Software Engineering

Minority Game

Report 1

Group 12

Academic Year 2014-2015

Rutgers University

February 22, 2015

Members: Matthew Chatten, Ameer Fiqri Barahim, Vicent Vindel Dura, Alexander Hill, David Lazaar, Orielle Joy Yu.

Table of Contents

Individual Contributions

REPORT 1 Distribution	**Team Member**					
	Matthew Chatten	Ameer Fiqri Barahim	Vincent Vindel Dura	Alexander Hill	David Lazaar	Orielle Joy Yu
Project Management (10 points)	*75%*	*5%*	*5%*	*5%*	*5%*	*5%*
Customer Statement of Requirements (9 points)	*75%*	*25%*				
System Requirements (6 points)				*33%*	*33%*	*33%*
Functional Requirements Specification (30 points)	*20%*	*20%*	*20%*	*20%*		*20%*
User Interface Specs (15 points)					*100%*	
Domain Analysis (25 points)	*20%*	*20%*	*20%*	*20%*		*20%*
Plan of Work (5 points)					*100%*	

Customer Statement of Requirements

Problem Statement

The goal of this project is to better understand the El Farol Bar problem(1) through the analysis of a Minority Game(2). The El Farol Bar problem has many variants, but the core idea remains the same: how to make a decision that is dependent on the decisions of others. In the original formulation, the question is whether or not to go to a bar. Going to the bar is a good decision only if most people decide it is a bad decision, and vice versa. There is no perfect answer to such a question. The Minority Game is an expansion of the problem in which a set of agents must use and test strategies for solving the El Farol Bar problem. If you choose the minority option, you win the round. Strategies are tested with the knowledge of how previous rounds turned out. By running a Minority Game, you can understand how independent agents will choose to answer the El Farol Bar problem, even if a “right” answer is impossible.

This is the real value of Minority Game. The El Farol Bar problem is game theory, and the game helps translate that into practical information. In the real world, you will always have imperfect information on other people’s decisions. In any case where there is competition for a limited resource, or where being in the minority is advantageous, accurately predicting other people’s actions and making the least popular choice is a valuable skill. This is a question with many real life applications beyond the original idea of trying to avoid a bar if it is too crowded. Examples include building a company in an underserved market, choosing a career in a field that needs more workers, finding a vacation spot that isn’t overcrowded, or picking a stock that’s undervalued. A good understanding of how independent actors act in a Minority Game can thus be used to better understand how to act most effectively in real life situations.

With this in mind, our desired product is a Minority Game simulation designed to model agents in a competitive fishing environment. In order to gain the maximum value from such a simulation, it must be customizable, complex, and able to deliver large amounts of relevant data. There is no better way to do this than through software. Only software can make customization easy, hide the complexity of the simulation, and deliver useful and readable data with ease.

The most basic Minority Game involves only a set of agents and a single bar, but real life situations are much more complex. This product should expand on the Minority Game in two main directions. First, there should be multiple fishing locations that the agents can choose from. Second, there should be multiple types of fish, each with their own fishing locations, that the agents can choose from. Ideally, an agent wants to choose both the least crowded fishing spot and the least popular fish species.

The goal of this Fishing Minority Game is to understand how agents should act in such a real life scenario. To facilitate this understanding, the software should have three key features. First, it should have a range of options that will make many unique simulation configurations possible. Choosing different numbers of agents or decreasing the amount of fish, for example, will provide more opportunities for insight than a single static scenario. Second, the simulation should show how the individual agents act over time and what strategies they use so a clear pattern of action and response can be discerned. Third and finally, the simulation should collect all relevant data and present it once complete for analysis.

With the basic framework of the Fishing Minority Game established, it is important that the details be explained as well. Fishing does not only depend on the number of people going fishing, it is also affected by the skills and experience of the fisherman. Factors such as weather, types of fish, and duration of fishing also affects outcomes. The simulator should take into account all of these factors.

Different initial variables can be set for the simulation: number of fishermen, frequency of communication between fisherman, skill and experience rank, amount of each type of fish, fishing spots, fishing duration, and weather patterns. There can be a limited number of fisherman per spot. Fishermen sometimes would talk to other fisherman about good spot or current information on a certain spot affecting fishing decisions. Experience and skill can be ranked from 1 - 5, with 5 as the highest rank. The higher the rank, the more often a fisherman would want to fish. There are also a limited amount fishing spots per area. A fisherman can set how long they want to fish in hours a day. The weather can be measured in degrees of temperature. More fisherman will fish if the weather is good but fish would only appear on certain temperatures.
After all these inputs, the simulation will run. Simulation can be “watched” rather than running and displaying the results automatically. When the simulation is done, the simulator should show graphs that can be easily understood by the user and thus, help the user to ease up the decision making. There would be a graph for number of fisherman versus the time of day (could be extended to a week). Other graphs should display the percentage of success versus factors such as the number of fisherman, weather, experience and skill.

Of course, the key to all of this is the strategies that the fishermen will use to choose fishing locations and types of fish to catch. These strategies should be based on the information available to the fishermen: how other actors have acted in the past, and the current conditions. In real life, that is the type of information available, and so it should be here. To make this simulation feasible, however, the strategies will be fairly straightforward. An example strategy would be going to a certain fishing spot if the current conditions are known to be good and it has been overpopulated more than once previously. Another could be choosing a fishing spot if it was overpopulated two rounds ago and underpopulated last round. Strategy development, implementation, and evaluation should follow these steps:

1. (Initialization) Agents randomly choose a set of strategies and a specific fish species to begin.
2. Each agent looks at their strategies and chooses the one with the highest success score, i.e. the one that would have made a good decision the highest number of times.
3. Each agent receives the outcome of the round and updates its strategies based on the new data.
4. Each agent looks at its strategies' success as a whole and decides whether or not to switch fish species.
5. Each agent returns to step 2, unless the simulation has ended.

With this sort of strategy implementation, the simulation will be sure to cover the two most important aspects of this Minority Game variation. First, it will allow for the testing and evaluation of strategies in direct competition with each other. The point of this product is to understand how different agents act, and the key to that is seeing which strategies work, which don't, and how strategy effectiveness changes over time. By having every strategy for every actor constantly evaluated, this goal can be achieved. Second, it will allow for a better understanding of the costs and benefits of switching fish species entirely. An agent who switches has a chance to be more successful, but will be taking a large risk since their strategy success scores will be tuned to a different fish species. This simulation should allow better insight into the tradeoffs involved in such a risky decision.

The final challenge in building this Minority Game fishing simulation is making it into a customer-friendly product. The simulation won't give any help understanding the El Farol Bar problem if it is difficult to use, or if the data is hard to read. With that in mind, there are a few basic expectations for this software project. There should be an easy to use graphic user interface for the simulation. This will allow for easy understanding and setting of simulation options, simple representation of the simulation itself, and an interface for the collection and presentation of simulation results. Second, the software should be self-contained. It should be a downloadable product that can be installed and run on its own. It only needs to be used by a single person at a time, and since the simulation only depends on initial parameters it shouldn't need any outside connections or products to function.

The construction of a Fishing Minority Game will not be an easy task. The problem itself, a question of game theory and complex interactions between independent agents, is only a model of the even more intricate machinations of real life. However, with this simulation we hope that we can come a little closer to understanding.

Glossary of Terms

- **Skill and experience rank** – number will be used to represent the skill and experience of a person in fishing. Experience and skill can be ranked from 1 - 5, with 5 as the highest rank.
- **Weather patterns** – degrees of temperature will be used to represent the weather patterns.
- **Strategy** – strategy that is used by the fisherman that will help in making the decision to choose fishing locations and type of fish to catch.
- **Simulation settings** – section for the user to initialize all the initial variables for the simulation.
- **Simulation log** – displaying the previous record of simulations.
- **Save** – allow the user to save the simulation settings and data into a file.
- **Load** – open previously saved file that contain all the simulation settings and data.
- **Reset** – change all the simulation settings to default.

System Requirements

Enumerated Functional Requirements

Identifier	Priority Weight (Low1-High5)	Requirement Description
REQ-1	4	The simulation shall incorporate initial values for number of fisherman, fish population size, and number of fishing locations.
REQ-2	1	The simulation will optionally account for weather by causing less people to want to fish during very hot or cold conditions.
REQ-3	1	The simulation will account for the amount of experience each fisherman by causing more experienced members them to want to go fishing more often and have a following.
REQ-4	2	The simulation will increase fishermen experience when they fish in unpopular areas or for unpopular types.
REQ-5	4	The simulation will save each actors decisions in a log file.
REQ-6	2	The simulation will account for reduction in fish due to overfishing by decreasing the amount of fishermen the area supports.
REQ-7	5	The simulation results will show the number of fishermen per area over time and the average success of fishermen as a whole.
REQ-8	4	The simulation will allow each fisherman to remember his past choices and the experienced results.
REQ-9	3	The simulation will provide different strategy options for fishermen to use when deciding when and where to fish.

Enumerated Nonfunctional Requirements

The FURPS model stands for functionality, usability, reliability, performance, and supportability. Each of these would be elaborated below for this particular software.

· Functionality - This software allows users to set parameters for different factors. User inputs parameters and the output would be graphs showing results of software simulation.

· Usability – The software should be easy to learn and navigate. The users would only need to input parameters and/or choose factors. User interface design should be well planned, organized and consistent. There would be documentations and demos of how to use the program.

· Reliability – The probability of system failures should be low. Errors that are caused by user inputs should be addressed clearly in order to have it corrected immediately. Saved files should have backup copies for recoveries in case of saving/loading failure. The resulting data of simulations should be accurate.

· Performance – The software should have high but efficient performance. Since the data simulation would be running on real time, the speed should be slow enough to for the user to follow but fast enough so that the user would not have a long waiting period. Power, time, and space consumption should be minimal.

· Supportability – The software should be easy to understand by the user and the programmers. There would be clear documentations of all the product's required files. This would help product maintenance and repair. Technical support by email should also be available.

Identifier	Priority Weight (Low1-High5)	Requirement Description
REQ-10	5	Can choose which factors to be included and not included in simulations.
REQ-11	1	Be able to create an account with username and email address.
REQ-12	4	The user interface should be easy to navigate and easy to learn. It should be user friendly.
REQ-13	3	Data results should be backed up in case of errors or system failures.
REQ-14	3	If an error is caused tell user what it is and provide guide on how to fix it.
REQ-15	2	Software should be easy to install and run.
REQ-16	2	Be able to send email to creators for technical support.

On-Screen Appearance Requirements

The Fisher's Simulator application has a simple design. Simulation settings are found on the upper left side of the window. They include options like the amount of fishers, number of fishing locations, number of fish types, fish population, simulation runtime, and weather. Checkboxes or sliders are used where appropriate. This method of data input provides a simpler, more guided user experience compared to fillable forms, which are too easily filled with invalid data by new users.

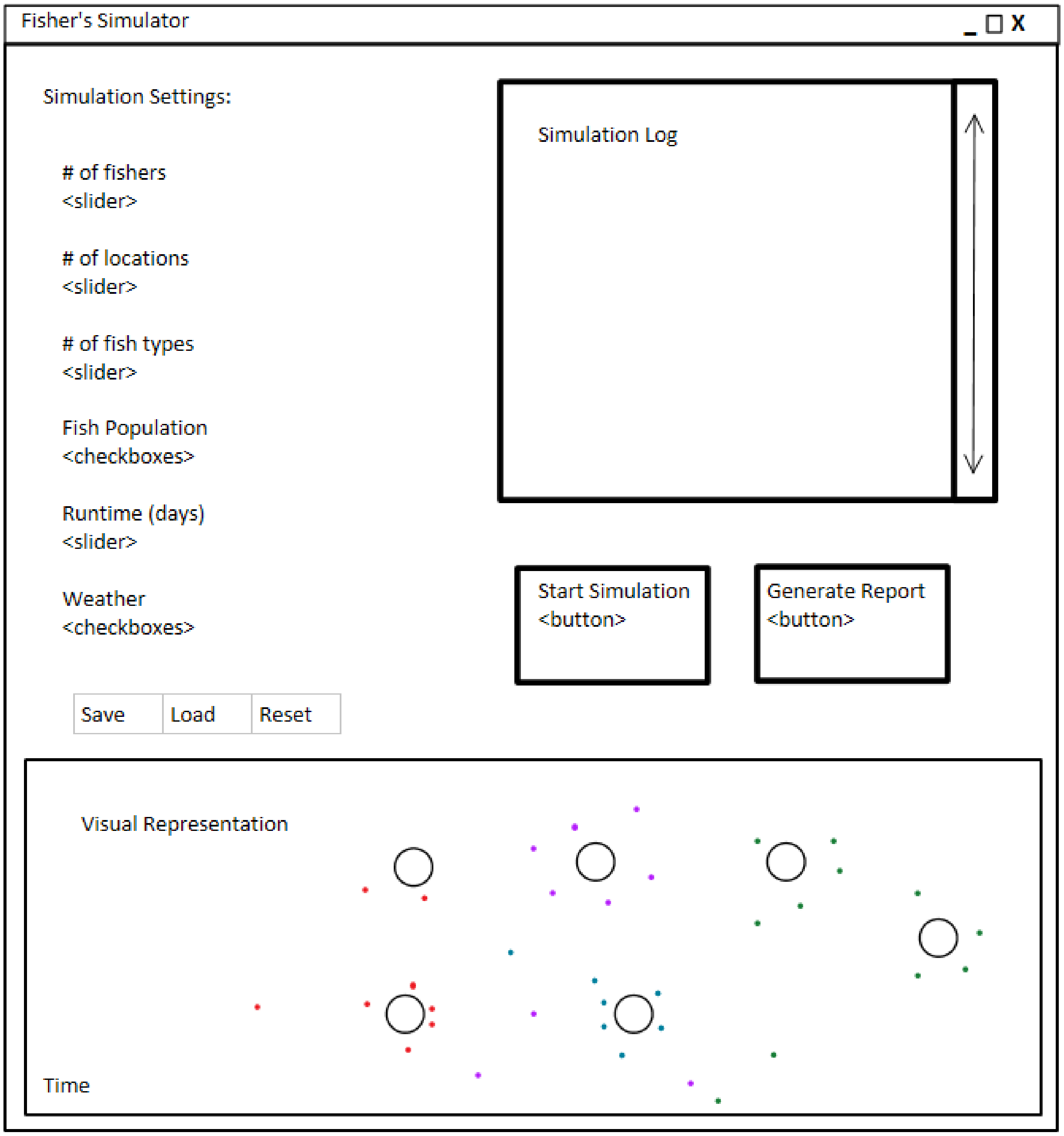
Directly below the settings area are save, load, and reset buttons. This will allow the user to save or load configuration files, or reset to an interesting default simulation.

The simulation log will dominate the right side of the application. It will have a text and number based description of the simulation, both during runtime and when the simulation ends.

Below the log will be large start and generate report buttons. The start button allows the user to start the simulation. It may turn into a pause button while the simulation is ongoing. The generate reports button will present the results from the simulation in a graph based format that can be then saved as a standard image file.

At the bottom of the window will be a graphical representation of the simulation as it runs.

Main Screen Mockup:



Function Requirements Specification

Stakeholders

This software is designed for fishermen who are looking to find the location that is least crowded and to maximize their chance of success of catching a certain species of fish. A fisherman would want to fish at the least crowded fishing spot and learn the least popular fish species in order to have a higher chance of enjoyment and success. Results are in graph form in order to make it easier to read and understand. The software can also be used to keep track of experience, strategies and changing factors in order to influence the fisherman's decision in a positive way.

Actors and Goals

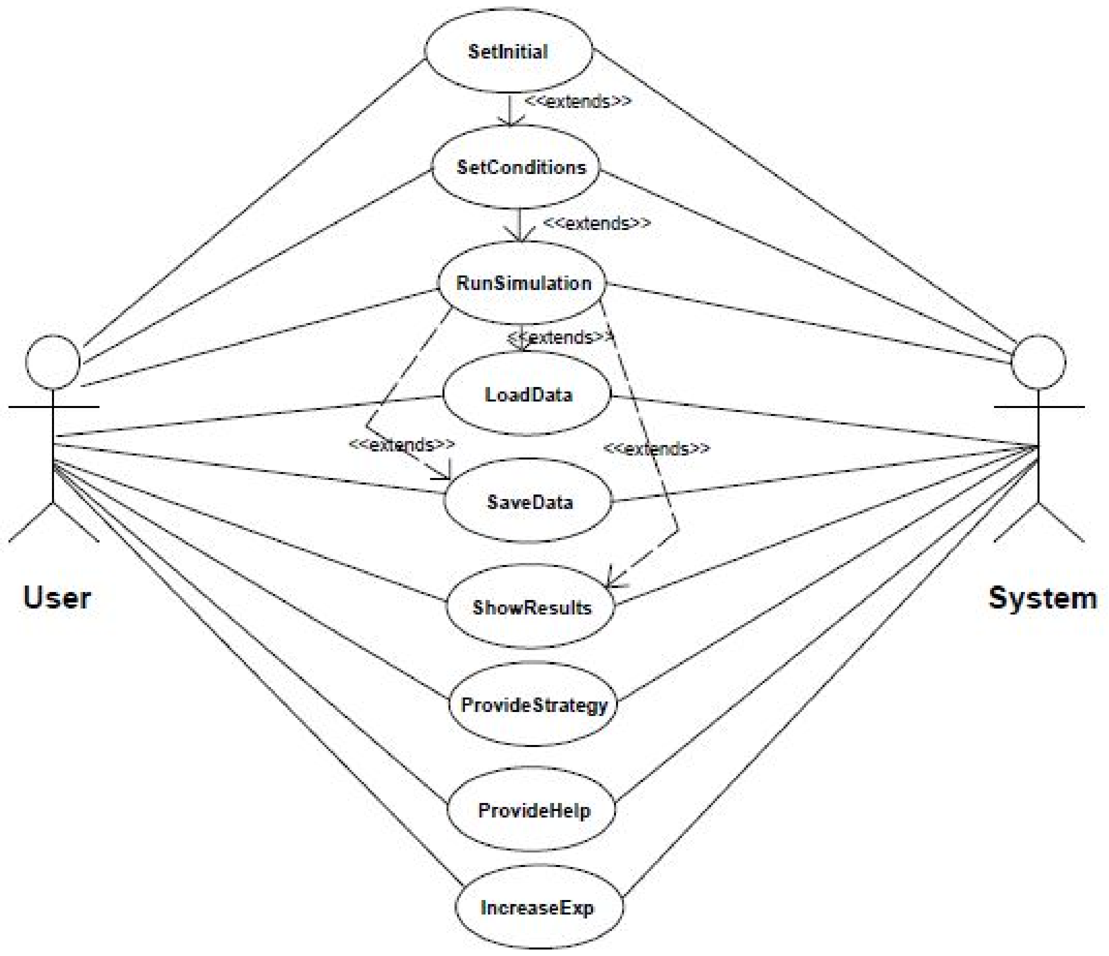
Actor	**Actor's Goal**
User	Set initial values of conditions
User	Choose conditions to include in simulation
User /System	Run simulation
User	Save results of simulation
User	Load results of simulation
System	Show graphs of simulation
System	Provide strategies to user
System	Increase experience points
System/User	Provide technical help

Use Cases

Casual Description

Identifier	Use Case Name	Casual Description	Related Requirements
UC-1	SetInitial	The user is going to set specific values to each of the selected conditions.	REQ-1
UC-2	SetConditions	The user can choose which conditions would affect the simulation and results.	REQ-2, REQ-3, REQ-6, REQ-10
UC-3	RunSimulation	The user would tell the system to start the simulation. The system would respond by running the simulation that the user can watch.	REQ-7
UC-4	SaveData	The user can save current simulation data. Saved data will be backed up.	REQ-5,REQ-8,REQ-13
UC-5	LoadData	The user can load previous data.	REQ-8
UC-6	ShowResults	The system would show the finished graphs of the simulation.	REQ-7
UC-7	ProvideStrategy	The system can provide different strategy options for user.	REQ-9
UC-8	IncreaseExp	The system would take account current and past data to increase experience points of user when necessary.	REQ-4
UC-9	ProvideHelp	When there an error the system can provide help or user can email for help.	REQ-14,REQ-16

Use Case Diagram



Traceability Matrix

Req	**PW**	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9
REQ-1	4	X								
REQ-2	1		X							
REQ-3	1		X							
REQ-4	2								X	
REQ-5	4				X					
REQ-6	2		X							
REQ-7	5			X			X			
REQ-8	4				X	X				
REQ-9	3							X		
REQ-10	5		X							
REQ-11	1									
REQ-12	4	X	X				X			
REQ-13	3				X					
REQ-14	3									X
REQ-15	2									
REQ-16	2									X
Max PW		4	5	5	4	4	5	3	2	3
Total PW		8	13	5	11	4	9	3	2	5

Fully Dressed Description

Use Case Detailed Descriptions

Use Case UC-2: SetConditions
Related Requirements: Req-2, Req-3, Req-6, Req-10
Initiating Actor: User
Actor’s Goal: Set up conditions for a simulation.
Participating Actors: System
Preconditions: None
Postconditions: Simulation is ready to run
Flow of Events for Main Success Scenario: 1. User opens menu for selecting conditions. 2. User selects options on menu. 3. User finishes selecting options. 4. System checks that all conditions are valid. 5. System enables user to run simulation.
Flow of Events for Extension: 2a. User does not finish selecting options and exits menu. 4a. Parameters are not all valid. 4b. System alerts user to which conditions are invalid.

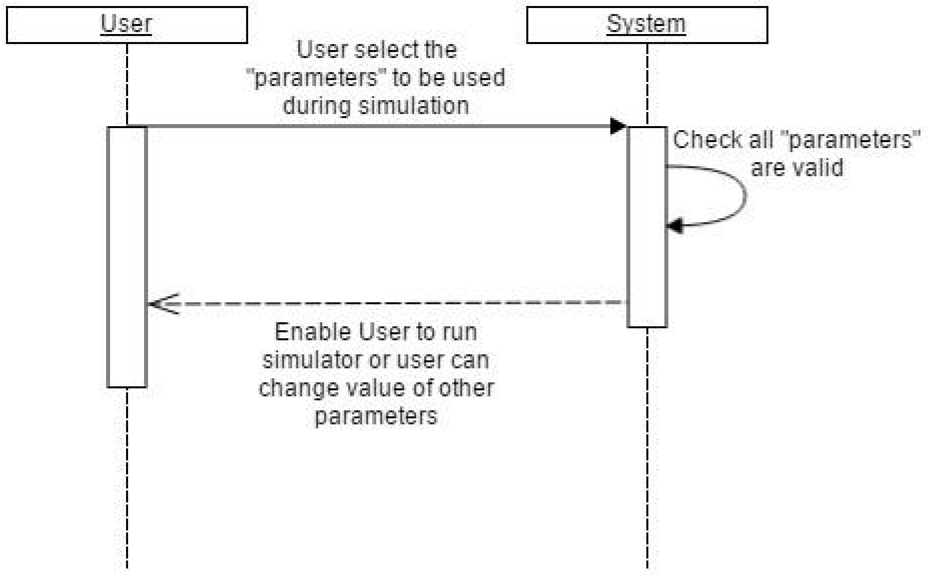
Use Case UC-1: SetInitial
Related Requirements: Req-1
Initiating Actor: User
Actor's Goal: Set values for conditions in simulation
Participating Actors: System
Preconditions: None
Postconditions: Simulation is ready to run
Flow of Events for Main Success Scenario: 1. User opens menu for selecting condition values.. 2. User selects values on menu. 3. User finishes selecting options. 4. System checks that all conditions are valid. 5. System enables user to run simulation.
Flow of Events for Extension: 2a. User does not finish selecting options and exits menu. 4a. Parameters are not all valid. 4b. System alerts user to which conditions are invalid.

Use Case UC-6: ShowResults
Related Requirements: Req-7
Initiating Actor: User
Actor’s Goal: Get results from simulation
Participating Actors: System
Preconditions: Parameters are all chosen
Postconditions: Simulation returns data.
Flow of Events for Main Success Scenario: 1. User clicks “Start simulation.” 2. Simulation runs on system. 3. System displays output in readable format.
Flow of Events for Extension: 2a. Simulation fails, error is reported.

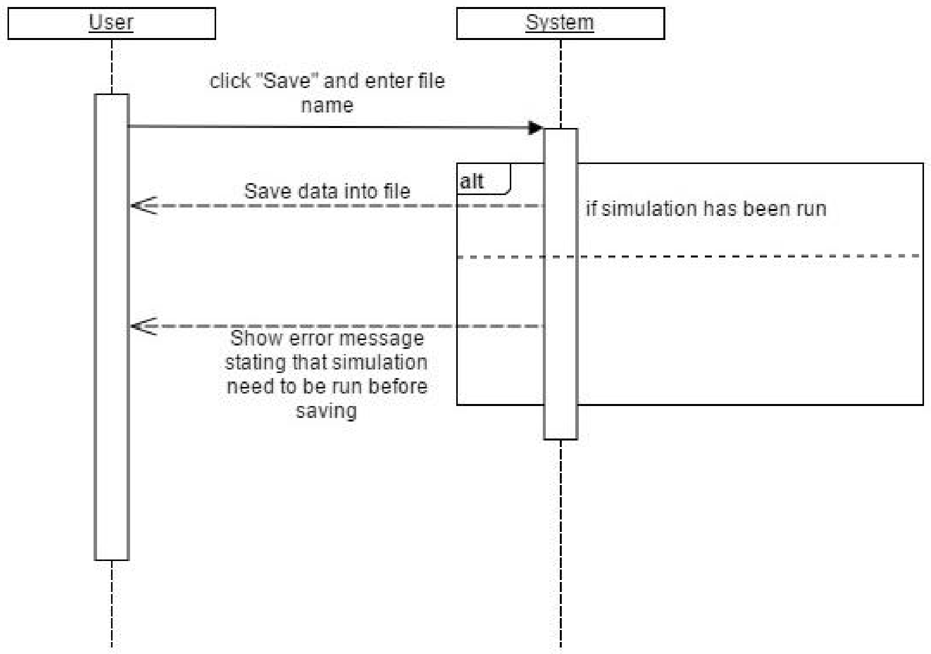
Use Case UC-4: SaveData
Related Requirements: Req-5, Req-8, Req-13
Initiating Actor: User
Actor’s Goal: Save data from simulation
Participating Actors: System
Preconditions: A simulation has successfully returned data
Postconditions: Data is saved
Flow of Events for Main Success Scenario: 1. User chooses save option. 2. User selects file name to save data into. 3. System saves data.
Flow of Events for Extension: 1a. No data is available because simulation has not been run, return error. 3a. System cannot save data, return error.

System Sequence Diagram

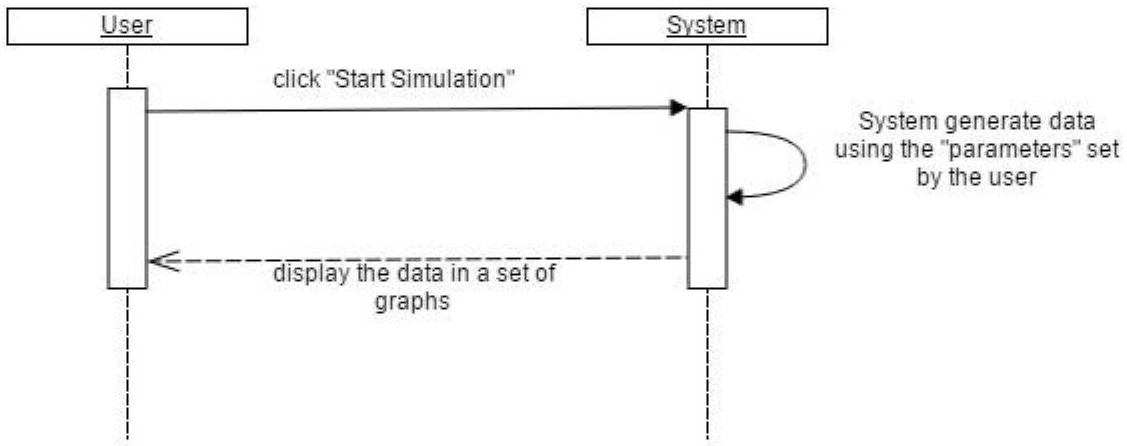
The system sequence will show the interaction between the system and the user. This said to be the boundary between the user and the system. By providing the system sequence diagram, every possible input from the user side will need to be predicted so that the system can manage the input. The most important is for the system to manage the invalid input so that the system will not crash. The more detailed descriptions of how system manage all the input will be provided in the design sequence diagram. Below are some of the system sequence diagram for the high priority use cases.


User
System
User select the "parameters" to be used during simulation
Check all "parameters" are valid
Enable User to run simulator or user can change value of other parameters

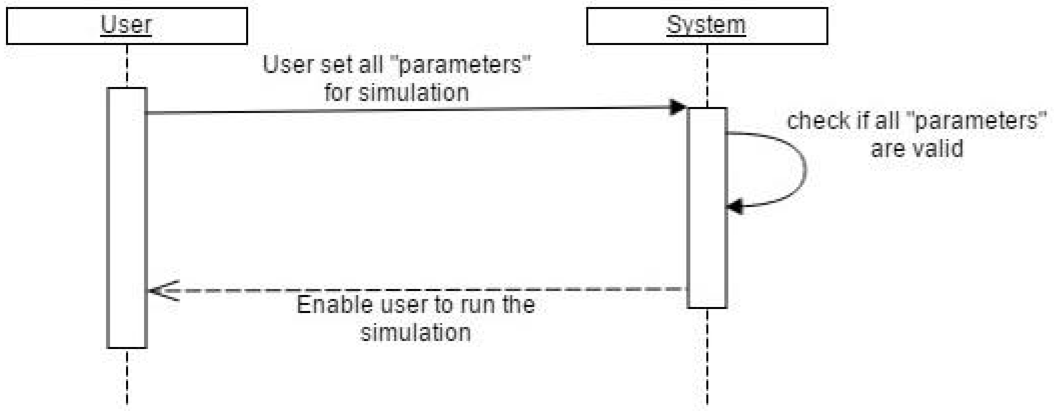
Use Case 2 (UC-2)


User
System
click "Save" and enter file name
alt
Save data into file
if simulation has been run
Show error message stating that simulation need to be run before saving

Use Case 4 (UC-4)


User
System
click "Start Simulation"
System generate data using the "parameters" set by the user
display the data in a set of graphs

Use Case 6 (UC-6)

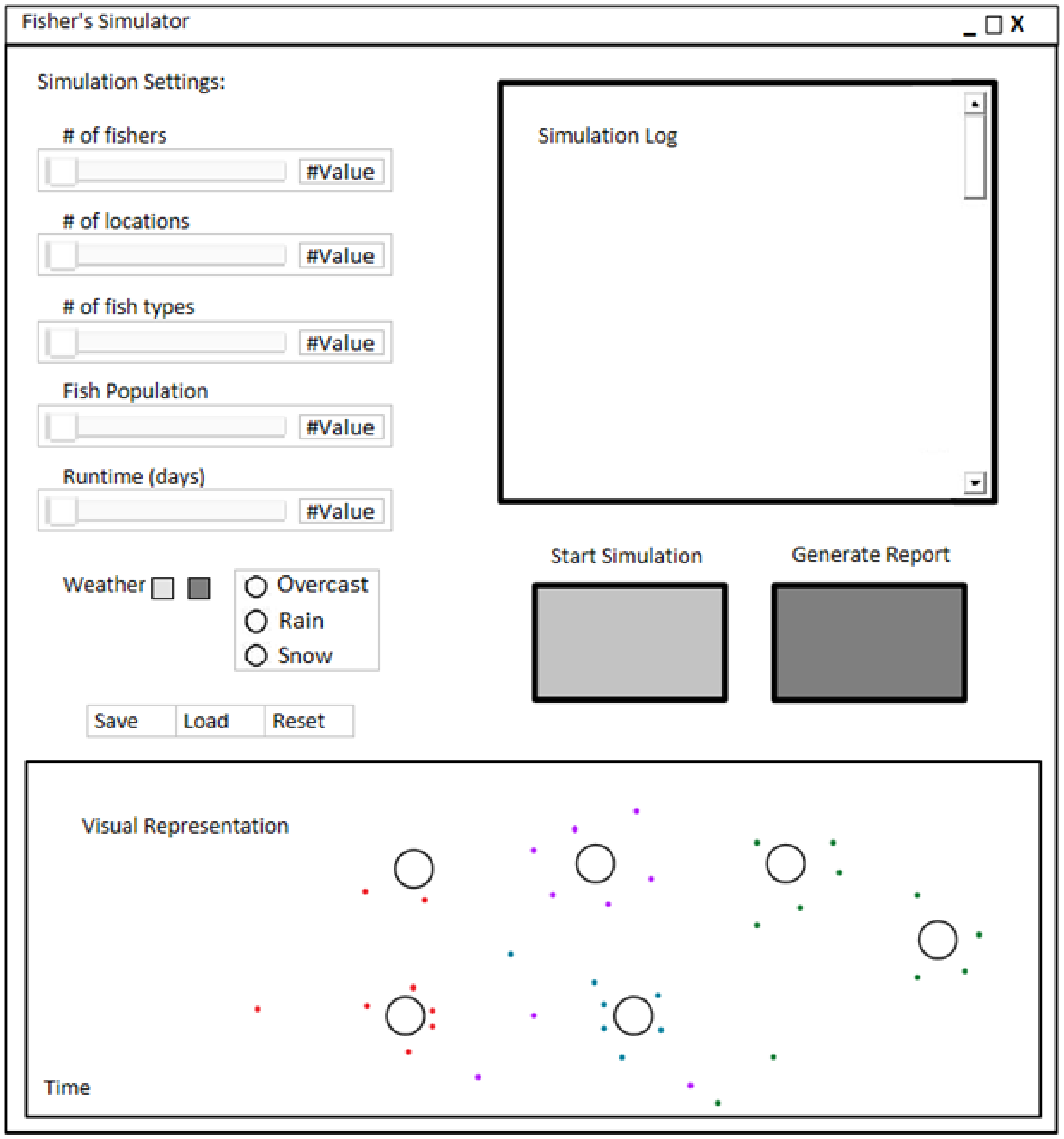

User
System
User set all "parameters" for simulation
check if all "parameters" are valid
Enable user to run the simulation

Use Case 1 (UC-1)

User Interface Specification

Preliminary Design

Below is a cleaned up version of the user interface mockup based on customer requirements.



User Effort Estimation

1. Run simulation (reasonable worst case scenario)
    a. Set number of fishers with a click and drag motion (one action)
    b. Set number of locations with a click and drag motion (one action)
    c. Set number of fish types with a click and drag motion (one action)
    d. Set fish population with a click and drag motion (one action)
    e. Set runtime with a click and drag motion (one action)
    f. Click the checkbox to enable weather (one click)
    g. Select from weather options (one click)
    h. Click “Start Simulation” (one click)

    Total actions: 8

2. Save simulation configuration
    a. Click “Save” (one action)
    b. Navigate to the desired directory (0 – many actions)
    c. Enter filename (0-10 keystrokes)
    d. Click save or press enter (one action)

    Total actions: 2 - many

3. Load simulation configuration
    a. Click “Load” (one action)
    b. Navigate to the desired directory (0 – many actions)
    c. Select configuration file (one action)
    d. Click load or press enter (one action)

    Total actions: 3 - many

4. Generate report and save output
    a. Run simulation (1 – 8 actions)
    b. Click “Generate Report” (one action)
    c. Click “Save Report” (one action)
    d. Navigate to the desired directory (0 – many actions)
    e. Enter filename (0-10 keystrokes)
    f. Click save or press enter (one action)

    Total actions: 4 - many

Domain Analysis

Domain Model

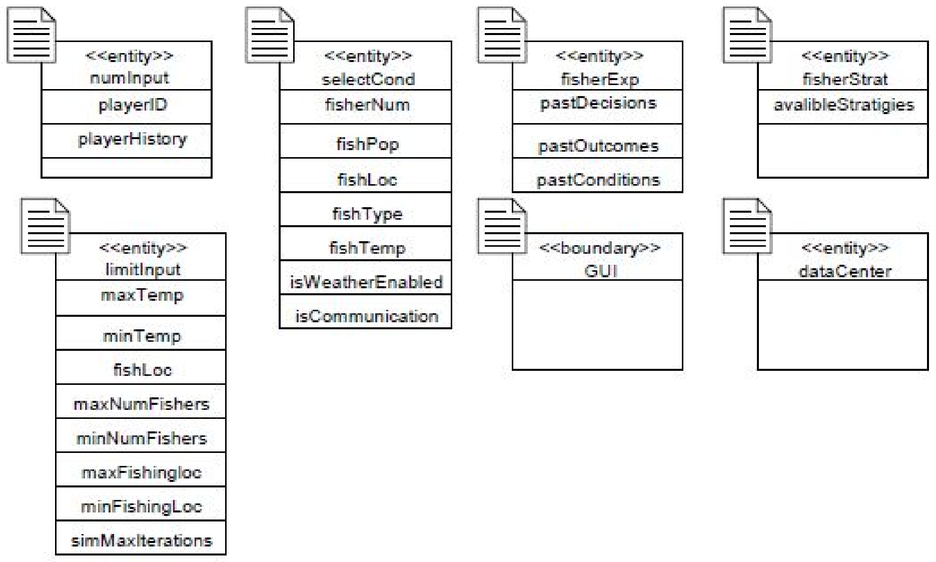
Concept Definitions

According to *Software Engineering* by Ivan Marsic, there are can be two types of responsibilities. There are type D and type K responsibilities which are about "doing" and "knowing". A type D or "doing" responsibility requires making an action to get the specified results like data calculation. Meanwhile, type K or "knowing" responsibility requires to only remember a specific data like a data container. Sometimes it would be difficult to distinguish between the two types and the label would be blank.These are used to label all concept definitions below.

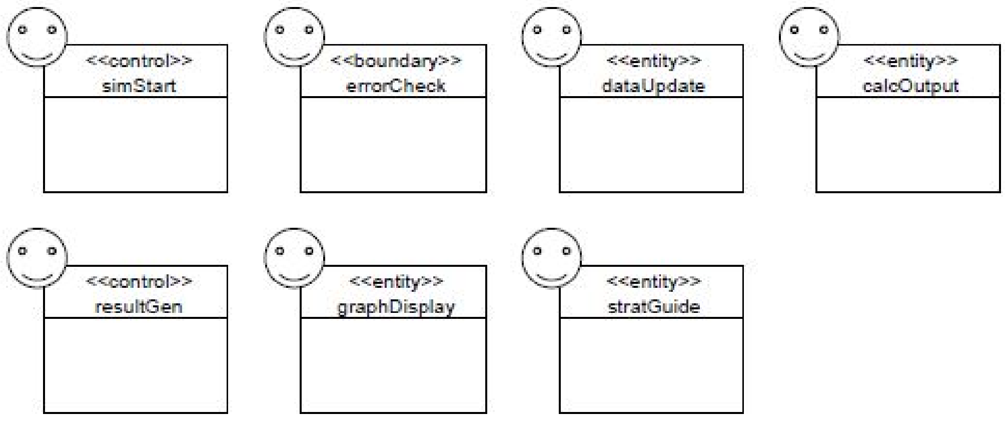
Responsibility Description	Type	Concept Name
The container that remembers all user's inputs	K	numInput
The container of all selected conditions/factors	K	selectCond
Contains the number of fisherman	K	fisherNum
Contains the population of fish	K	fishPop
Contains the number of fishing locations	K	fishLoc
Contains the user's experience	K	fisherExp
Contains the user's strategies	K	fisherStrat
Contains the types of fish	K	fishType
Contains the weather temperature	K	fishTemp
Contains the limits of all input parameters	K	limitInput
Starts the simulation program	D	simStart
Checks if user inputs are valid and sends error message	D	errorCheck
Contains all the user's saved results, experience and strategy.	K	dataCenter
Updates the user's experience, strategies and selected conditions after simulations in the data center.	D	dataUpdate

Calculates output results using user inputs using algorithm/mathematical model.	D	calcOutput
Generates the results of the simulation in real time simulation.	D	resultGen
Shows the final graphs of the simulation.	D	graphDisplay
Give out strategies according to user data	D	stratGuide
Provides overall display of the software program for the users.	K	GUI

Type K diagram:



Type D diagram:

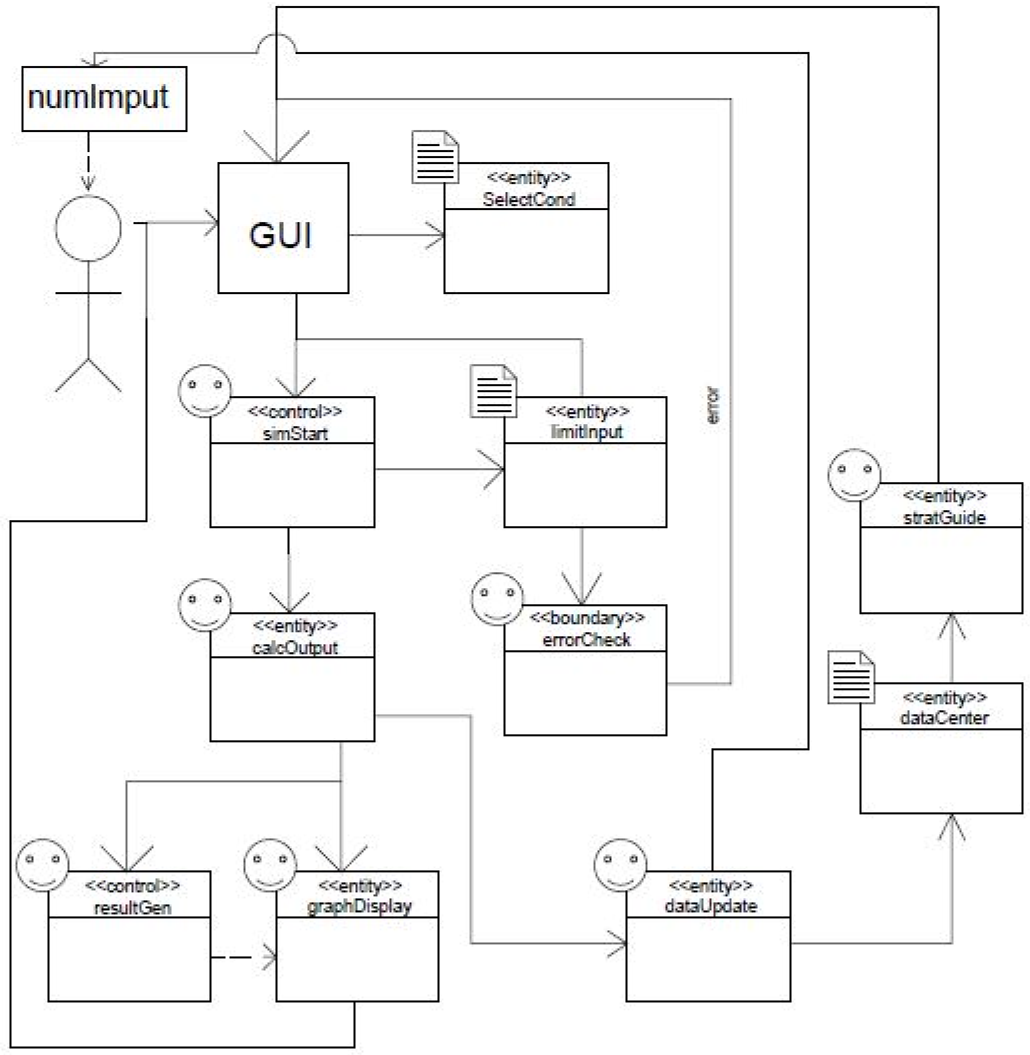


Association Definitions

The responsibilities above can be associated with each other to convey or update information. The arrows show that there is a relationship between two responsibilities.

Concept Pair	Association Description	Association Name
numInput ↔ GUI	The user inputs parameters via GUI.	Simulation input
selectCond ↔ GUI	User selects conditions on the GUI.	Simulation conditions
simStart ↔ GUI	User presses start to begin.	Press start
simStart ↔ limitInput	When user presses start the inputs would be compared with limits.	Compare limits
limitInput ↔ errorCheck	If the user inputs are not within limits the simulation stops, else it begins.	Simulation check
errorCheck ↔ GUI	If there is an error, a message is sent back to user on the GUI.	Error return
simsStart ↔ calcOutput	The software begins calculating for results when user presses start and there are no errors.	Simulation start
calcOutput ↔ resultGen	Output is calculated and shown on real time for users to watch.	Simulation output
calcOutput↔ graphDisplay	After all calculation, the graphs are displayed.	Simulation graphs
graphDisplay ↔ GUI	The final graphs are shown on the GUI.	Graph Output
numInput ↔ dataUpdate	User experience and strategies would be updated.	Save values
selectCond ↔ dataUpdate	User selected conditions would be updated.	Save conditions
graphDisplay↔dataUpdate	The final result would be saved and updated.	Save graphs
dataCenter ↔ stratGuide	A strategy guide would be created based on data.	Create strategy

stratGuide ↔ GUI	Send user new strategies on GUI.	Return strategy
dataUpdate ↔ dataCenter	All updated data would be in the data center.	Save data



Attribute Definitions

The following attribute definition seeks to identify the attributes that will be needed in each of the concepts identified. The selectCond concept deals with the settings that can be chosen by the user when setting up the simulation.

Concept	Attribute	Responsibility
numInput	playerID	saves the id of each player
	playerHistory	saves a history of each players past decisions
selectCond	fisherNum	contains the number of competing fishermen
	fishPop	contains the starting fish population
	fishLoc	Contains the number of possible fishing locations
	fishType	Contains the type of fish available. IE Bass, Catfish, etc.
	fishTemp	Contains the water temperature
	isWeatherEnabled	When true weather is considered in the simulation
	isCommunication	When true communication between fishermen is enabled in the simulation calculation.
fisherStrat	avalibleStratigies	A list of all of the strategies available for the fishermen to use in determining if they should fish and for what.
fisherExp	pastDecisions	Contains the past decisions that this fisher has made
	pastOutcomes	Contains the results of the fishers choices (win / loose)
	pastConditions	Contains the conditions (weather/ temp) when each past choice was made
limitInput	maxTemp	The maximum allowed water temperature.
	minTemp	The minimum allowed water temperature.
	maxNumFishers	The maximum number of fishermen allowed
	minNumFishers	The minimum allowed water temperature.
	maxFishingloc	The maximum number of fishing locations the

		simulator will handle.
	minFishingLoc	The minimum number of fishing locations. Default (1)
	simMaxIterations	The maximum number of iterations the simulation can run.
dataCenter		Contains all of the past data. This includes the past elements that could affect the simulation, fishermen choices, and the aggregated results.

Traceability Matrix

Each of the use cases have the following dependencies on concepts. Because UC-3 is running the simulation, it depends on almost all of the concepts. Also the Saving and restoring use cases (UC-4, UC-5) could depend on more concepts depending on how much data we save. Lastly UC-9 deals with providing help for the user and notifying users of errors. This use case encompeses each concept because they each could have errors that the user must be notified about.

Use Case	**PW (1-5)**	**numImput**	**selectCon**	**fisterStrat**	**fisherExp**	**LimitInput**	**dataCenter**
UC-1	4	X	X			X	
UC-2	4	X	X			X	
UC-3	5		X	X	X	X	X
UC-4	2				X		X
UC-5	2	X	X		X		X
UC-6	4				X		X
UC-7	3			X			
UC-8	2				X		X
UC-9	1	X	X	X	X	X	X

System Operation Contracts

Operation:	errorCheck
Cross Reference:	UC-1, UC-2
Preconditions:	User must have input settings
Postconditions:	Returns error if input settings are incorrect, allows simulation to be run otherwise

Operation:	dataUpdate
Cross Reference:	UC-4
Preconditions:	Simulation must have run successfully
Postconditions:	Stores data from simulation into data center

Operation:	calcOutput
Cross Reference:	UC-6
Preconditions:	Must have run a valid simulation
Postconditions:	Returns data on simulation to be stored or viewed

Operation:	simStart
Cross Reference:	UC-6
Preconditions:	User must have input settings that were checked and found to be valid
Postconditions:	Simulation begins running

Operation:	graphDisplay
Cross Reference:	UC-6

Preconditions:	Simulation must have successfully run, data must have been calculated
Postconditions:	Displays graphs built from the data calculated from the simulation

Mathematical Model

Minority games model can help an agent to make a decision in a competitive fishing environment. The basic fundamental for this mathematical model is an agent considered win if the fishing spot is least crowded because the agent will have a higher chance to get a fish. We will take an odd number of players N to choose one of two choices independently at each turn. Each of the N players takes an action deciding either to go to the bar (denoted as $a_i(t) = 1$) or stay at home (denoted as $a_i(t) = -1$).

$$A(t) = \sum_{j=1}^{N} a_j(t)$$

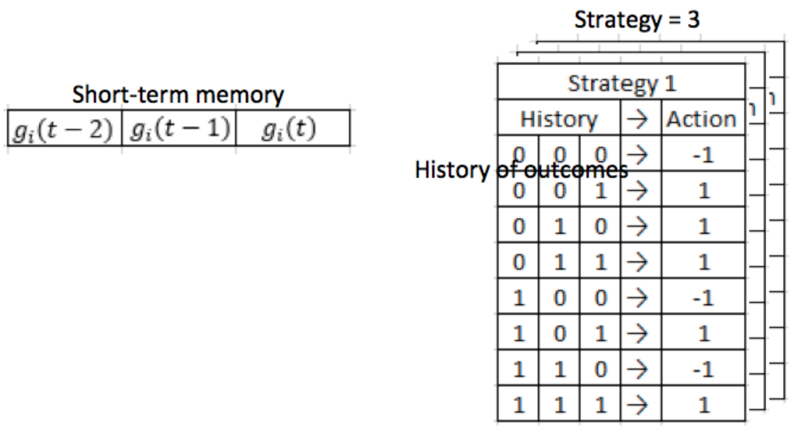
Since the number of players is odd, the summation formula will not go to zero. So, by using the above formula, we can conclude that if $A(t) < 0$, majority of the players stayed at home and the fishing spot is not crowded and a higher chance for the players to get a fish. On the other hand, if $A(t) > 0$, majority of players go to fishing and this will imply a lower chance for a player to get a fish.

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

The function $g_i(t)$ will represent outcome of the current round for the player i and will make sure the agents with minority action are rewarded. Since we are only interested in winning (denoted by 1) or losing (denoted by 0) we will take the absolute value of $g_i(t)$.

$$g_i'(t) = \begin{cases} 0 & if\ g_i(t) < 0 \\ 1 & if\ g_i(t) > 0 \end{cases}$$

Players will have a short-term memory and limited to 3 previous outcomes. On top of that, each player will have sets of strategy which also limited to 3 strategies each player. Below is the illustration of the short-term memory and long-term memory.



Strategy 1				
History			→	Action
0	0	0	→	-1
0	0	1	→	1
0	1	0	→	1
0	1	1	→	1
1	0	0	→	-1
1	0	1	→	1
1	1	0	→	-1
1	1	1	→	1

When the simulation starts, each player will start with a random strategy. Each strategy will be able to keep score of its success rate. The higher the success rate of that certain strategy, the more likely for the player to use that certain strategy. Calculation of the success rate of that certain strategy as follows. If the S_{ij} suggested to stay at home ($a_{ij} = -1$), and the fishing spot turn out to be crowded ($A(t) > 0$), then S_{ij} is added with one point. Conversely, if the fishing spot turn out to be less people ($A(t) < 0$), the point will be lowered by one.

$$\sigma_{ij}(t) = \begin{cases} \sigma_{ij}(t-1) - 1 & if\ a_{ij}A(t) < 0 \\ \sigma_{ij}(t-1) + 1 & if\ a_{ij}A(t) > 0 \end{cases}$$

Since we are adding more features that will influence the decision making of the players, we will add one more step before calculation the success point of each strategy. After each players gets the suggested decision, a_{ij}, from the strategy S_{ij}, the decision can be change by the following factors.

· Skill and experience rank
· Frequency of communication
· Amount of each type of fish
· Fishing duration
· Weather pattern

We decided that each factors will contribute 20 percent increase of influence to change the current decision, a_{ij}, of the player. The higher percentage of the influence the more likely the player to change their decision. The threshold influence percentage to change the player decision is 70 percent. To calculate the influence percentage, p, we will use the formula below.

· Skill and experience rank

o This will be rank from 1 – 5, so to get the influence percentage for this factor we calculate it as below.

$$E = \frac{rank}{5} \times 20$$

· Frequency of communication

o Since this represent the percentage of communication between the players, this can be easily calculated as below.

$$C = \frac{Frequency\ of\ communication}{N, number\ of\ players} \times 20$$

· Amount of each type of fish

o The type of fish that players wanted to catch will be accumulated first. Then the percentage of players that interested in catching the fish will be calculated. The higher the percentage of interested players in catching a certain type of fish, the lower the influential rate of the fish. This is because, more players are interested in one type of fish, will cause a higher competitiveness. So it is more likely that a player to just stay at home. To get the influence percentage of this factor we will calculate it as below.

$$F = \left(\frac{Number\ of\ interested\ players\ in\ the\ type\ of\ fish}{Amount\ of\ type\ of\ fish} \times 20\right) - 20$$

· Fishing duration

o We assume that the player that spend a longer time in fishing will have a higher chance in getting a fish and since we go for a success rate of each player in a day, we will calculate the fishing duration influence rate as follow.

$$T = \frac{Fishing\ duration}{24\ hours} \times 20$$

· Weather pattern

o In general, the best temperature to fishing is around 55 – 80 Fahrenheit. The average temperature would be 67.5 Fahrenheit. The influence rate for weather pattern will be calculated as follow.

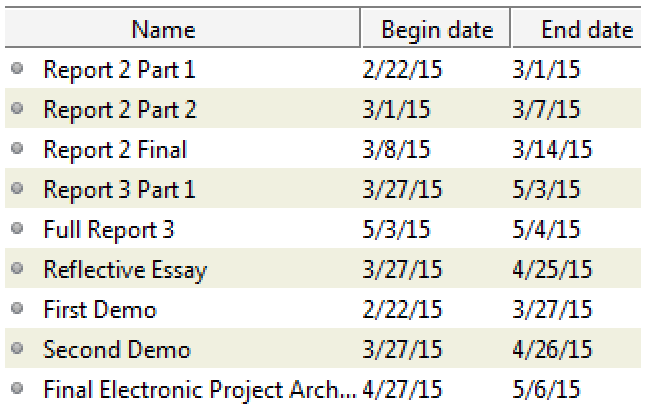
$$W = \left(-\left|\frac{Average\ Temperature - Today's\ Temperature}{Average\ Temperature}\right| \times 20\right) + 20$$

After all the influence rates of each factors has been calculated, we can get the overall influential percentage (denoted by p) and decide if the player should changes the decision. If $p < 70$, the decision a_{ij} that has been made by S_{ij} will not be change. If $p > 70$, any decision $a_{ij} = -1$, will be change to 1.

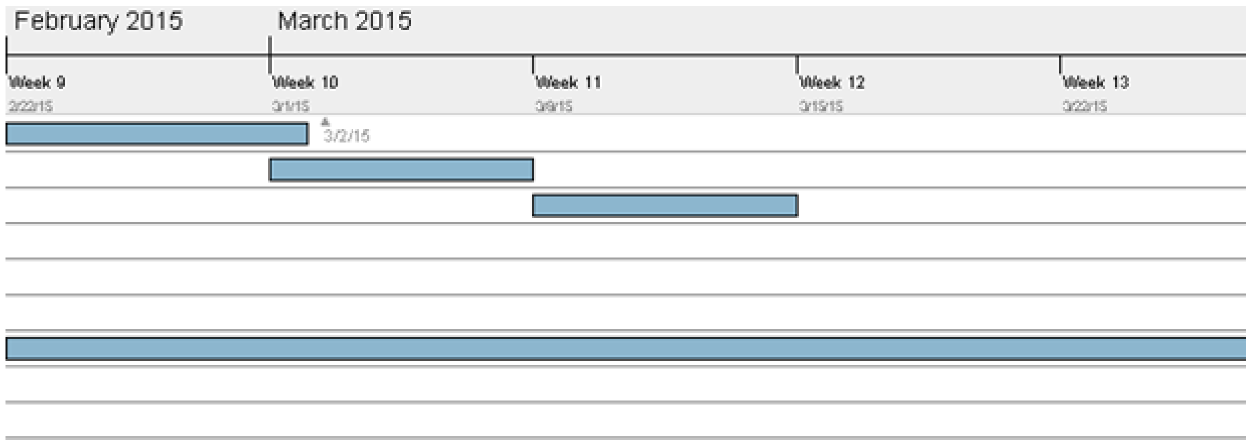
$$g_i'(t) = \begin{cases} 0 & if\ g_i(t) < 0 \\ 1 & if\ g_i(t) > 0 \end{cases}$$

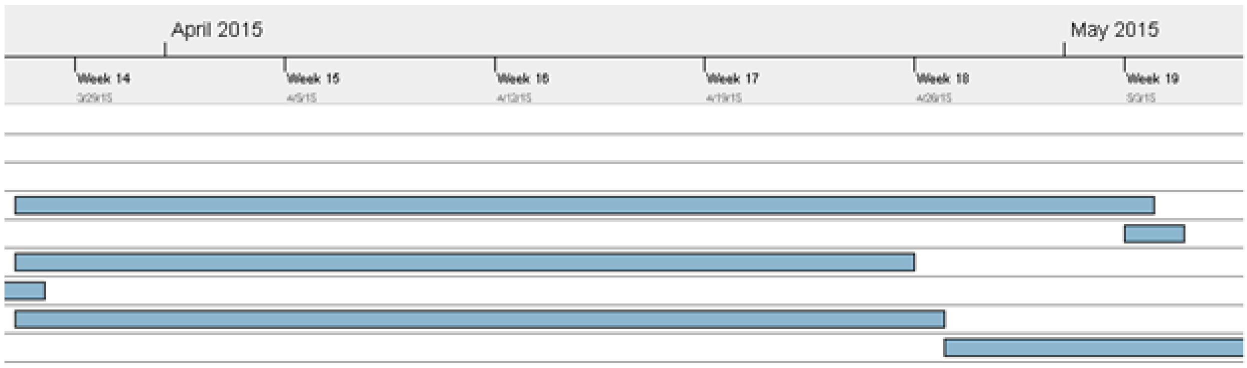
Plan of Work/Project Management

With the first report behind us, the group will move to more directly planning software development. After an analysis of previous projects, we may decide to adopt one to improve upon or create our own from scratch. Work on the next report, focusing on software design, will begin simultaneously. Key features will be implemented and tested in time for demo 1 scheduled for March 27. Following demo 1, more focus will be paid to getting our novel and unique features implemented, such as the real time display of simulation results. These will be completed in time for demo 2 on April 28.



Name	Begin date	End date
Report 2 Part 1	2/22/15	3/1/15
Report 2 Part 2	3/1/15	3/7/15
Report 2 Final	3/8/15	3/14/15
Report 3 Part 1	3/27/15	5/3/15
Full Report 3	5/3/15	5/4/15
Reflective Essay	3/27/15	4/25/15
First Demo	2/22/15	3/27/15
Second Demo	3/27/15	4/26/15
Final Electronic Project Arch...	4/27/15	5/6/15





Contribution Table:

PROJECT Distribution	Team Member					
	Matthew Chatten	Ameer Fiqri Barahim	Vincent Vindel Dura	Alexander Hill	David Lazaar	Orielle Joy Yu
Simulation Settings and Options **(20 points)**	50%		50%			
Simulation Results Output **(20 points)**		50%				50%
Simulation Processing **(20 points)**	100%					
Simulation Real Time Display Output **(20 points)**				50%	50%	
Graphical User Interface					100%	

References

1.) "El Farol Bar Problem." *Wikipedia*. Wikimedia Foundation, n.d. Web. 08 Feb. 2015.

2.) "The Minority Game: An Introductory Guide." *Implicit None*. N.p., 31 Aug. 2004. Web. 08 Feb. 2015.

3.) “Concepts: Requirements” *UPEDU*. Polytechnique Montreal, 2014. Web. 07 February 2015.

4.) Marsic, Ivan. "Object-Oriented Software Engineering." *Software Engineering*. 2012. 110-113. Print.