B.A.R. G.A.M.E.
Better Arithmetic Reasoning Generated by Acknowledging Minority Experiences

http://www.bargame.info/

Group 7

Michael Chiosi
Andrew Conegliano
Patrick Gray
Christopher Jelesnianski
Marshall Siss
Siva Yedithi

Report 1

February 17, 2012

Breakdown

	Points	Andrew Conegliano	Christopher Jelesnianski	Marshall Siss	Michael Chiosi	Patrick Gray	Siva Yedithi
Project Management	15	40%	60%				
Sec.1: Customer Statement of Requirements	9	10%	40%	10%		40%	
Sec.2: System Requirements	6		10%	40%	50%		
Sec.3: Functional Requirements Specification	30		10%	60%	15%	15%	
Sec.4: User Interface Specs	15	80%				10%	10%
Sec.5: Domain Analysis	20		20%			10%	70%
Sec.6: Plan of Work	5	50%			50%		

Responsibility Allocation

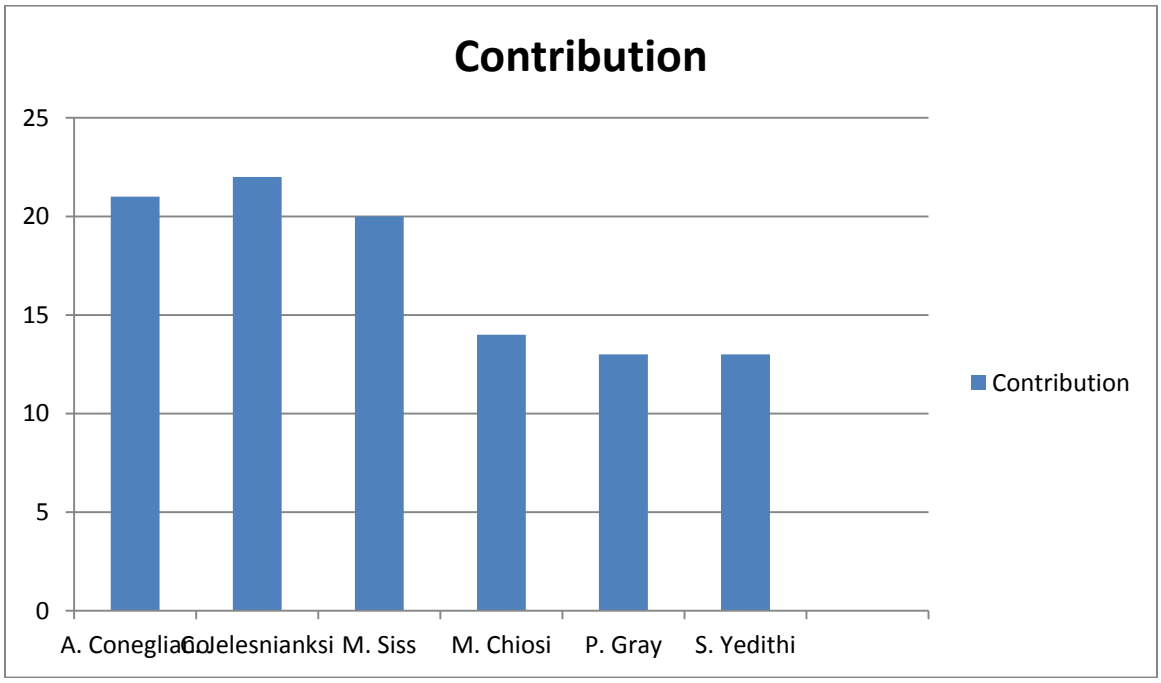


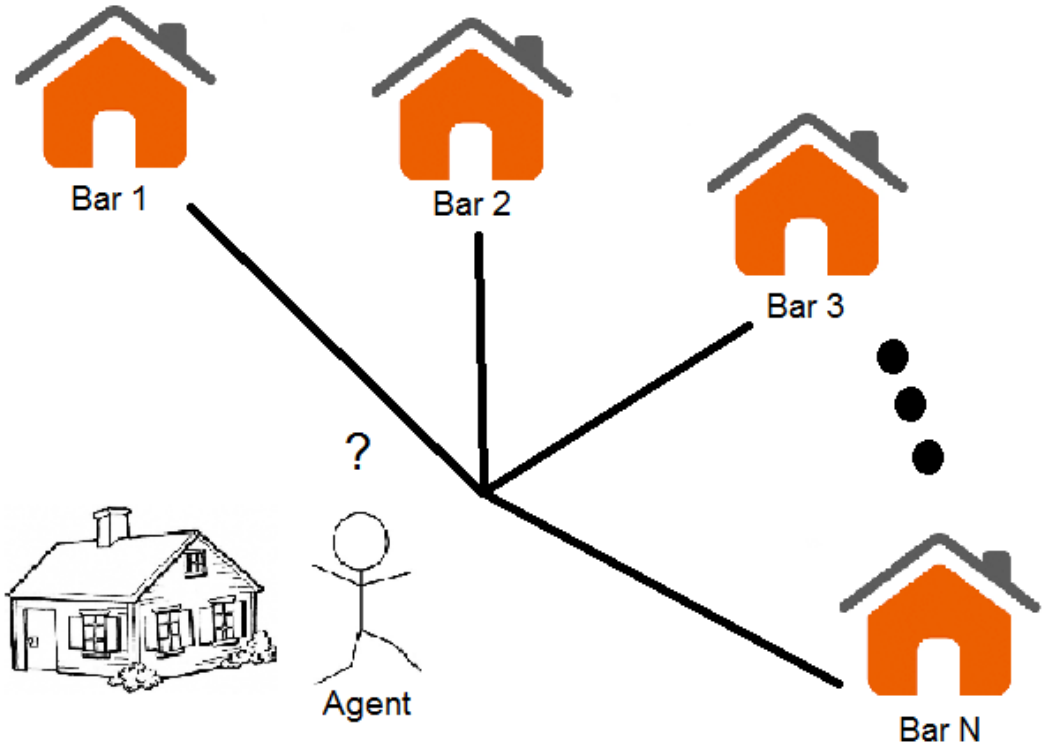
Table of Contents

List of Figures

Customer Statement of Requirements

Problem Statement

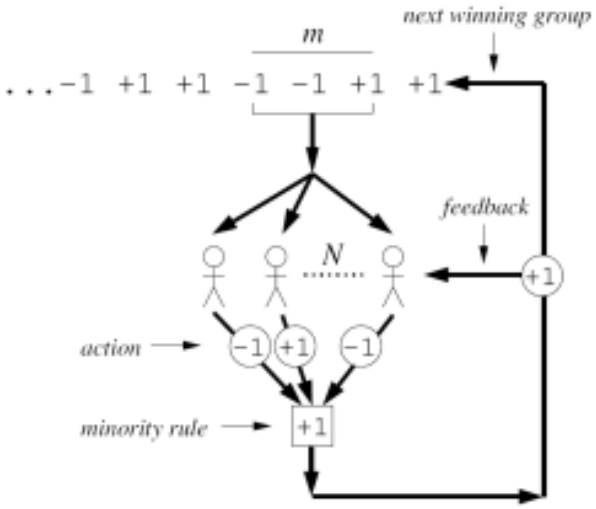
The program designed needs to be able to model a simulation based off the “The El Farol Bar Problem.” This entails simulating multiple agents to decide whether or not to attend a certain bar based on their own past decisions stored in their current Short Term Memory of a preset size N. An agent initially is given S random strategies from the strategy space of possible strategies and stores them in its long-term memory at the beginning of the simulation. After this initialization, depending on the experience the agent has had, the agent will compare his strategies to decide if he will go to a certain bar or stay home. If an agent goes to a certain bar (we are taking into account that there could be multiple destinations implemented to choose from i.e. Bar A, Bar B, Bar C...) when the majority of other agents participating in the simulation also go to the same bar for a given round, then this situation would yield that the agent made the incorrect decision and none of them have a good time because its crowded. Those agents who stayed at home for this round avoided a bad time at the certain bar, therefore making the correct decision and have won that round. On the other hand, if the agent went to the certain bar, while the majority of agents in the population being considered stayed home or did not go to the same bar as him that round, then the agent made the correct choice and won that round. In essence the agents in the minority group always win, while agents who are in the majority group lose. It shows how agents collectively behave in an ideal situation while competing, by means of adaptation, for a scarce resource even without interacting with each other.



Scenario in which multiple Bars are present and an Agent must decide which one to choose

The program should allow for the user to choose the number of agents in the simulation, set the number of certain bars available to go to, and set the capacity type of certain bars whether it is a “recommended maximum occupancy” which the user will also have to specify or based solely on a partial percentage of the total population of agents participating. The simulation should be able to run for multiple rounds, which could be chosen by the user. Furthermore, the user should be able to choose whether to include agent death, birth, and/or marriage. These functions are ones which allow agents special social phenomenon. Death introduces an exit strategy for agents to leave the game, which is equivalent to a person losing interest in the stock market or a

specific venue. Marriage adds a real world aspect to the simulation, emulating an increase in demand for a specific stock, etc.



A simple model of a minority game

Agents should decide whether or not to go to a bar though a combination of different strategies and random chance. Each agent is given gamma strategies upon creation all with a starting value of zero. For the first 5 turns the agent relies of random chance (even chance of all options) after that they rely on strategies. Strategies are possible patterns that may result in a win for the agent. An agent chooses one of their strategies based on random chance skewed by past results. Some of these may include the winner of two of the last 3, always going to the bar, always staying home, loser of two of the last 3, and others. After each round each agent sees which strategies would have gotten them a win and then give those a higher chance of happening in future rounds.

Agent death will occur at a rate consistent with the data provided in the Social Security's Period Life Table[1]. Each agent will keep track of their age and, based on given probabilities, an agent will either live or die. This table gives the probability of

death for ages 0-119, male and female. Because agents in our simulation have no gender, an average of male and female probabilities will be used.

Agent marriage is a possible type of mortality process in which the agents are given a chance alpha to meet at the bar if the are having a good time and a chance beta of having a child every turn that they both win after that. The child then receives the gamma top preforming strategies of its parents. This mirrors the stock world as the parents are like traders who work well together and have decided to train a new stock trader with their best strategies

In addition, the user should be able to set whether the agents may interact with each other. This specifies that all the agents will have access to a group structure which holds all the agents personal decisions, from here individual agents can browse and choose a group strategy based off how well they worked in the past rounds. Group decisions may include going with the majority, going against the majority, or using their personal decision. This mirrors businesses in the stock market world as people do not invest in stocks alone.

The graphical use interface will be created for the simulation program based upon the required fields and settings that a user might need in order to control the simulation that they are trying to create. A welcome/begin screen will greet the user and present a very concise description of what the program does. Below the small description, there will be a list of the developers and other possible copyright or other such information. Seeing as the user will want to get past the welcome screen rather quickly on a regular basis, the welcome screen will only pop up for a small amount of time at the launch of the program.

After the welcome screen fades away, the user will be presented with a screen that contains the variables and fields that they will use to input the important initialization information for each separate simulation. The simulation will only be able to be run if all the required fields have valid values. Valid values will be programmed into the code and if any discrepancies arise, the user will be prompted to change the variables in order to achieve a plausible simulation.

Along with the validation of user defined inputs, all variables will have default values which will provide for a “quick use” scenario. The default settings contain simple numbers that demonstrate the basic ability of the program.

From the input process, the program will progress into the simulation stage. In this stage, the user will be presented with the possible live graphs to display, immediate data, and events that occur in each round of the simulation. This environment will provide a cohesive experience of the agents progressing through the stages of the simulation. By making graphs available during the stages, the program will provide instant feedback as opposed to making the user look back through data at the end of the simulation. This design was chosen because of its obvious transparency/ability to transmit data. While the importance of seeing the information being generated can be debated, the ability to drive the simulation through its stages is integral to the functionality of the program.

Buttons will be placed at the bottom of the simulation environment screen with the ability to step back through stages in the simulation, progress the simulation forward, clear the whole simulation, and any other function deemed necessary or suitable as the program’s build progresses. These means of driving the program are

simplified so that all the work is done by the program, and the user is able to sit back and observe/interpret results.

At any point during the execution of simulations and observation of outputs, the user will be able to access a “Help Menu” that will contain detailed definitions and descriptions of the effects and abilities of each input, output, graph, or GUI item. The information provided in the menu will be simplified to the point that any user will be able to ascertain the possible usage scenarios providing they have knowledge of the El Farol Bar Problem. Possible errors and usage problems related to the input/output/operation of the program will be provided in order to help troubleshoot easily foreseeable situations that any random user might encounter.

After progressing through the stages of the simulation, the user will be greeted with an output/final data screen displaying the graphs replete with all the information generated in the simulation process. These graphs will be selected via drop down menus and the information displayed upon them will be selectable in the same manner. These are important in that they are abstractions of the data generated during the simulation and will increase the user’s ability to understand what it was that went on at each bar, for each agent, or for the whole population. Towards the bottom of the output screen will be methods, implemented with simple labeled buttons, that will enable the user to output the complete, interpreted, or total data to other programs such as Microsoft Excel, Word, and also to plain text. This will be a challenge to implement because of the different methods used by external programs but will be crucial to the use of the program as a tool for end users.

Glossary of Terms

Agent - A virtual “person” that has no gender and decides to either go to the bar or not based on its STM and strategy, with the goal to win.

Minority- the smaller group of population, defined as the lesser group (by number) in the population

Population- the total number of agents present in a simulation

Win - A win is defined by an agent who is present in the minority. For example, when the minority is determined to be at the bar, all the agents there have a good time and therefore win. On the other hand if the minority is determined to be those agents who stayed home, they could have had a bad time at the bar and therefore win since they stayed at home. This is a strict definition for the case when there is only one bar. In the case of multiple bars, a win is defined by an agent who is in the minority group when compared to all other places an agent could have went.

Loss - A loss is the opposite of a win.

Short Term Memory(STM) - Each agent has a static amount of past round data to base it’s next strategy on.

Strategy - An agent’s final decision that gives them the highest percentage to win, based on previous decisions.

Percentage - non-integer representation of probability (ex. 75% chance means something will happen 3 out of every 4 times)

Agent decision-altering events - Any event that alters an Agent's decision. Specific implementations include bar advertisements which increase an Agent's decision to go to the bar and bar fights which decrease an Agent's decision to go the bar.

Self-Optimization - The process of an Agent's adding and dropping strategies based on previous performances of those strategies.

Necessary Variables - The variables that the game needs in order to run. These include the starting number of agents, the number of bars, whether the bar capacity is percent based or number based, the specific bar capacities, morality type, speed, agent interaction, and special events. The user can either set these or they will be set to default values.
Round - A round is defined as one unit of time in which all agents decide their strategy and the results are calculated.

RAW(data) - the simulated data can be exported to plain text (.txt file) for the user.

System Requirements

Enumerated Functional Requirements

Identifier	Priority Weight (Low 1 - 5 High)	Requirement Description
REQ1	5	Agents with minority decision win the current round. Those who choose with majority lose that round. (Still have to decide won/loss conditions upon consideration of multiple bars)
REQ2	4	The number of Agents and Bars should be able to be modified.
REQ3	5	Previous round scores affect Agents future performance/decisions.
REQ4	1	Multiple simulations must be accessible for the user to run in a single session using the application (GUI will not crash on exit).
REQ5	2	Agents decision strategies will differ and change over time due to self-optimization.
REQ6	4	User must be able to stop simulation when they are satisfied with the currently gathered data.
REQ7	2	User should be able to choose which death/birth type is most relevant to their current simulation needs.
REQ8	3	User should be able to specify whether agents will have ability to interact with one another.
REQ9	3	Plots can be chosen to display data of interest to the user, such as a bar graph of an agents success, number of deaths per round, number of births per round, a certain bars attendance over time, and list of the 3 best strategies derived from simulation.
REQ10	1	User should be able to specify whether the simulation will include agent-decision-altering-events. This includes events such as Bar Advertisements and Bar fights.
REQ11	1	The user should be able to save the simulated output data in RAW or formatted versions for the user to use later

Enumerated Non-functional Requirements

Identifier	Priority Weight (Low 1 - 5 High)	Requirement Description
REQ12	3	A “Help Menu” button will give the user access to a more detailed explanation for use of the user input types and help the user input a reasonable value
REQ13	1	The design of our GUI shall be basic to be user friendly, such as having an outlined option section where you select check boxes for which options (multiple bars, death, etc.) you would like to run during the simulation.
REQ14	2	Outlined boxes for input of values will be clearly labeled such as for the number of agents and turns you would like to have in the simulation.
REQ15	5	Before the simulation will begin, the program will identify any errors and explain what is wrong to the user through a pop-up box.
REQ16	2	The program will minimize overhead time by limiting the valid range of input and using the most efficient data structures.
REQ17	2	A familiar and reliable framework is utilized (i.e. Microsoft Visual Studio) that is easy to modify and change on the fly.
REQ18	1	Multiple types of graphs will be easily accessible to the user when simulating.

Functional Requirements Specifications

Stakeholders

The ideal user would be stockbrokers to use for the stock market. The game theory behind the Minority Game[2] problem often appears in financial markets. Agents would represent a buyer, a bar would represent a stock, going to a bar would represent buying a stock, and staying home would represent selling a stock. This would be a very beneficial simulation for a stockbroker in order to determine trends for a market and decide if they want to buy a stock or not. Other options that will be included in the application, such as multiple bars, is a great improvement to the original problem because in the real world there are multiple types of stocks. With this change we can see the trends of multiple stocks in the market due to events. Also, the user has an option of how many bars(stocks) they want to simulate. This gives stockbrokers options to simulate smaller sections of the larger stock market. Birth, death, and marriage for agents can be tweaked to emulate the real world. Random events apply to the stock market and essential emulation tool because stock market crashes occur and companies can one day be profitable, and the next day bankrupt.

Other possible users for this application include, but are not limited to, businesses with a limited amount of a space such as bars, amusement parks, event venues, transportation terminals, and school districts. Bars would be a great example of this because if they reach a certain amount of people, they become overcrowded and a person will have a bad time. This also applies to an amusement park with multiple rides. If a ride becomes too popular the line is usually very long, and most people hate waiting

in a long line for a few minutes of fun. Adjusting this program, it would be possible to determine and extrapolate a rough estimate of the maximum amount people attending an amusement park to have a good time.

Actors and Goals

Actors	Actors Goals	Use Case Name
User	To choose a number of Agents participating in the game, set the number of bars, set the capacity type of bars (number capacity or % of total population), set the turn speed (to show clearly changes in results over time), the death type being utilized by agents, and whether the agents may interact with each other.	InitializeConfiguration (UC-1)
User	User stops simulation after he/she is satisfied with configuration and/or results obtained.	StopGame (UC-3)
User	To start the simulation after configuration is finished.	RunGame (UC-2)
User	To choose which plots to print out.	PrintGraphs (UC-4) can be mid game
System	To have the agents self-optimize their own strategy periodically	RunGame(UC-2)
System	To have the system appropriately generated agent-decision-altering-events.	RunGame (UC-2)
System	The computer running the simulation software	All Use Cases

Use Case Descriptions

Use Case UC-1:	InitializationConfiguration
Related Requirements:	**REQ2, REQ7 REQ8, REQ10, REQ12, REQ13, REQ14, REQ15**
Initiating Actor:	**User**
Actor's goal:	**To set the different settings for the simulation.**
Participating Actors:	**System**
Preconditions:	**Program is running correctly user is looking at GUI, all values set to default**
Postconditions:	**The user has set all of the necessary variables and they are ready to start the game**
Flow of Events for Main Success Scenario:	
→	1. **User** (a) selects the box labeled "Number of Agents" and (b) types in the desired amount
→	2. **User** (a) selects the box labeled "Number of Bars" and (b) types in the desired amount
→	3. **User** (a) looks at the box labeled "Bar Capacity Type" and (b) chooses whether he/she want the bars to be "Percentage Based" or a "Fixed value" by clicking the selected radio button..
→	4. **User** (a) looks in the box labeled "Bar Capacities" and (b) types in the capacities for the number of bars specified above by using numbers separated by commas
→	5. **User** (a) looks at the box labeled "Morality" and (b) chooses a death type that they would like to use from a drop down menu.
→	6. **User** (a) looks at the box labeled "Strategy" and (b) chooses a strategy type that they would like to use (Pure percentage based vs. Decision based where agent chooses from its STM)
→	7. **User** (a) looks at the box labeled "Random Events" and (b) chooses if they would like to include special events

→	8. **User** (a) looks at the box labeled “Agent Interaction” and (b) chooses if they want agent interaction
→	9. **User** (a) looks at slide bar labeled “Speed” and (b) sets a game speed for the simulation
←	10. **System** readies all variable for deployment when user starts game
Flow of Events for Alternate Scenarios:	
→	1a. **User** does not place a value in “Number of Agents” field, therefore the default value is used.
→	1b. **User** places an invalid entry into the box
←	1. **System** notifies **User** upon trying to start program and gives the chance to renter a valid entry
→	2. **User** provides valid entry
→	2a. **User** either does not place a value in number of bars field a defaults value is used
→	2b. **User** places an invalid entry. user is notified upon trying to start program and given chance to reenter information
←	1. **System** notifies **User** upon trying to start program and gives the chance to renter a valid entry
→	2. **User** provides valid entry
→	3a. **User** does not chose a bar capacity type
←	1. **System** sets the Bar Capacity Type for the simulation to Percentage-based.
→	4a. **User** does not place anything in this box therefore the default Bar Capacity is used for all bars.
←	1. **System** sets the Bar Capacity for all bars to the default value
→	4b. **User** does not place enough values in this box to cover the number of bars that the user specified.
←	1. **System** uses all present values for the number of bars whose capacity was specified and sets the Bar Capacity of all the other

	bars to the default value
→	4c. **User** places more numbers in this box than there are bars.
←	1. **System** assigns each bars Bar Capacity to the first numbers given such that it ignore the extraneous values
→	4d. **User** places a .csv file in this field
←	1. **System** reads and assigns all Necessary Variables according to the .csv file
→	5a. **User** does not choose a morality type
←	1. **System** sets that Mortality is not included in the simulation.
→	6a. **User** does not choose a strategy type
←	1. **System** sets the simulation to use the default Strategy type
→	7a. **User** selects the pushbutton “Random Events”
←	1. **System** allows Random Event Occurance to be set by **User**
→	2. **User** provides valid entry
→	7b. **User** does not choose whether there are random events
←	1. **System** sets that Random Events are not included in the simulation.
→	8a. **User** does not chose a starting speed
←	1. **System** sets the simulation to use the default speed.

Use Case 2	**RunGame**
Related Requirements:	REQ1, REQ3, REQ4, REQ5,REQ16, REQ18
Initiation Actor:	User
Actor's Goal:	To start the simulation and let it start simulating based on the inputted criteria
Participation Actors:	System
Preconditions:	User has set up the game as they wish for it to run and all inputted values are valid
Post conditions:	The User is now actively running his simulation and may pause or stop the simulation when the user is satisfied with data
Flow of Events for Main Success Scenario:	
→	1. **User** presses the button "Start Game"
←	2. **System** GUI changes to running mode with the default graphs viewable and the system begins the simulation
→	3. **User** believes that they have enough data and pauses or stops the program (see **UC-3**)
	Includes:: PrintGraphs
	Extends::InitializeConfiguration
Flow of Events for Alternate Scenarios:	
→	3a. **User** decides to change the speed at which the turns are going by modifying the position the scroll
←	1. **System** changes the current speed that the rounds progress
→	3b. **User** decides to request to change the graphs showing (see **UC-4**)

Use Case UC-3:	**StopGame**
Related Requirements:	REQ4, REQ6, REQ11, REQ18,
Initiation Actor:	User
Actor's Goal:	To finish the current simulation
Participation Actors:	System
Preconditions:	User thinks that they have enough data or that the system is stagnant
Post conditions:	Program is stopped and ready to begin a new simulation. Data is also collected by System
Flow of Events for Main Success Scenario:	
→	1. **User** presses the button "Stop Game"
←	2. **System** stops the game and exports generated simulation data to a plain text log file
→	3. **User** may still request to see different graphs through the GUI interface
→	5. **User** presses "start new game" button (button name is replaced "stop game")
	Includes::PrintGraphs
	Extends::StartGame
←	6. **System** directs the GUI to return to the initial setup screen
Flow of Events for Alternate Scenarios:	
→	3a. **User** closes program

Use Case UC-3:	**PrintGraphs**
Related Requirements:	REQ9, REQ11, REQ18
Initiation Actor:	User
Actor's Goal:	During or after a simulation, to select which graphs to print out or view.
Participation Actors:	System
Preconditions:	A simulation has been initialized and has run for a minimum of 1 round and the simulation is currently paused.
Post conditions:	A simulation is still able to be continued to run or has completed and the graphs that the user has requested are displayed by GNUPLOT
Flow of Events for Main Success Scenario:	
→	1. **User** (a) pauses the simulation by sliding the "Speed" slide bar to the left to pause the simulation and (b) selects the graphs that they would like to be displayed
←	2. **System** prints out requested graphs to the **User**
←	3. **System** outputs information gathered so far into a plain text file as well as formatted excel data

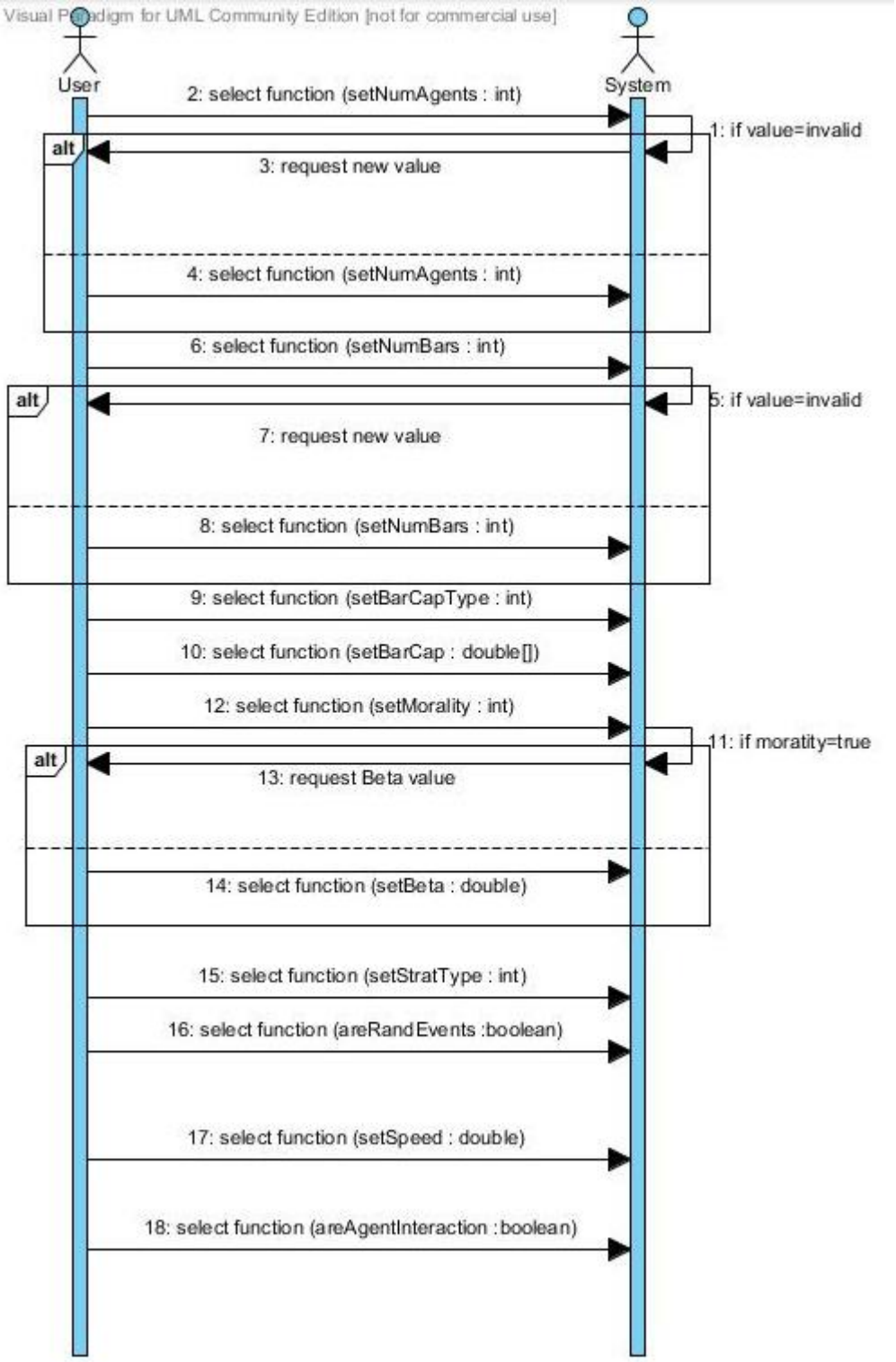
Errors that the GUI will detect and notify the user about:

If a negative value or a non-numerical (i.e. letters of the alphabet) is entered in any field.
If a value is entered that exceeds the bounds of the allowed values in any field
Bounds for each:

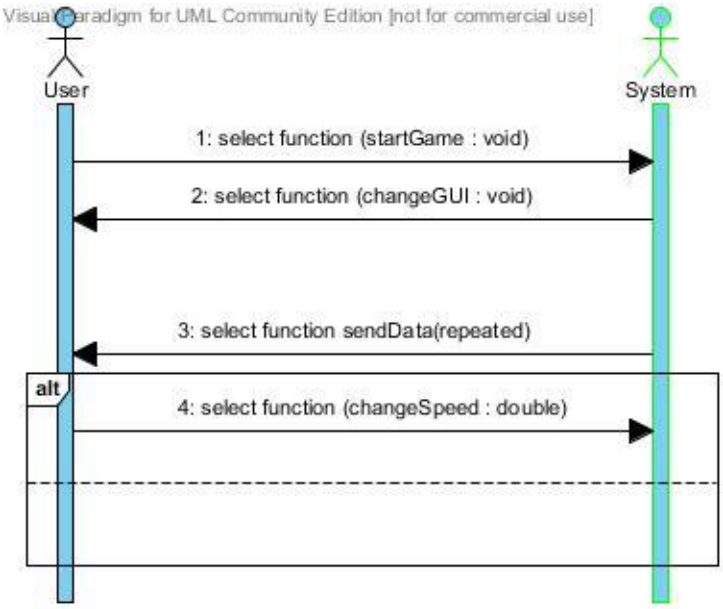
- Number of Agents: 1 to 100,000
- Strategies per Agent: 1 to 4
- Number of Bars: 1 to 10
- Total Rounds: can be in infinite
- Speed: 0 (pause) to 100 percent on a logarithmic scale based on seconds

Use Case Diagram

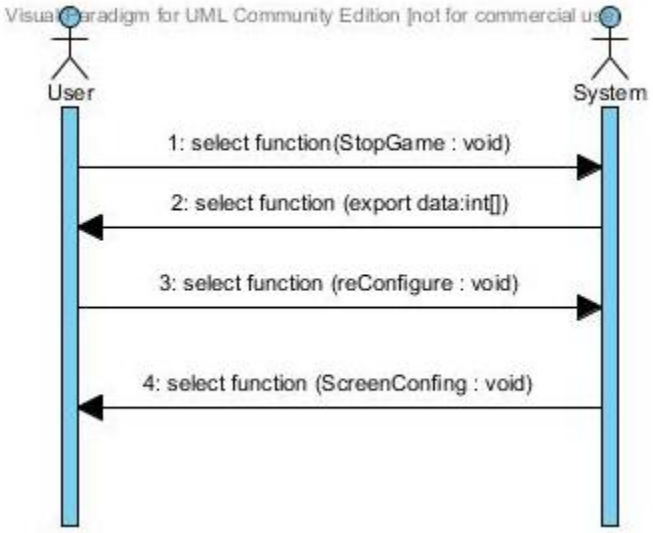
InitializeCinfiguration



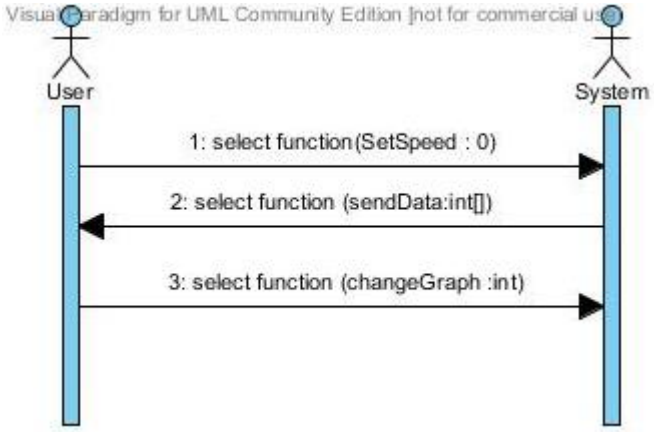
RunGame



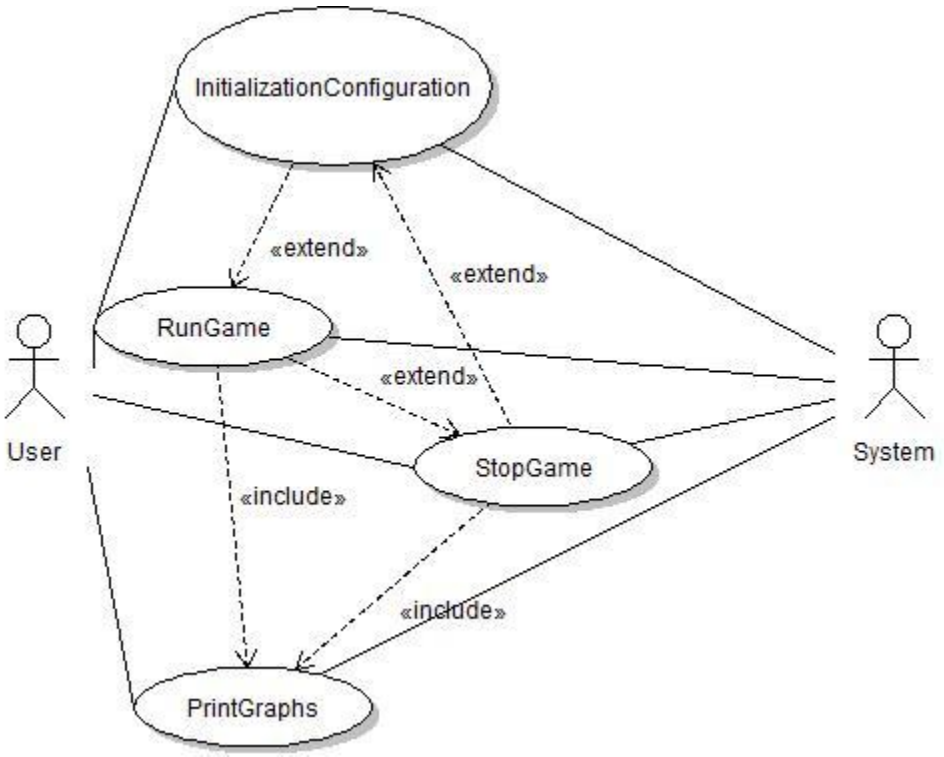
StopGame



PrintGraphs



System Sequence Diagram



Traceability Matrix

Traceability (vs. Requirements)	UC-1 config	UC-2 run game	UC-3 stop game	UC-4 print graph
REQ1: Agents with minority decision win the current round. Those who choose with majority lose that round. (Still have to decide won/loss conditions upon consideration of multiple bars		X		
REQ2: The number of Agents, Bars, and Rounds should be able to be modified.	X			
REQ3: Previous round scores affect Agents future performance/decisions.		X		
REQ4: Multiple simulations must be accessible for the user to run in a single session using the application.		X	X	
REQ5: Agents decision strategies will differ and change over time due to self-optimization.		X		
REQ6: User must be able to stop simulation when they are satisfied with the currently gathered data.			X	
REQ7: User should be able to choose which death/birth type is most relevant to their current simulation needs.	X			
REQ8: User should be able to specify whether agents will have ability to interact with one another.	X			
REQ9: Plots can be chosen to display data of interest to the user, such as a bar graph of an agents success, a list of the 3 best strategies derived from simulation, and				X
REQ10: User should be able to specify whether the simulation will include agent-decision-altering-events. This includes events such as Bar Advertisements and Bar fights.	X			
REQ11: The user should be able to save the simulated output data in RAW or formatted versions for the user to use later				X

User Interface Design

Preliminary Design

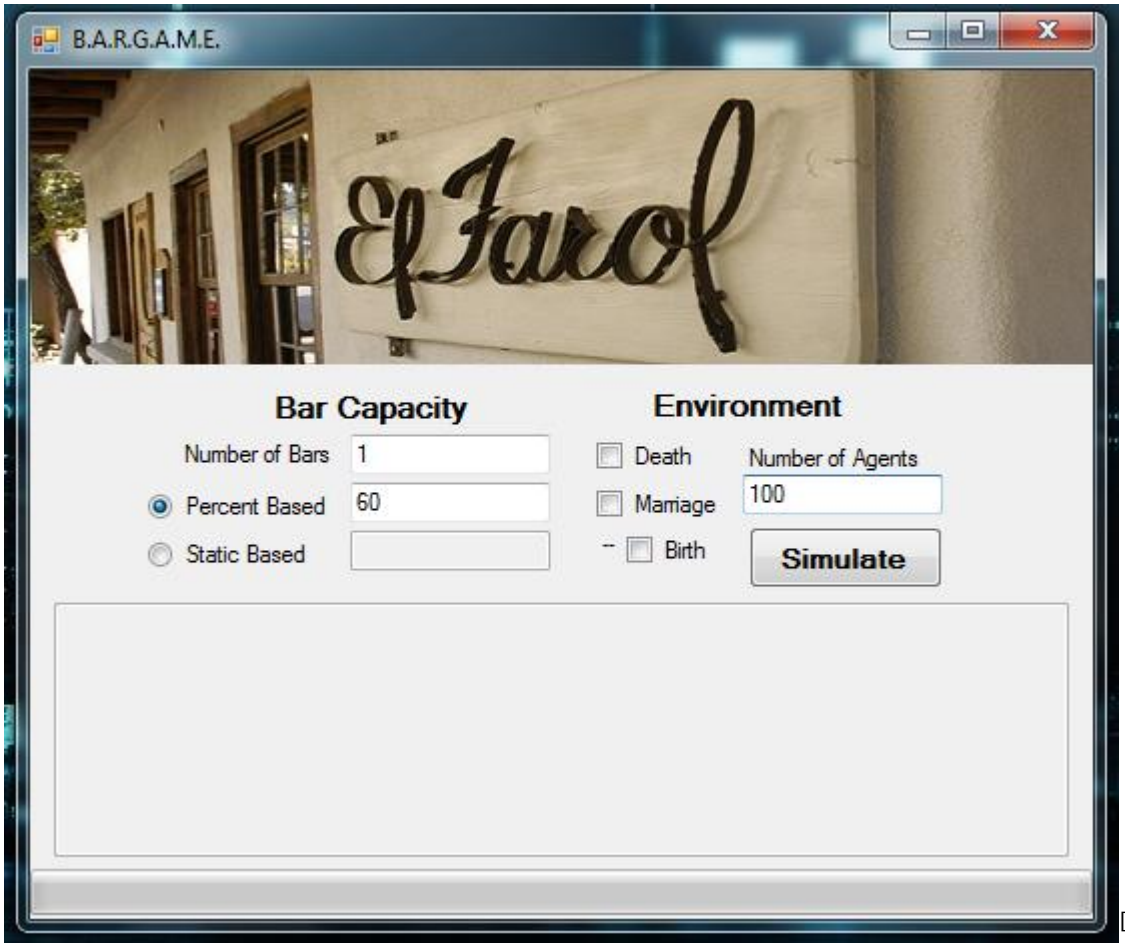
This is our preliminary vision for the GUI. There are two main windows for the GUI: the configuration view and the runtime/data view. The configuration view is where the user inputs the variables desired, and after clicking the “Simulate” button, the runtime/data view will pop up and output two graphs that the user can change to different types at any time during the simulation.

The game options in the configuration view have been categorized to help the user understand what the options do. The left column gives the user options to change the “Bar Capacity”. This includes how many bars that should be simulated and if so the capacity of the bar should be based on either the percent of agents or a constant value. By default, “Percent Based” is checked. Checking either radio button disables input for the other option. The right side gives the user options to change the environmental variables. When box is checked, the condition will be introduced to the simulated game environment . “Number of Agents” sets the population that will be used during the simulation. The box located on the bottom displays the results of the game. Lastly, a progress bar is located at the very bottom, letting the user know the progress of the current simulated game.

For the runtime/data view, two drop downs with the same options are visible. These options change the graph on display, under its appropriate column. A graph can be changed during simulation, and will be updated in real time. At the bottom, there is a slider that changes the speed per round, which will make the graphs and simulation progress faster or slower depending in which direction the slider is put. There is also a

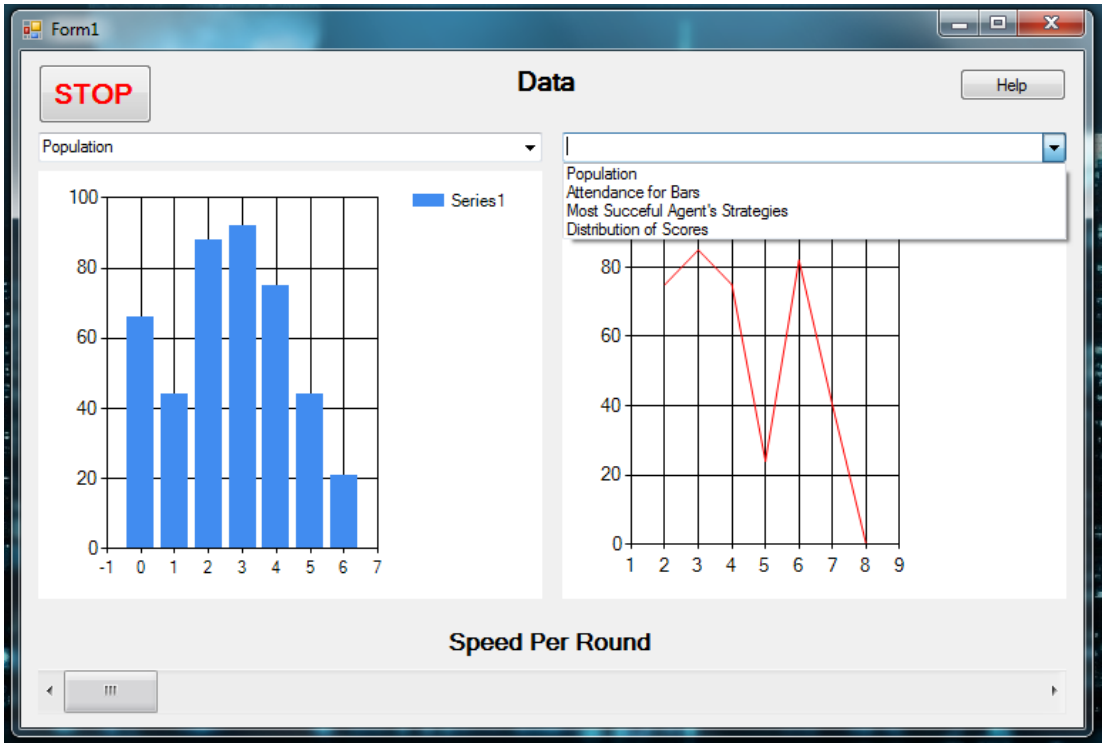
stop button to stop the simulation and a help button that will pop up a window with a more detailed explanations of the given functions.

In future iterations, error checking and more options will be added, with the possibility of reading in a text file to populate required input fields. When a user inputs an invalid number, a pop up will be displayed letting the user know the acceptable range that can be inserted.



[4]

Default GUI Configuration View



Runtime/data view with population graph and drop down of available graphs

User Effort Estimation

Running B.A.R.G.A.M.E. with default values
1. NAVIGATION total 1 mouse click, as follows
a. Click “Simulate”

Running B.A.R.G.A.M.E. with static based bar capacity of 600 agents and 1000 agent population
1. NAVIGATION: total 2 mouse clicks, as follows
Click radio button near “Static Based”

---after completing data entry as shown below--
Click “Simulate:

2. DATA ENTRY: total 2 mouse clicks and 4 key strokes, as follow
Place cursor over the input box next to “Static Based” and click
Press the ‘6’ key and ‘0’ key twice
Place cursor over the input box under “Number of Agents” and click
Press the ‘0’ key to append another zero at the end of “100”

Running B.A.R.G.A.M.E. with 2 bars and death enabled
1. NAVIGATION: total 2 mouse clicks, as follows
Click check box near “Death”

---after completing data entry as shown below--
Click “Simulate:

2. DATA ENTRY: total 2 mouse clicks and 2 key strokes, as follow
Place cursor over the input box next to “Number of Bars” and click
Press the ‘backspace’ key
Press the ‘2’ key

B.A.R.G.A.M.E. with attendance for bars and population graphs
1. NAVIGATION: total 4 mouse clicks, as follows
Click drop down on left column
Click “Attendance for Bars”
Click drop down on right column
Click “Current Population”

Domain Analysis

Domain Model

Concept Definitions:

Responsibility Description	Type	Concept Name
Runs a simulation of the El Farol Bar Game, which updates the current status of the game	D	Game Simulator
Creates and oversees a set of strategies, and makes decisions based on those strategies	D	Agent
Contains information on whether or not an agent should attend the bar in a given situation. Also keeps track of its rate of success	K	Agent
Provides the user with a clear and concise way of setting game parameters and viewing program feedback and results	K	GUIDisplay
Keeps track of the number of people at each bar	K	Town
Asking to see who is going to the bar	D	Town
Outputs data to RAW or formatted version to user for later user	D	Archiver

Association Definitions:

Concept Pair	Association Description	Association Name
GUIDisplay ↔ Simulation	The GUIDisplay sends the user input to the Simulation to set initial parameters	Provides Input
Simulation ↔ GUIDisplay	Sends the current data to GUIDisplay	Updates
Agent ↔ Town	Town polls bars and agents for information and records who is going to a certain bar	agent poll

Town ↔ Simulation	Town sends data to Simulation to be compared against known data, to calculate outcome	calculate outcome
Simulation↔Archiver	Simulation gives current collected data to Archiver to be outputted to user	provides output

Attribute Definitions:

Concept	**Attributes**	**Attribute Description**
Simulation	MortalityType	contains enumerated type of mortality model being using
	Speed	is the current speed at which simulation progresses
	currRoundData	contains current information about the past turns
Town	currentTurn	keeps track of the current round of the game
	numBars	the number of bars available (user defined)
	numDeaths	the number of deaths that occur a specific round
Agent	age	Each agent is responsible for keeping track of their own age
	strategies	contains an agents current personal strategies
	g_strategies	an agents strategy relating to group interaction, also each entry has ranking based on performance
	wins	the number of current wins so far
	deathDate	the age at which an agent will expire
GUIDisplay	graph_A	the graph to be displayed in the first graph box
	graph_B	the graph to be displayed in the second graph box
	simRunning	boolean determining whether the game is running

Traceability Matrix

	Priority Weight (1 - 5)	**Domain Concepts**			
Domain Concepts (vs. Use Cases)		GUIDisplay	Game Simulator	Agent	Town
UC1 (InitializeConfiguration)	4	X			
UC2 (RunGame)	5	X	X	X	X
UC3 (StopGame)	1	X	X		
UC4 (PrintGraphs)	3	X		X	

Operations Contract

Name	runTurn
Precondition	The simulation has been initialized The simulation is running
Postcondition	Each users age is incremented and age is matched against death date A winning group has been determined Strategy rankings and wins have been updated numDeaths is determined

Name	changeGraphs
Precondition	The simulation is running or has concluded
Postcondition	User specified graph (graph_A, graph_B) has replaced current graph displayed

Name	startGame
Precondition	All necessary variables that have been inputted are valid User is ready to begin simulation
Postcondition	The simulation is running Necessary variables are set to user specified values simRunning has been changed to true

Name	stopGame
Precondition	The simulation is running User is ready to end simulation
Postcondition	The simulation is paused/stopped simRunning has been changed to false

Plan of Work

There will be 2 demos over the course of this project. Our first demo will be basic and functional, the objective for this demo is to have working agents, graphs, and a bar. The GUI will allow the user to set the number of agents and choose if winning is percentage based or a numerically value. Agents will have multiple user options, such as age, death, and strategies.. For the final demo we plan to improve features from the first one, but also include more user options. A large difference from the first demo will be that the final demo will give the user the options of having multiple bars. Other improvements will be more graphs and additional user options such as birth and marriage for agents. Also, agents will be more advance by having more complex strategies that will determine if they attend the bar or not. Finally, we will add a random event option that will occur in the town.

Breakdown of Future Responsibilities:

Michael Chiosi: will be responsible for coming up with the Interaction Diagram specifics and help with System Architecture and Design.

Andrew Conegliano: Andrew will continue to work on Graphical User Interfaces and completing the mapping to of values subsystems to the simulation. He will also complete the User Interface Design and Implementation

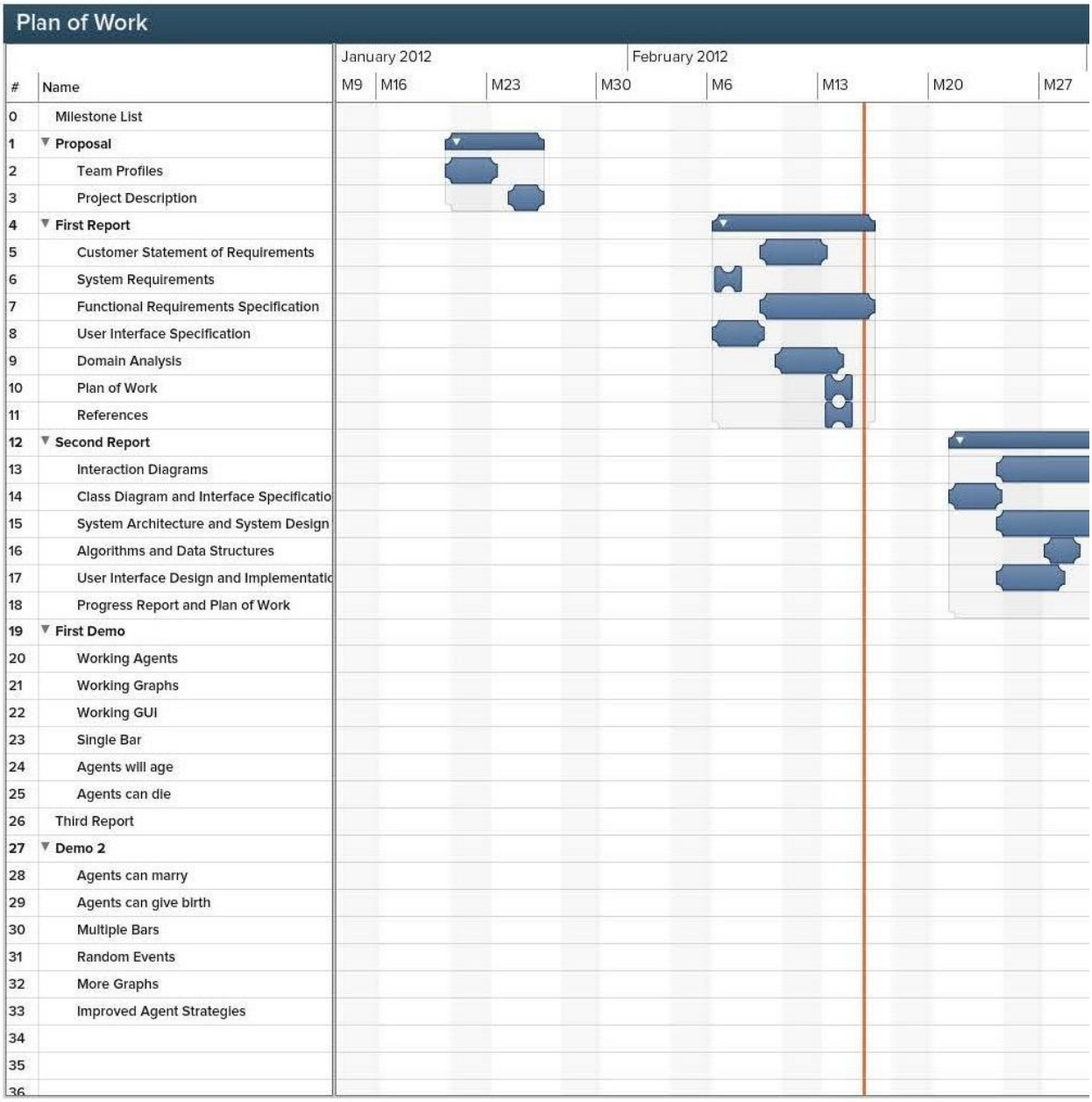
Patrick Gray: will be the main person responsible for System Architecture and Design, along with helping with the writing of the report

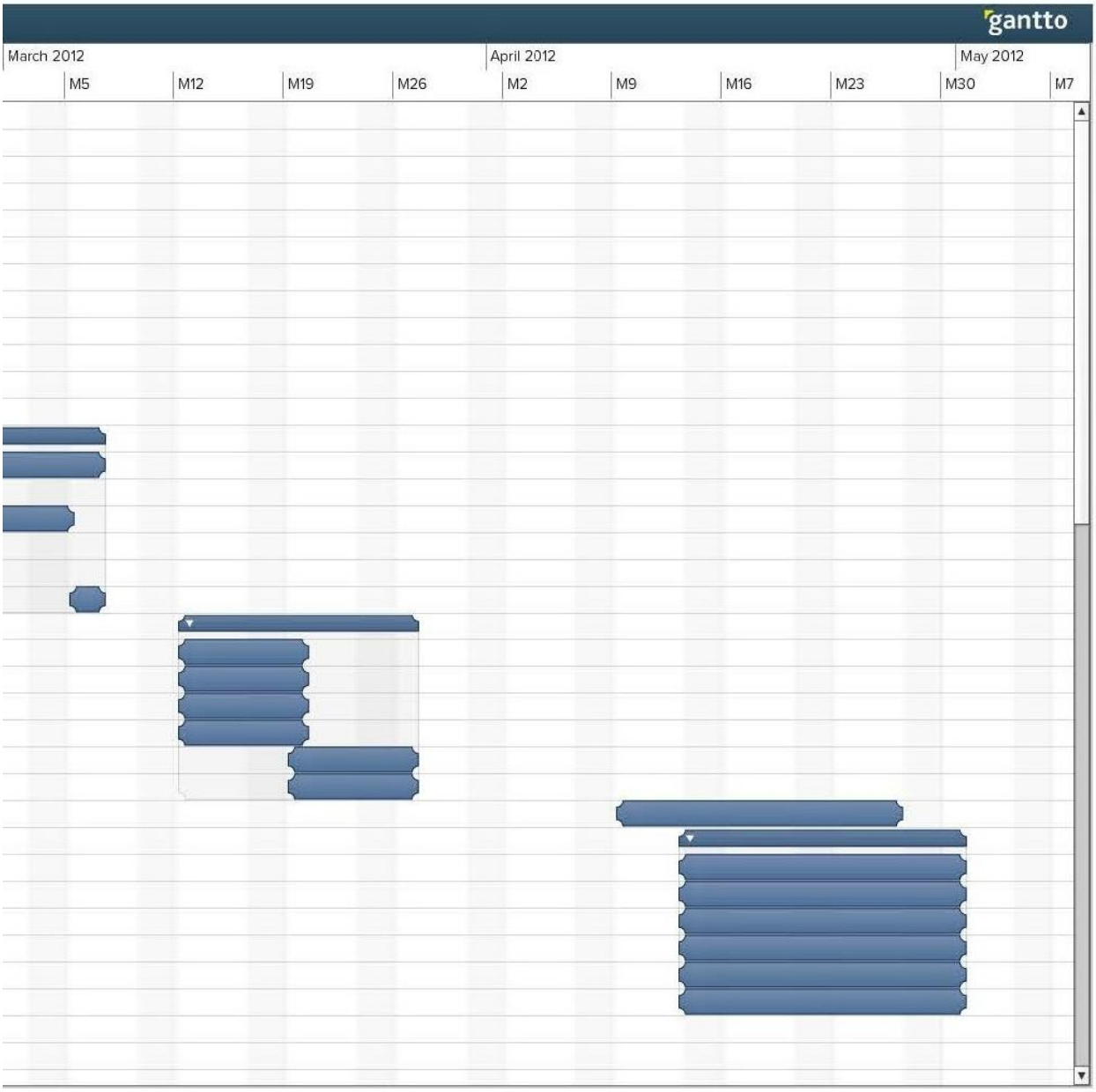
Christopher Jelesnianski: will help with concept implementation of code and will perform the majority of testing as well as documentation and editing of specifications in header files to clearly document code working alongside with Andrew, Mike, Patrick and Siva.

Marshall Siss: will be in charge of coming up with Algorithms, data structures and UML diagrams to provide continuity between other members of the team and their programs to facilitate the creation of a cohesive and streamlined application.

Siva Yedithi: will be interpreting the generated UML diagrams to come up with class diagrams and interface specification. His responsibility will be to strategically perform design analysis in choosing good data types and operation signatures for easy understanding

(URL for a larger version of the Gantt graph below)
http://bargame.info/gantt_pw.jpg


Plan of Work
January 2012
February 2012
M9
M16
M23
M30
M6
M13
M20
M27
Name
0 Milestone List
1 Proposal
2 Team Profiles
3 Project Description
4 First Report
5 Customer Statement of Requirements
6 System Requirements
7 Functional Requirements Specification
8 User Interface Specification
9 Domain Analysis
10 Plan of Work
11 References
12 Second Report
13 Interaction Diagrams
14 Class Diagram and Interface Specificatio
15 System Architecture and System Design
16 Algorithms and Data Structures
17 User Interface Design and Implementatic
18 Progress Report and Plan of Work
19 First Demo
20 Working Agents
21 Working Graphs
22 Working GUI
23 Single Bar
24 Agents will age
25 Agents can die
26 Third Report
27 Demo 2
28 Agents can marry
29 Agents can give birth
30 Multiple Bars
31 Random Events
32 More Graphs
33 Improved Agent Strategies
34
35
36


gantto
March 2012
April 2012
May 2012
M5
M12
M19
M26
M2
M9
M16
M23
M30
M7

References

1. Period Life Table. *Social Security Online.* 2/16/12
http://www.ssa.gov/oact/STATS/table4c6.html

2. O. Brandouy, Stock markets as Minority Games: cognitive heterogeneity and equilibrium emergence, Physica A: Statistical Mechanics and its Applications, Volume 349, Issues 1–2, 1 April 2005, Pages 302-328, ISSN 0378-4371, 10.1016/j.physa.2004.10.019.
(http://www.sciencedirect.com/science/article/pii/S037843710401355X)

3. FURPS. *Wikipedia.* Retrieved February 10, 2012
http://en.wikipedia.org/wiki/FURPS

4. El Farol Picture (used in GUI)
http://www.thesantafevip.com/activities/santa-fe-dining/el-farol-on-canyon-road/

Software Engineering, by Ivan Marsic

Project #3, group #4, Spring 2011 (Last year's project)
http://www.ece.rutgers.edu/~marsic/books/SE/projects/MinorityGame/2011-g4-report3.pdf

Design Decisions

It was asked whether we should only use the male or female statistics from the Period Life table provided by United States Social Security Administration. Since we assumed the agents have no gender, we decided to average the death probability between a male and female for a certain age. Using only one genders statistics would make the application biased.

Another important design decision that we chose was to make the GUI with Microsoft Visual Studio 2010 versus other software. Not only does Visual Studio provide a reliable framework, but it is very easy to modify and change on the fly. In addition, because the majority of the group is most familiar with C++, Visual Studio best met our requirements for a familiar programming platform.