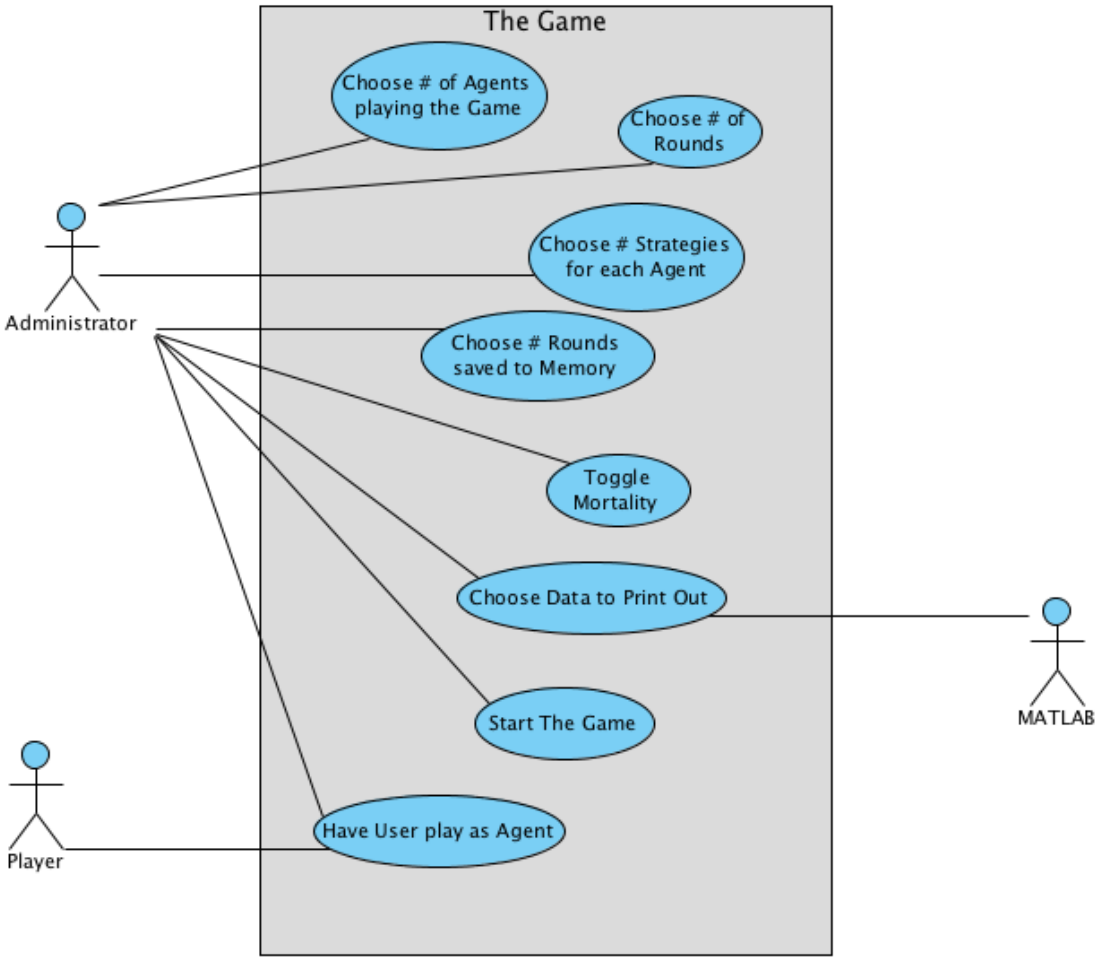
Use Case UC-6:	SubAgent
Related Requirements:	REQ8
Initiating Actor:	Administrator
Actor's Goal:	To start a game with the user playing as an agent
Participating Actors:	Player
Preconditions:	None
Postconditions:	The User is now actively playing the game
Flow of Events for Main Success Scenario:	
→	1. **Administrator** selects the Pushbutton item "Play as Agent" (a) Yes to play, (b) No to not play
→	2. **Administrator** selects Pushbutton item "Go" or "Stay" – Whether his agent will go or stay
←	2. **System** creates an actor "Player" for use by the **Administrator** in the game

Use Case UC-7:	PrintOut
Related Requirements:	REQ9, REQ10, REQ8, REQ2, REQ7
Initiating Actor:	Administrator
Actor's Goal:	To choose charts to print out, which can display results of all games for successful strategy, number of people in bar each round, etc
Participating Actors:	MATLAB
Preconditions:	A game has been played
Postconditions:	The charts are displayed in MATLAB that contains details about the game
Flow of Events for Main Success Scenario:	
→	1. **Administrator** selects the items in the Menu "Print Out" which he wants to have printed
←	2. **System** (a) accesses **MATLAB** and (b) sends data during Game Play
←	3. **MATLAB** prints out data after end of Game

Use Case UC-8:	Start
Related Requirements:	ALL
Initiating Actor:	Administrator
Actor's Goal:	To start the Game
Participating Actors:	MATLAB, Player
Preconditions:	A game has been played
Postconditions:	Game ends, data is collected, The charts chosen are displayed in MATLAB that contains details about the game
Flow of Events for Main Success Scenario:	
→	1. **Administrator** selects the Menu item "Play"
	Include::*NumAgent, NumRound, NumStrat, MemSize, Mortality, SubAgent, PrintOut*
←	2. **System** (a) checks all entries in the GUI for errors (b) Then runs the game
Flow of Events for Alternate Scenarios:	
	2a. **Administrator** enters an invalid number in any Menu item (ex. negative value, Rounds to Memory value higher than total number of Rounds available as set in UC-2)
	1. **System** (a) detects error, (b) signals to the Actor, (c) re-requests entry
	2. **Administrator** provides valid entry

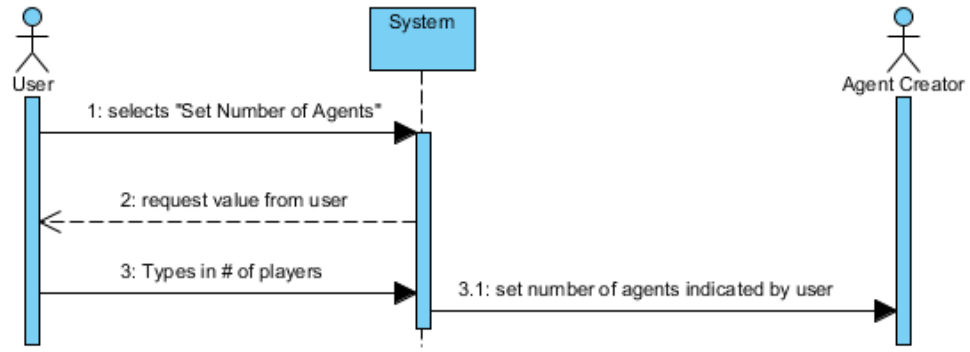
Use Case UC-8:	Start
Related Requirements:	ALL
Initiating Actor:	Administrator
Actor's Goal:	To start the Game
Participating Actors:	MATLAB, Player
Context of Use:	The game will be played in situations where the user wishes to find out the possible attendance of consumers at local area bars, stores or other places. Based of previous attendance data and other factors, the Game will be able to identifiy at any given time how many people will decide to go or stay away from a location.
Scope:	The system being considered is the strategies users use to determine staying in or going to a location based on previous experience
Stakeholders and Interest:	Gyms, Stores, Bars, Parks and companies that service them, will have an easier time adjusting their stock depending on the amount of customers they can expect at any given time
Minimal Guarantee:	The Game gives a good idea of how consumers might strategize given certain conditions
Success Guarantee:	The Game accurately depicts real-life strategies of the average consumer
Preconditions:	A game has been played
Postconditions:	Game ends, data is collected, The charts chosen are displayed in MATLAB that contains details about the game
Flow of Events for Main Success Scenario:	
→	1. **Administrator** selects the Menu item "Play"
	Include::*NumAgent, NumRound, NumStrat, MemSize, Mortality, SubAgent, PrintOut*
←	2. **System** (a) checks all entries in the GUI for errors (b) Then runs the game
Flow of Events for Alternate Scenarios:	
	2a. **Administrator** enters an invalid number in any Menu item (ex. negative value, Rounds to Memory value higher than total number of Rounds available as set in UC-2)

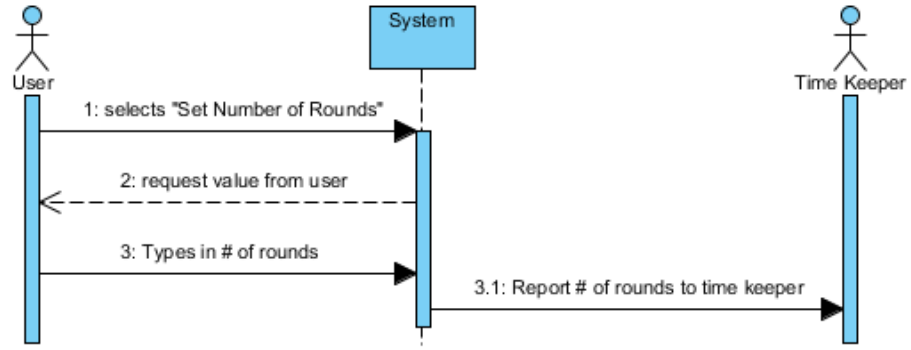
	1. **System** (a) detects error, (b) signals to the Actor, (c) re-requests entry
	2. **Administrator** provides valid entry



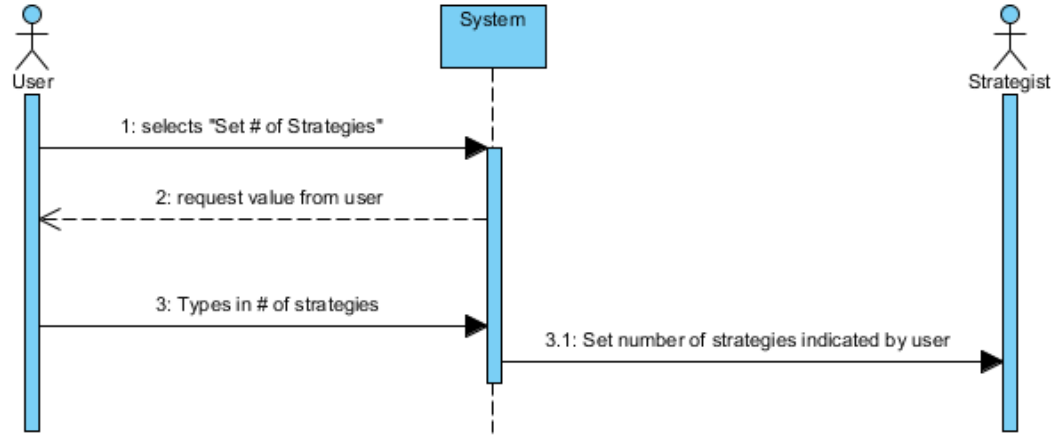
Traceability (vs. Requirements)	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
REQ1: Agents deciding whether or not to go to a venue based on previous rounds								X
REQ2: Number of agents and rounds adjustable	X	X					X	X
REQ3: Agents with the minority decision win								X
REQ4: Agents have a limited memory of previous rounds				X				X
REQ5: Multiple strategies per agent			X					X
REQ6: Strategies keep score to determine most successful			X					X
REQ7: A variable β is used to set the mortality for the agents					X		X	X
REQ8: Human participant resulting in weighted strategies						X	X	X
REQ9: Weight assigned based on how recent an outcome is							X	X
REQ10: Plots success rate vs. rounds and number of agents in attendance vs. rounds							X	X

Use Case 1:

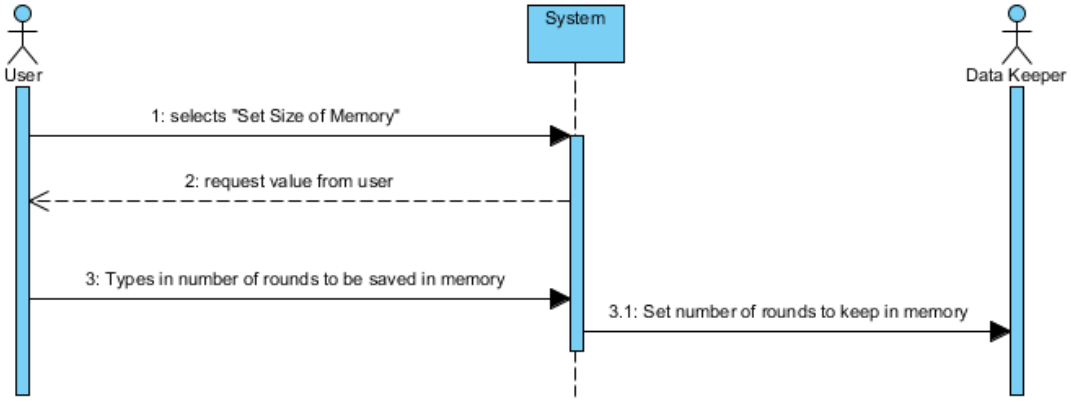


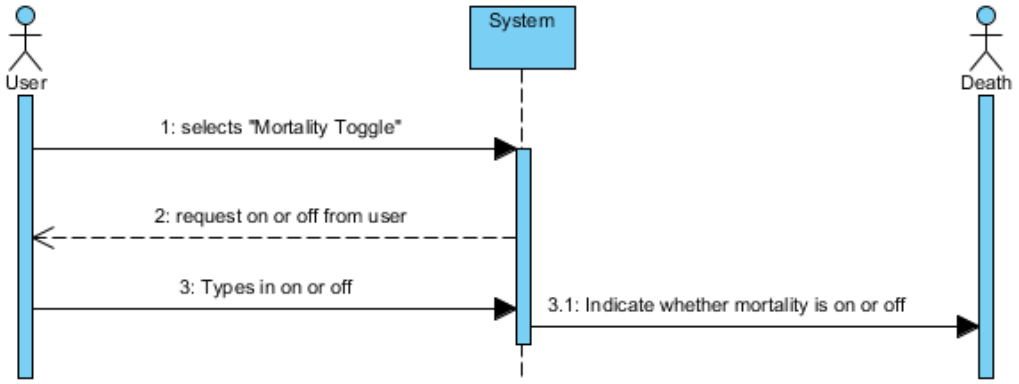


Use Case 3:

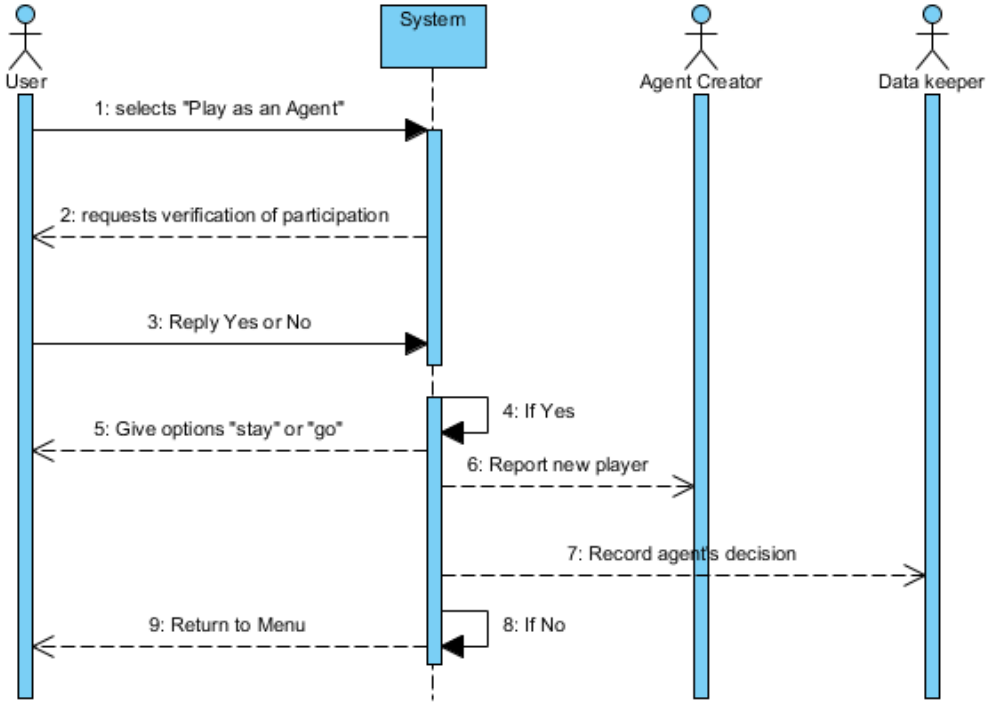


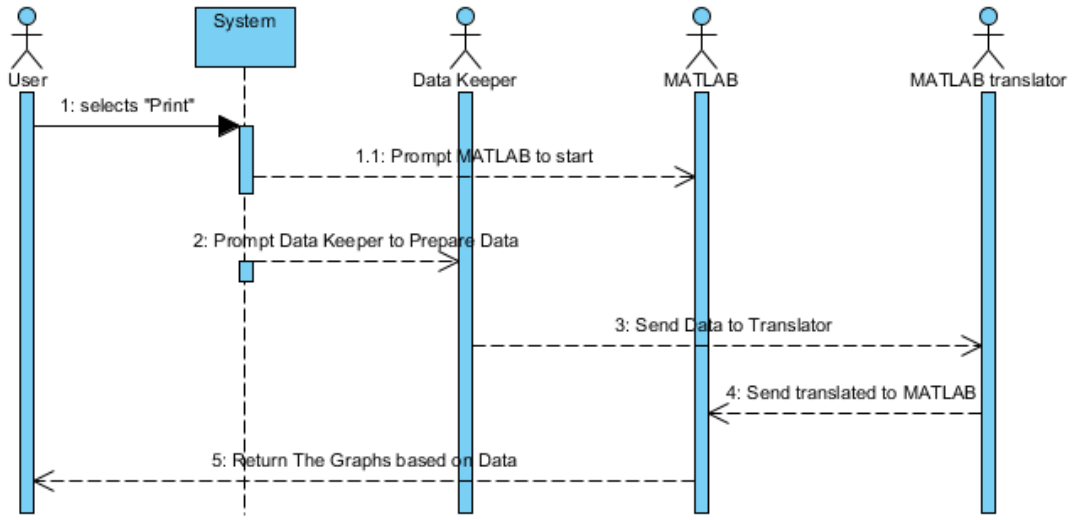
Use Case 4:



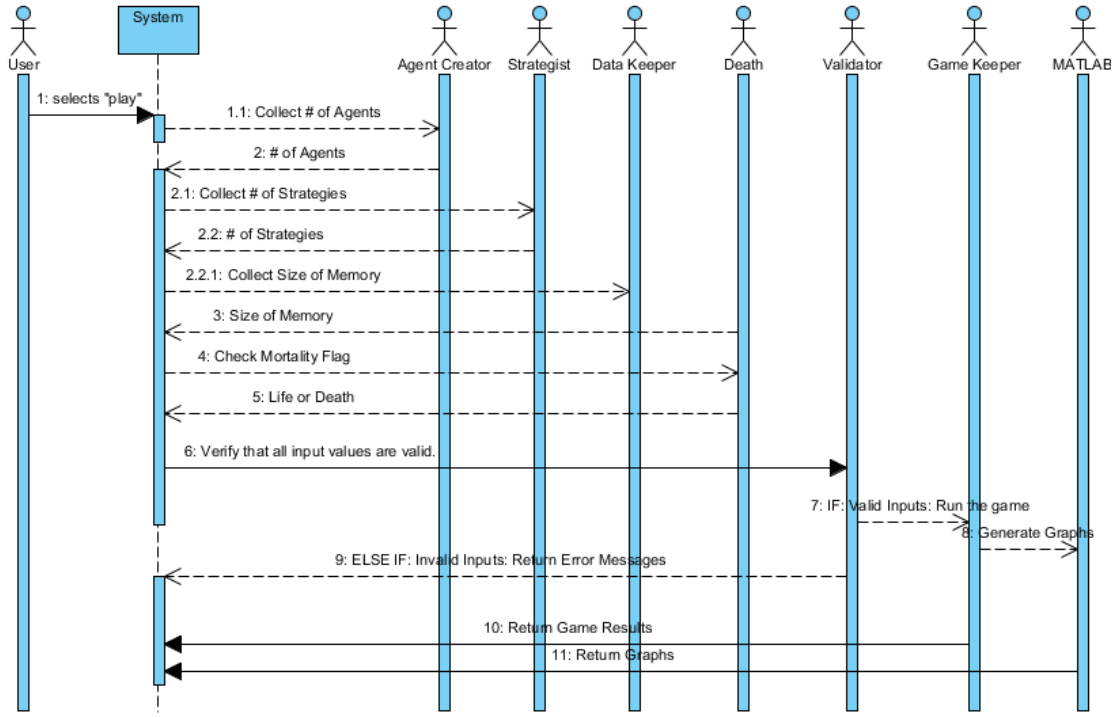

User
System
Death
1: selects "Mortality Toggle"
2: request on or off from user
3: Types in on or off
3.1: Indicate whether mortality is on or off

Use Case 6:


User
System
Agent Creator
Data keeper
1: selects "Play as an Agent"
2: requests verification of participation
3: Reply Yes or No
4: If Yes
5: Give options "stay" or "go"
6: Report new player
7: Record agent's decision
9: Return to Menu
8: If No


User
System
Data Keeper
MATLAB
MATLAB translator
1: selects "Print"
1.1: Prompt MATLAB to start
2: Prompt Data Keeper to Prepare Data
3: Send Data to Translator
4: Send translated to MATLAB
5: Return The Graphs based on Data

Use Case 8:


User
System
Agent Creator
Strategist
Data Keeper
Death
Validator
Game Keeper
MATLAB
1: selects "play"
1.1: Collect # of Agents
2: # of Agents
2.1: Collect # of Strategies
2.2: # of Strategies
2.2.1: Collect Size of Memory
3: Size of Memory
4: Check Mortality Flag
5: Life or Death
6: Verify that all input values are valid.
7: IF: Valid Inputs: Run the game
8: Generate Graphs
9: ELSE IF: Invalid Inputs: Return Error Messages
10: Return Game Results
11: Return Graphs

Non-functional Requirements

As previously outline in section 3 of this report, the program has several functional requirements. To recount some: agents deciding to go to a venue or not, varying number of agents and rounds, multiple strategies to decide, and limited short term memory. While these requirements define the program's functions, there are also non-functional requirements.

The user interface needs to be accessible for the user. All variables and fields of interest should be displayed clearly so the desired simulation can be run. The runtime for the most complex situation should still be relatively short. The user does not want to wait an excessive amount of time for the results. The system should also be able to handle values not expected, such as a negative number of agents. In the event of a failure in the simulation, the program should reset and notify the user of the failure.

An email for support would be useful for users. In the event of an error a report could be sent leading to fixes. In addition the email could allow for users to recommend new features. An increase in maintainability directly increases the appeal for the user.

Domain Analysis

Concepts:

Responsibility Description	Type	Concept Name
Coordinates actions and data of associated concepts; essentially the controller of the system.	D	Main
Container for the game's settings, including the number of agents, the number of strategies, agent mortality, etc.	K	Global

Creates and oversees a set of strategies.	D	Agent
Container for an agent's set of strategies; used to determine attendance at bar.	K	StrategyList
Analyzes an agent's current set of strategies in order to determine the most successful, and reports whether or not an agent should attend the bar.	D	Decider
Updates both an agent's memory as well as the score of its strategies based on the previous round's outcome.	D	Updater
Provides the user with a straightforward way of initializing game settings via a GUI.	D	DataEntry
Provides the user with a clear and concise way of viewing program feedback via a GUI.	D	StatusDisplay

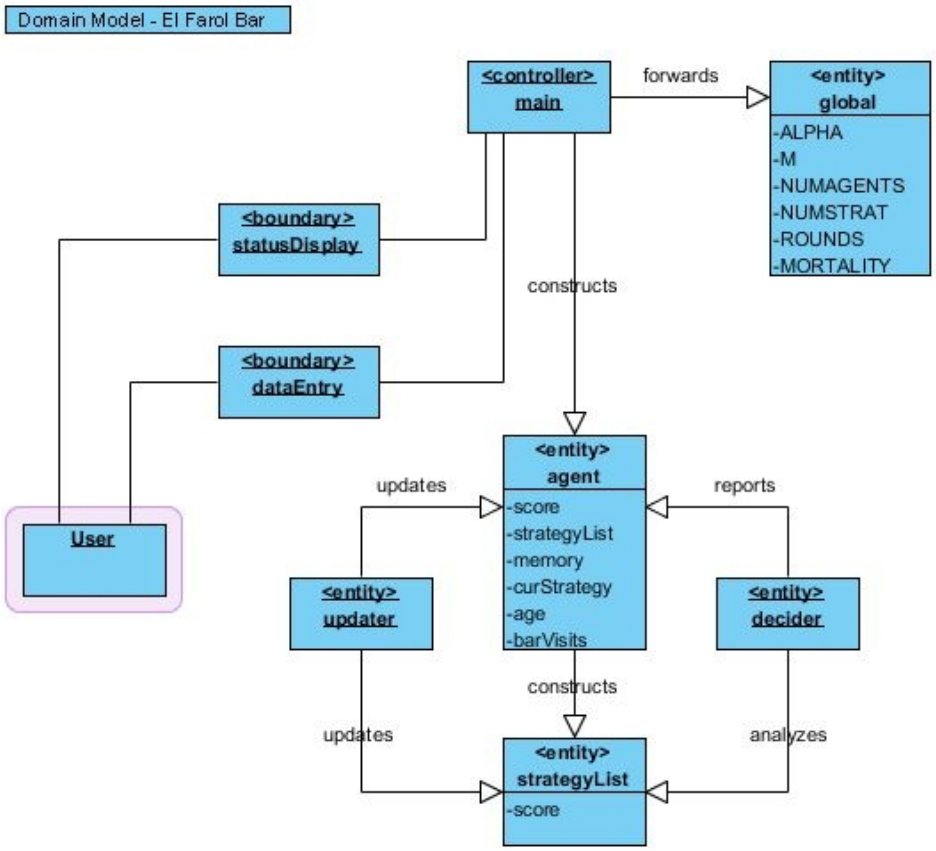
Associations:

Concept Pair	**Association Description**	**Association Name**
Main - Global	Main passes data fields originally entered via DataEntry into Global.	forwards
Main - Agent	Main calls the Agent class constructor.	constructs
Agent - StrategyList	Agent adopts a set of strategies randomly out of a pool.	constructs
Updater - Agent	Updater adjusts an Agent's memory according to the outcome of the previous round.	updates
Updater - StrategyList	Updater adjusts a StrategyList's scores and strategies according to the outcome of the previous round.	updates
Decider - StrategyList	Decider analyzes the available strategies within StrategyList and determines the most successful one.	analyzes
Decider - Agent	Decider reports the appropriate course of action to an Agent.	reports

Attributes:

Concept	**Attributes**	**Attribute Description**
Global	ALPHA	Plays a role in the memory decay factor of an agent.
	M	Number of past outcomes considered when making a decision to go to the bar or not.
	NUMAGENTS	The number of agents participating in the game.
	NUMSTRAT	The number of strategies available to an agent at any given time.
	ROUNDS	The number of rounds played before the end of the

		game.
	MORTALITY	Whether agent mortality is enabled or not.
Agent	score	Overall success of an agent participating in the game.
	strategyList	List of strategies available to a given agent.
	memory	Simulated memory of an agent.
	curStrategy	The current strategy being used by an agent.
	age	The virtual age of an agent.
	barVisits	The number of times an agent has visited the bar.
StrategyList	score	The score of a particular strategy within the list.



System Operation Contracts

The operation contracts for the few elaborated use cases are summarized below:

Operation	UC-1: NumAgent – set number of agents participating in the game
Preconditions	- NumAgent field must be a numerical value - number of agents specified > 0
Postconditions	- the NumAgent field is set in game settings

Operation	UC-2: NumRound – set number of rounds in the game
Preconditions	- NumRound field must be a numerical value - number of rounds specified > 0
Postconditions	- the NumRound field is set in game settings
Operation	UC-3: NumStrat – set number of strategies available to each agent
Preconditions	- NumStrat field must be a numerical value - number of strategies specified > 0
Postconditions	- the NumStrat field is set in game settings
Operation	UC-4: MemSize – set number of rounds stored in an agent's memory
Preconditions	- the total number of rounds in the game must be specified - MemSize field must be a numerical value - 0 < memory size specified ≤ NumRound
Postconditions	- the MemSize field is set in game settings
Operation	UC-5: Mortality – toggle agent mortality in the game
Preconditions	- Mortality field must be boolean in nature (true/false)
Postconditions	- the Mortality field is set in game settings
Operation	UC-6: SubAgent – toggle program user participation in game
Preconditions	- SubAgent field must be boolean in nature (true/false)
Postconditions	- the user can play as an agent in the game
Operation	UC-7: PrintOut – print the results of a completed game
Preconditions	- a game has already come to completion and its data resides in memory
Postconditions	- data relating to the completed game is displayed onscreen for the user
Operation	UC-8: Start – finalize game settings and being new game
Preconditions	- valid selections reside in all GUI fields
Postconditions	- a new game begins with specified settings

Mathematical Models

Basic Models:

The El Farol Bar problem and similar minority games are heavily dependent on mathematical and logic computation. Agents participating in the game are completely

reliant on probability derived and calculated from previous rounds when making the decision whether to attend the bar. The model for such a game is outlined below. It is important to note that the agent with the highest score at the end of the last round is the winner, while the score of each individual strategy in used during each round in determining an agent's next move. In this first case, it is assumed that agents do not yet experience memory decay.

- The headcount at the bar can be modeled via the following equation, where A(t) can never be zero in the case of an odd number of agents:

$A(t) = j=1Naj(t)$

If $A(t) < 0$, the majority of the agents stayed home and the bar at round t was enjoyable.

If $A(t) > 0$, the majority of the agents went to the bar and the bar at round t was crowded.

-Payoffs to those agents who made appropriate decisions can be modeled by:

$g_i(t) = -a_i(t) * A(t)$

If $g_i(t) > 0$, agent i won round t.

If $g_i(t) < 0$, agent i lost round t.

Agents winning a round have their individual scores incremented one point appropriately.

-Payoffs to those strategies that ultimately produced a winning outcome can be modeled by:

$\sigma_{ij}(t) = \sigma ijt-1, \text{ if } aij*\&At<0 \; \sigma ijt-1+1, \text{ if } (aij*At)>0$

The score of strategy j of agent i is incremented one point only if it yielded a correct prediction. Otherwise, the score remains the same as in the previous round t-1.

Extensions:

One extension to be implemented in this minority game is memory decay for all agents. Under this model, a memory decay factor α is introduced into the scoring of strategies for each agent. The ultimate affect is that more recently won rounds have a greater impact on which strategy an agent will choose for their next move.

-Payoffs to those strategies that ultimately produced a winning outcome can be modeled by:

$\sigma_{ij}(t) = \alpha * \sigma$ ijt-1, if aij*&At<0 $\alpha * \sigma$ ijt-1+1, if (aij*At)>0

Where $0 < \alpha \leq 1$, and a smaller value of α correspond to a faster decay in memory. The previous score for the strategy j of agent i is first adjusted via α before it is incremented one point if it yielded a correct prediction. Otherwise, the score remains the same as in the previous round t-1 adjusted by the factor α.

User Interface Design

a) Preliminary Design

Figure 8.1: Screen Mock-up of User Interface

Users will have to enter the number of agents, strategies for each agent, the size of the agent's memory, how many rounds, whether they have mortality, whether they wish to play as an agent, and the outputs they wish to display, examples of some shown in figure 8.2. If the user chooses to play, each time they will choose whether to stay or go and press enter. The game will run and display the round and score.

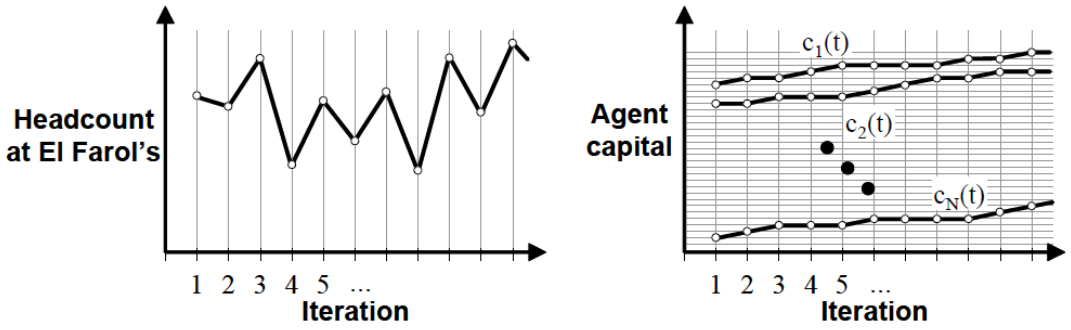


Figure 8.2: Examples of Possible Outputs As Seen in Project Description

b) User Effort Estimation

Run the Minority Game with Computer Agents Example

1. Navigation: total 1 Click, as follows
    - Click "Play" to run the game.

2. Data Entry: total 6 clicks and 8 keystrokes, as follows
    a. Click on the "Number of Agents:" text box
    b. Press the key "5"
    c. Press the "Tab" key to move to the next text field ("Strategies per Agent:")
    d. Press the key "2"
    e. Press the "Tab" key to move to the next text field ("Size of Memory for Each Agent:")
    f. Press the key "3"
    g. Press the "Tab" key to move to the next text field ("Number of Rounds:")
    h. Press the key "1" and "0"
    i. Click the "No" option for the "Mortality:" field
    j. Click the boxes for the "Choice 1", "Choice 3", and "Choice 5" for the "Print Out:" field
    k. Click the "No" option for the "Play as Agent:" field

Run the Minority Game while Playing as an Agent Example

1. Navigation: total 10 clicks, as follows
    a. Click "Go" option for the "Play as an Agent:" field
    b. Click "Play".
    c. Click "Stay" option for the "Play as an Agent:" field
    d. Click "Play".
    e. Click "Stay" option for the "Play as an Agent:" field

f. Click “Play”.
g. Click “Go” option for the “Play as an Agent:” field
h. Click “Play”.
i. Click “Go” option for the “Play as an Agent:” field
j. Click “Play” for the final round.

2. Data Entry: total 6 clicks and 7 keystrokes, as follows
    a. Click on the “Number of Agents:” text box
    b. Press the key “5”
    c. Press the “Tab” key to move to the next text field (“Strategies per Agent:”)
    d. Press the key “2”
    e. Press the “Tab” key to move to the next text field (“Size of Memory for Each Agent:”)
    f. Press the key “3”
    g. Press the “Tab” key to move to the next text field (“Number of Rounds:”)
    h. Press the key “5”
    i. Click the “Yes” option for the “Mortality:” field
    j. Click the boxes for the “Choice 2”, “Choice 4”, and “Choice 6” for the “Print Out:” field
    k. Click the “Yes” option for the “Play as Agent:” field

Plan of Work

After the submission of report #1, the group plans to further develop the software as well as begin work on report 2. A basic demo should be ready 1 week in advance of the demo date. After demo 1, the group will add a number of extensions to the project and begin work on the Electronic Project Archive.

Dates:

Feb 19: Begin work on Second Report / Develop Software
March 10: Finish Second Report
March 22: Finish Demo 1
March 29: Give Demo
April 1: Begin addition of extensions
April 15: Begin Electronic Project Archive
May 1: Finish all software Development
May 2: Give Second Demo
May 4: Finish Electronic Project Archive

References:

Software Engineering by Ivan Marsic

http://www.ssa.gov/oact/STATS/table4c6.html

http://www.blog.joelx.com/odds-chances-of-dying/877/

http://dying.about.com/od/causes/tp/leastdying.htm