Simualtion of Minority Game

Group Members:	Zhan Chen	johnalwaysyoung@gmail.com
	Xiaoheng Liu	fsr0023120@gmail.com
	Boyu Ni	niboyu@live.com
	Pengcheng Wan	kevinwan1991@hotmail.com
	Jinhe Shi	jhshi@hotmail.com
	Zhengyang Zhong	timothy7784@gmail.com

Report 1 2013/10/15

Breakdown

	point	Team Member Name					
		Zhan Chen	**Xiaoheng Liu**	**Boyu Ni**	**Pengcheng Wan**	**Jinhe Shi**	**Zhengyang Zhong**
Project management	10	**10%**	**10%**	**40%**			**40%**
CSR	9	**50%**			**50%**		
System Requirements	6		**33%**	**33%**		**33%**	
Functional Requirements Specification	30	**50%**	**50%**				
User Interface Specs	15						**100%**
Domain Analysis	25				**50%**	**50%**	
Plan of Work	5			**100%**			

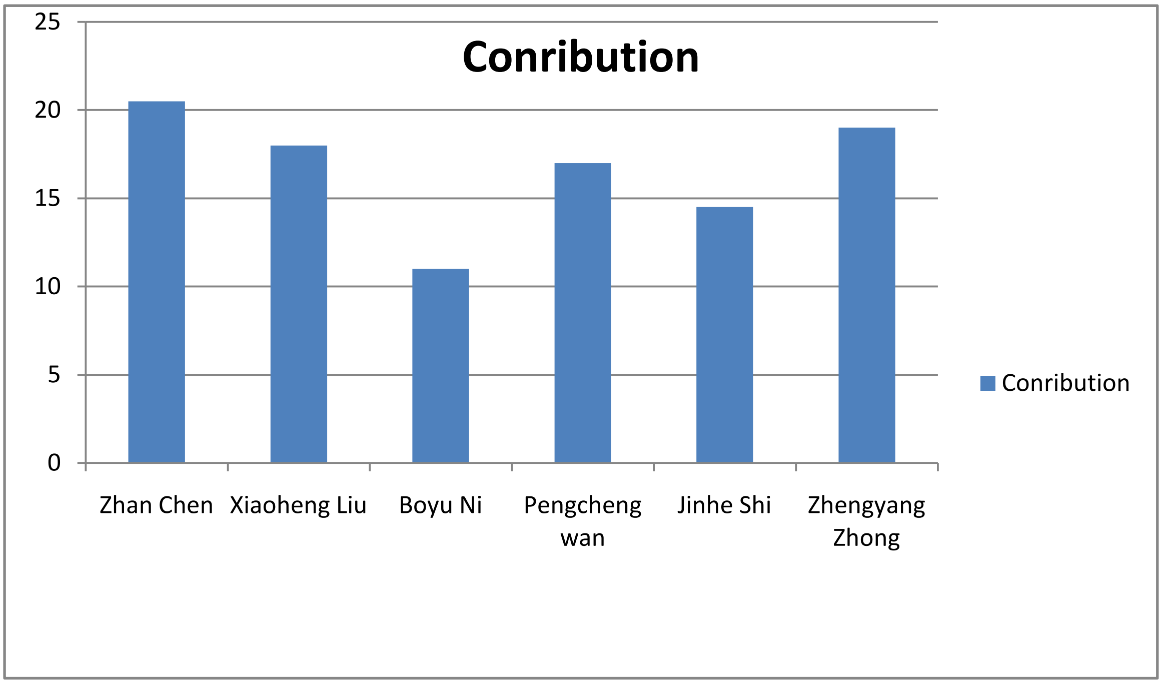

Conribution
25
20
15
10
5
0
Zhan Chen
Xiaoheng Liu
Boyu Ni
Pengcheng wan
Jinhe Shi
Zhengyang Zhong
Conribution

Table of Contents

1. Customer Statement of Requirements

1.1 Problem Statement

There has been a time that people are confused about how to make the right decision. Thanks to quick developing of the technology, now we have computers to help us simulate the situation of real world to solve our problems. Through simulation, it is not difficult to figure out what decision should be made in certain situations.

In order to make it convenient for us to make decisions in various circumstances, we need software which is capable of simulating the situations. Thus, we can easily deal with our problems just by adjusting some parameters. In other words, by clicking the mouse (adjusting the parameters) we can simulate the environment we are involved in, so we can get the result in advance.

The system should be mainly designed for finance field, but it should be used in other cases as well.

As for the us, by using this software, for example, we are actually shown a film or watching a documentary film about what's happening for a simulated market and experience various results that different parameters I choose bringing about.

In fact, this is a simulated market presented in front of us, and it is us, the customers to decide what the basic factors in the market are. A certain numbers of agents will come to buy or sell a stock. And the winners will certainly, be the minority. To simulate the real market, we need to set that the money lost by the majority will be redistributed to the winners as only the minority parcel social property. So, strategies for choosing to buy or sell at each time become of great significance, and this is the reason we need this software as well. Since we only see the choice and result of a single agent, we want someone to make the algorithms and useful methods.

Literally, we want the model to be even more like the real world, we want to see that every choice has its own impact to the next round. And we want information to be stored in each agent's memory. On one hand, we give some equal numbers of money to every agent, and drop the one who lost all his/her money; on the other hand, we will do some research of the person who gains more money than others.

However, this is not the end, as the situation can be more complex in the real world. We will find out that agents are divided by their characteristics , so there must be some excellent ones who win frequently, while noise agents (individuals gaining profit by following the herd) also exist. This will result in more complex situation, to better reflect the real competition, the herding will be a common sense for almost every one, and experts will be favored for herding.

Since we have a good interest in social relations, we may further see some experts cheating the others so that they will get the largest share of the benefits. Through running again and again, the final results will be shown for us to analyze the situation we are involved in and help us get the conclusion.

We hope this software can be applied on market trading, though some details are not exactly as the real financial market, but it should be a fairly good reference as a matter of fact. In other words, it should show relatively reasonable strategy for deciding whether to buy or sell.

To better describe our needs, at least 5 main parameters should be involved: short-term memory, long-term memory, scores of agents, life duration and herding.

We need this software to calculate 2 types of score: Agent score and strategy score. Agent score is used to estimate the successful rate of each agent. Strategy score is used to estimate the quality of the strategy.

The program should allow us to customize the memory of each member consists of long-term memory and short-term memory.

Short-term memory: The agents should be able to memorize the game results of previous rounds and make decision according to these results.

Long-term memory: Each agent should have several strategies and scores them in its long term memory. All strategies have a initial score and after every round, the strategies' score should add or subtract according to the result of the game. Thus, each agent can keep a running tally of each strategy's score in comparison to other strategies. The member will make decision depends on these "scores" to choose the strategy. The program should output the strategy with highest score and the agent who has the highest success rate. After several rounds, the strategy with bad performance should be excluded by the agent. In order to better simulate the real-world situation, agents should have different memory abilities.

The program should also include life duration (mortality) in the game, which will make members stop participating in the game when they die. Every agent will be given an age value randomly. The whole population of these dead people will be replaced by a whole younger generation. This is consistent with the law of nature and life.

Furthermore, when they die, part of their scores, or capital, should be distributed to the entire system, the other will be inherited by younger generation.

Converging crowd causes group effect. As a consequence, different and complex situations come into being. Generally, there are two situations: one is that people have their own fixed group and the agents of that group do not change; the other is that the agents of the group change with time owing to the historical records of others in the same group: agents with bad performance will be dropped out of the group while the relatively competitive ones stay, nevertheless, while someone gets out, there should be new group member added, simply make the total numbers constant.

In the situation that the agents of the group sway, we need a role for agents called advancer, with a definition of people who win a lot in this sequence of the game. These roles sometimes cheat in their own group so that they can obtain benefits of their own.

Moreover, we need a role called giant who ranked among the top. This role is endowed two behaviors: cheating and broadcasting. Besides cheating, giants broadcast to influence other people. All the behaviors above are purposed for minimizing the income of the rest and gaining the most benefits for themselves (sometimes they may guide other agents to win the game in order to acquire their trusts, but sometimes they prefer to minimize the numbers of winners so as to maximize their own benefits). In such situation, agents should be able to keep track of other agents' credit. If a certain agent cheats others for many times, its advice will be of less importance in other agents' decision making process.

1.2 Glossary of Terms

To better illustrate the contents, we list some important terms and our system graph below:

Scoring rules - It was introduced to help the customers to judge the gain and loss of the agents.

Memory effect - Every agent has its own memory length, like the situation in the real world that different people have different abilities of memorizing.

Life duration - Because in the real world, people have different life length, we introduce this parameter.

Herding effect - It is for simulating more complicated situations which is to simulate the behaviors of people who make their decision by referring other people's decision. This kind of behaviors especially exits in financial market.

Cheating - In order to gain the maximum profits of their own, the role advancer (including someone who ranks the top in this game) sometimes provide wrong information to their own group.

Broadcasting - The agents ranked among the top in this game are privileged to broadcast their advice to influence other agents' decision. This behavior is often related with cheating, since the ultimate goal of the top agents is to maximize their own benefits.

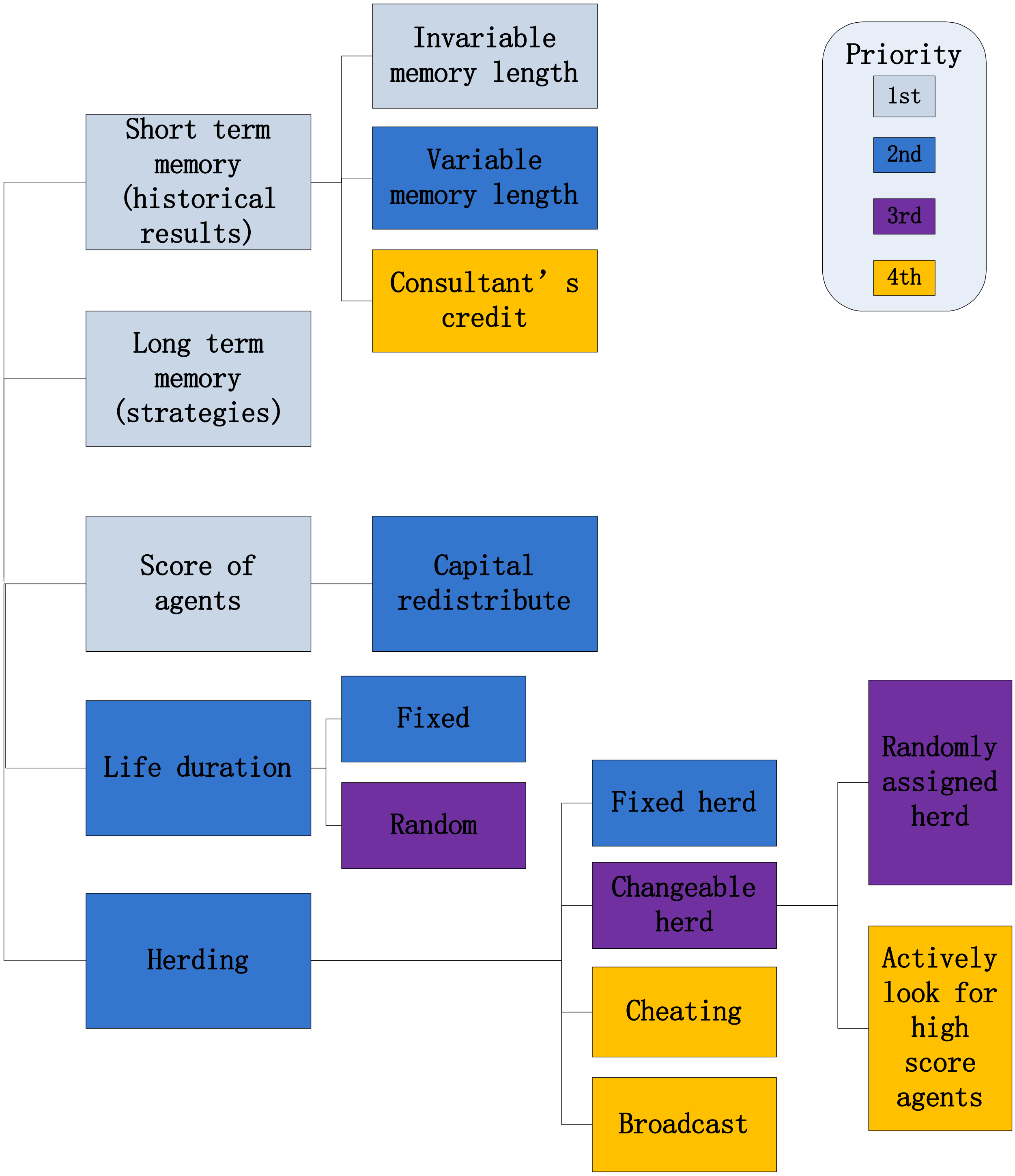


Fig 1 Flow sheet of parameters and priority

2. System Requirements

2.1 Enumerated Functional Requirements

Identifier	Priority weight(Low1-5High)	Requirement Description
REQ-1	4	Users should be able to choose whether or not to include mortality in this game
REQ-2	5	Agents with the minority decision win the round. Those who choose the majority lose that round.
REQ-3	4	Users should be able to choose whether or not to include memory in the game
REQ-4	3	Agents should have multiply strategies to use when making decisions each round and they choose each strategy randomly from strategies space (long-term memory).
REQ-5	3	High scoring strategies will be reused and low scoring strategies will be dropped
REQ-6	2	When agents die, their scores will be distributed to the system and inherited by younger generation(if mortality option chosen)
REQ-7	5	User should be able to set all initial conditions
REQ-8	3	Strategy should keep score to estimate the quality of each strategy
REQ-9	3	Agent should keep score to determine the successful rate of each agent.
REQ-10	3	Agent also should be given a credit score to estimate the credit degree of each agent

REQ-11	2	Agents should be able to retain memory of previous rounds' scores and make next decision depend on the previous scores (short-term memory).
REQ-12	4	User can select whether or not to include herding in the game
REQ-13	3	Agents should be able to choose to join a group to discuss and make decisions together(if herding option chosen)
REQ-14	3	Low score agents will be dropped out of the group, and new agents will be added. (if herding option chosen)
REQ-15	2	Agent who wins the most will be the advancer in the group and may cheat in their own group to obtain benefit (credit score).
REQ-16	1	Users must be able to change speed while simulation is running.
REQ-17	2	The program can be used in various circumstances (trading market)
REQ-18	5	Each agent will be given equal number of scores in the game.
REQ-19	1	User must have access to data collected in the form of graphs.
REQ-20	3	The length of short-term memory can be variable depend on different people.
REQ-21	5	User can decide the numbers of agents and rounds.
REQ-22	3	The user can decide life length of the agents(random or fixed)
REQ-23	3	Agents whose scores are less than m will be dropped out of the game and new people will be added.
REQ-24	3	User can choose whether the herding is fixed or changeable.

REQ-25	3	Agents will look for those who have high scores to be a group.
REQ-26	1	The program should keep running until the user feel they have enough data.
REQ-27	5	The totally number of agents in the game is constant.
REQ-28	2	The giants' strategy will broadcast to influence others

2.2 Enumerated non-functional requirement

A model called FURPS+ will be used here to qualify software attributes, which stand for functionality, usability, reliability, performance, supportability, and the + stands for other possible attributes needed. We will be focusing on the non-functional requirements which cover FURPS+.

Functionality: The functionality of our project if one of the most essential aspect. And our software will contain numerous functions and parameters which would help to solve problems in different situation. The logic level of the whole system, however, will be two as maximum. So the system will be characterized by multi-functionality and usability.

Usability: The logic layers of the system will be no more than two. Customers only need to select situation and add parameters to the simulation. We will do our best to minimize the mouse pointing time and maximize the function intuitionism.

Reliability: Frequency of failure should be very low. Customers only need to restart the software to recover and are able to choose whether to recover the last step they did. And software will be updated every month if not never.

Performance: The whole system also has high performance. Customers would wait no more than 5 minutes during the whole simulation procedures. And the running time, which depends on the parameters that chose by customers, will be few seconds as minimum and no more than 1 minute as maximum.

Supportability: The system is easy to understand by every user and programmer. There will be a user document to introduce how to use the program and an introduction demo will also be contained to the software. A technique support via E-mail will also be available and primarily to deal with the bug and imperfect part.

2.3 On-Screen Appearance Requirements

Identifier	Priority weight(Low1-5High)	Requirement Description
REQ-29	2	Help-button：A “help” button should on the top right corner next to the “close” button. A help document will appear after touching, every detail like the glossary, graphs and button use will be explained.
REQ-30	5	Error checking：The user interface has error checking function for all inputs. It will indicate what the error is to customers and how to revise it.
REQ-31	3	Range checking：not like error checking, a large range can still be operated, but will spend a lot of time sometimes even lead to computer crash. So we should warn the customer when the range is too big.
REQ-32	3	User-friendly：The operation of the user interface should be as easy to operate, no specialized training is need for new customers.
REQ-33	1	Aesthetic value：Nowadays, we live in the Market-Economy situation, a great application should not only pay attention to the function, but also to the aesthetic value. Thus our user may spend more time on the app.
REQ-34	1	Flexible：the user interface should be convenient for customers so it should be removable and can be amplified and lessened.

3. Functional requirements

3.1Financial professionals and related personnel

This software is mainly designed for people participating in financial market. It can also be used for other purpose, considering we provide many parameters into the software.

In the financial market, people make decision according to previous data, so we made this character the basis of the software. Briefly, in this software, all agents make decision according to the past results of others and themselves. We introduce the concept "short-term memory" and "long-term memory". Short-term memory refers to the memory of historical results. In this game, users can choose let the agents have variable or invariable short-term memory. Long-term memory relates to strategies memory. With long-term memory, agents make decisions according to the success rate of their past strategies.

In order to make this software more similar to real world situation, we introduce some other parameters. In the real world, people have life duration, so we introduce "mortality" in this software. Considering the variation of people's life duration, we add this choice in this software. User can chose whether life duration varies among agents. What's more, in the financial market, agents tend to communicate and exchange information before making decision. So we add the choice "herding". User can select this item to simulate the situation in the market that some people in this market take others' advices to make their own decisions.

Besides financial market, this software can be used in many other cases. In situations concerned with people and decision, this software is a useful way to help simulate the situation and analyze the decision accordingly. Airport, super mall and even the policy makings can use this software to simulate. Manufacturers can make decisions with this software to better participate in the market. Policy makers use this software to analyze whether the legislation can be passed. In fact, since there are so many parameters in this software, users can easily simulate many kinds of situations they need.

3.2 Actors and goals

Actor	Actor goals	Use case name
User	Set initial condition for the simulation	SetInitialConditions (UC-1)
User	Run simulation	RunSimulation (UC-2)
User	Change graphs	ChangeGranph(UC-3)
User	Change speed	ChangeSpeed(UC-4)
User	Stop simulation	StopSimulation(UC-5)
User	Show graph using data log of past simulation	ShowPastData(UC-6)

3.3 Use cases

UC-1	**SetInitialConditions**
Related requirements	Req-1,Req-2,Req-7,req-8,req-11,req-12,req-16,req-17,req-18,req-19,
Initiating actor	User
Actor's goal	To set initial conditions for the simulation
Participating actors	System, User
Preconditions	Initial screen is showing
Post conditions	Initial conditions are set and Simulation is ready to start
Flow of events for main success scenario	
→	User (a) selects the menu item "long memory" (b) types in value
→	User (a) selects the menu item "short memory" (b) types in value
→	User (a) selects the menu item "score of agents" (b) types in value
→	User (a) chooses to include or not include "life duration"
→	User (a) chooses to include or not include "herding"

→	User chooses name of output file
←	System verifies all values sets them in the Simulation
Flow of Events for Alternate Scenarios:	
→	5.a.User chooses to include “mortality”
→	User (a) selects the menu item “life duration” (b) types in value
→	6a. User chooses to include “herding”
→	User (a) selects the menu item “herding” (b) types in value

UC-2	**RunSimulation**
Related requirements	Req-1,Req-2,req-6,Req-7,req-8,req-11,req-16,req-19 ,req-24,req-25
Initiating actor	User
Actor’s goal	To run a Simulation
Participating actors	System
Preconditions	Initial conditions have been set
Post conditions	User believes that they have gathered enough data
	Extends :: SetInitialConditions Includes :: SetSpeed, ChangeGraphs
Flow of Events for Main Success Scenario:	
→	1.User chooses the button “Start Simulation”
→	2.User chooses the initial graphs to be shown
←	3.System generates the specified data and shows user selected Graphs
→	4.Graphs and speed may change based on user preference
→	5.User hits the “Stop Simulation” button

UC-3	ChangingGraphs
Related requirements	REQ-3 REQ-9, REQ-17, REQ-24,
Initiating actor	User
Actor's goal	Change the graph form while simulation is on.
Participating actors	System
Preconditions	1.Simulation is on 2.User wants to change graphs on the screen.
Post conditions	1.Show new graphs. 2.The simulation is still on.
Flow of events	
→	User chooses a new graph.
→	User chooses how many post rounds they want to show in the new graphs.
←	System changes the current graphs into the chosen one.

UC-4	SetSpeed
Related requirements	REQ-15
Initiating actor	User
Actor's goal	Change the speed of the system.
Participating actors	System
Preconditions	1.Simulation is on. 2.User wants to change the speed of the system.
Post conditions	1. Speed is changed 2. The simulation is still on.
Flow of events	
→	1. User chooses a new operating speed.
←	2. System changes the currently speed into the chosen one.

UC-5	**StopSimulation**
Related requirements	REQ-24
Initiating actor	User
Actor’s goal	To stop the simulation and generate output file
Participating actors	System
Preconditions	Simulation is running and User wants to stop it
Post conditions	Simulation has been stopped, log file has been output and Simulation is ready to run again.
	Extends :: RunSimulation
Flow of Events for Main Success Scenario:	
→	1. User presses button “Stop Simulation”
←	2. System finishes updating data to log file
←	4. System returns to Initial Screen for another run

UC-6	**ShowPastSimulation**
Related requirements	REQ-17, REQ-27
Initiating actor	User
Actor’s goal	To see graphs from past Simulation
Participating actors	System
Preconditions	A past Simulation has finished and User wishes to review it
Post conditions	User has reviewed past Simulation
Flow of Events for Main Success Scenario:	
→	1. User presses “read file” button
→	2. User selects a log file
←	3. System retrieves data from the file
→	4. User chooses graphs and number of turns that they care about
←	5. System Shows the requested graphs

Use Case Diagram

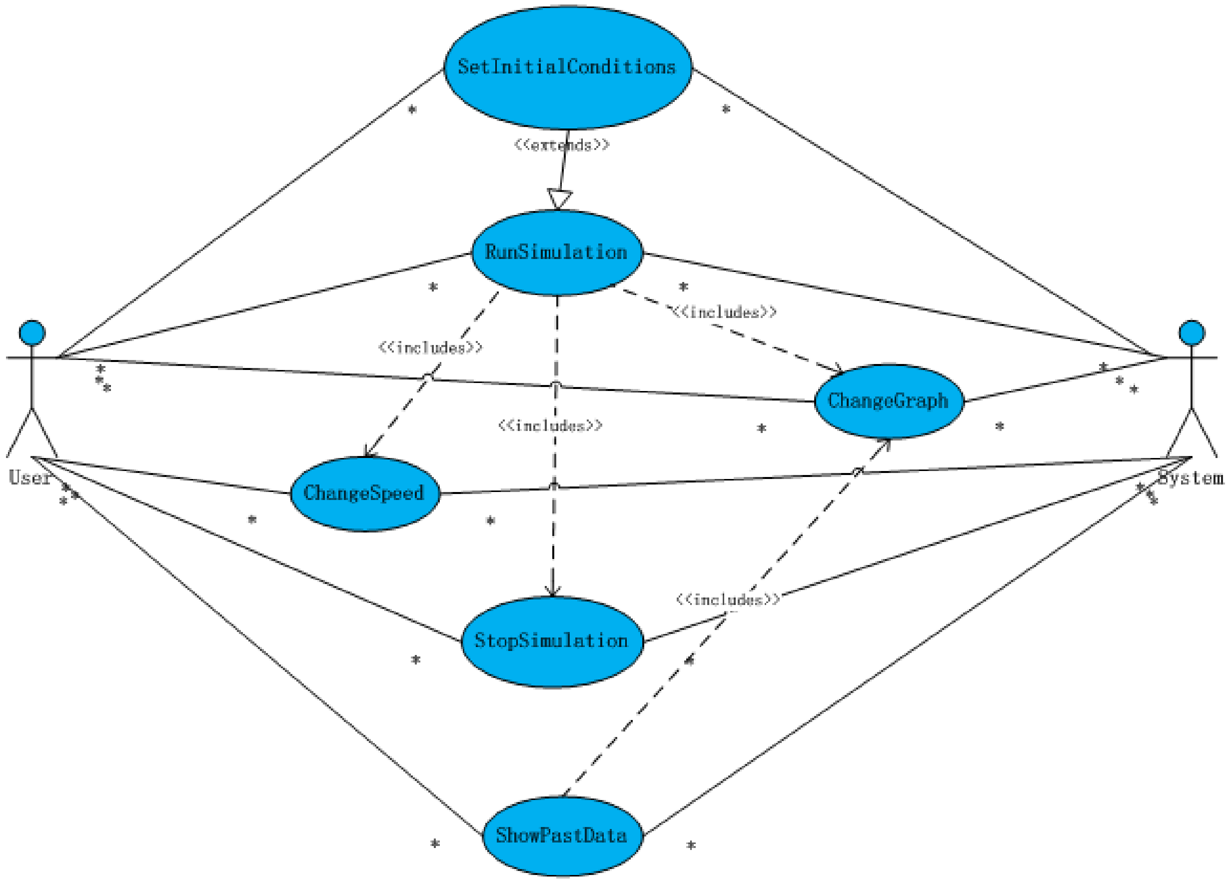


Fig 2 Use Case Diagram

Traceable Matrix

Traceability VS Requirement	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6
REQ-1	×	×				
REQ-2	×	×				
REQ-3			×			
REQ-4						
REQ-5						
REQ-6		×				
REQ-7	×	×				
REQ-8	×	×				
REQ-9			×			
REQ-10						
REQ-11	×	×				
REQ-12	×					
REQ-13						
REQ-14						
REQ-15				×		
REQ-16	×	×				
REQ-17	×		×			×
REQ-18	×					
REQ-19	×	×				
REQ-20						
REQ-21						
REQ-22						
REQ-23						
REQ-24		×	×		×	
REQ-25		×				
REQ-26						
REQ-27						×

3.4 System Sequence Diagrams

Uc-1:SetInitialConditions

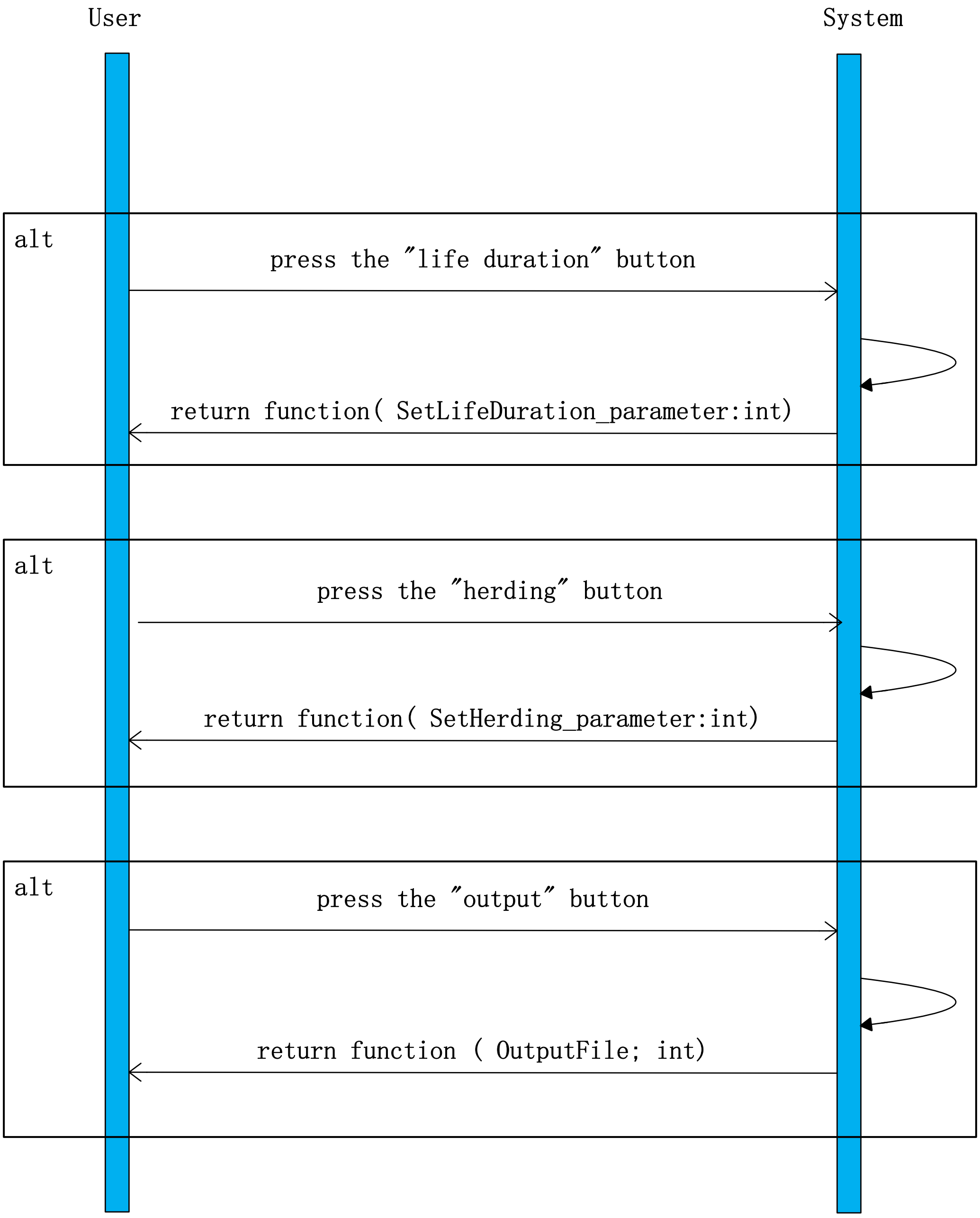


Fig3 SetInitialConditions

UC-2: RunSimulation

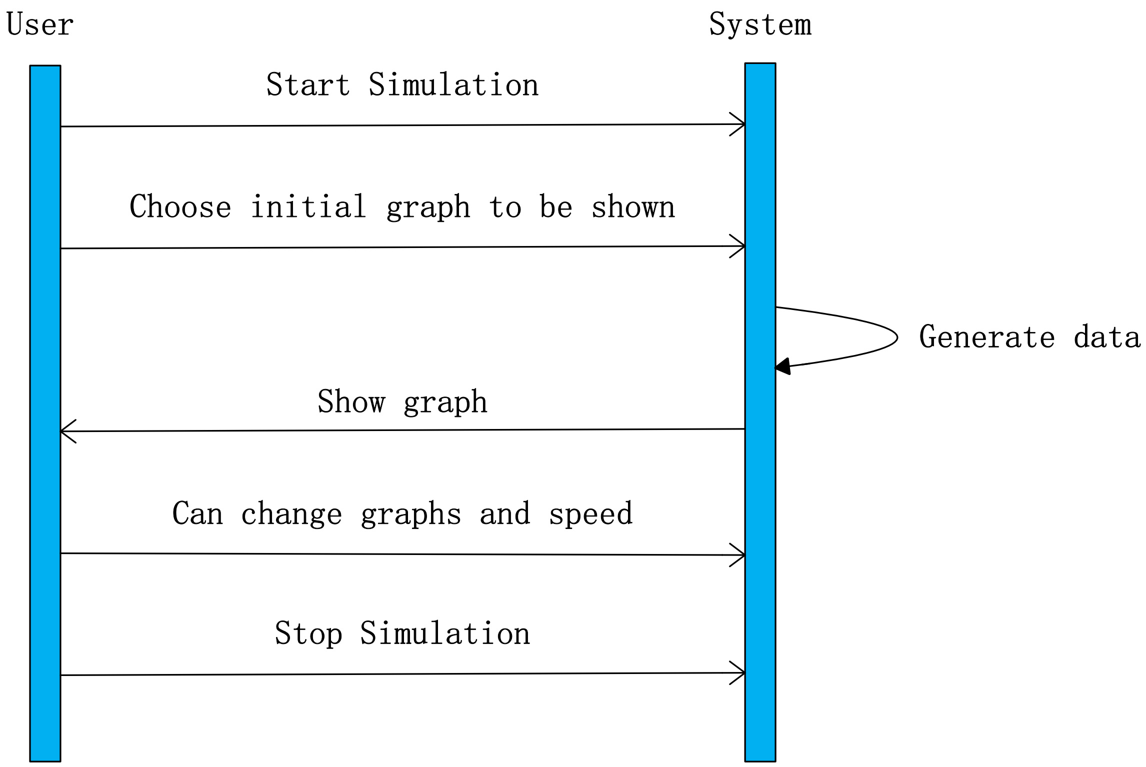


Fig 4 RunSimulation

4. User Interface Specification

Effort Estimation Using Use Case Points

Standard Equations:

Duration=UCP+PF

UCP= UUCP*TCF*ECF

UUCP=UAW+UUCW

Definition

Duration: the whole hours we need for a project

UCP: Use Case Point

PF: Productivity Factor

UUCP:Unadjusted UCP

TCF:Technical Complexity Factor

ECF: Environmental Complexity Factor

UAW:Unadjusted Actor Weight

UUCW: Unadjusted Use Case Weight

1. UAW:

Simply	1
Average	2
Complex	3

User actor	3
System	3

So our UAW = 6

2. UUCW

Simple	5
Average	10
Complex	15

UC-1: Set Initial Condition	5
UC-2: Run Simulation	15

UC-3: Change Graphs	15
UC-4: Change Speed	10
UC-5: Stop Simulation	5
UC-6: Show past Simulation	15

So our UUCW = 65

3. TCF

TCF = constant1 + constant2*calculated factor

Constant1 = 0.6 (according to the textbook)

Constant2 = 0.01 (according to the text book)

Technical Factor	Description	Weight	Perceived complexity	Calculated Factor
T1	Distributed system	2	0	0
T2	Performance objective	2	5	10
T3	End user efficiency	1	4	4
T4	Complex internal processing	1	5	5
T5	Reusable design or code	1	2	2
T6	Easy to install	0.5	1	0.5
T7	Easy to use	0.5	4	2
T8	Portable	2	0	0
T9	Easy to change	1	3	3
T10	Concurrent use	1	0	0
T11	Special Security Feature	1	0	0
T12	Provide direct access to 3rd party	1	0	0
T13	Special User Training	1	0	0

So the calculated factor = 10+4+5+3+3+2+0.5 = 26.5

Then the **TCF** = 0.6 + 0.01 * 26.5 = 0.865

4.ECF

ECF = Constant1 + (-constant2)*Calculated Factor standard equation

Constant1 = 1.4 (according to the text book)

Constant2= -0.03 (according to the text book)

The explanation of the impact:

0: no impact

1: strong **negative** impact

3: average impact

5: strong **positive** impact

Environmental Factor	Description	Weight	Perceived impact	Calculated factor
E1	Familiar with the development process	1.5	1	1.5
E2	Application problem experience	0.5	1	0.5
E3	Paradigm Experience	1	5	5
E4	Lead analyst capability	0.5	3	1.5
E5	Motivation	1	3	3
E6	Stable Requirement	2	2	4
E7	Part-time staff	-1	5	-5
E8	Difficult programming language	-1	3	-3

So the calculated factor = 1.5+0.5+5+1.5+3+4-5-3=7.5

Then the **ECF** = 1.4 **-** 7.5*0.03 = 1.175

5.PF

Although the best solution for estimating the Productivity Factor is to calculate our organization's own historical average from past project, our team is the first time to cooperate and the ability of team members are different. Considering that all of our team members have little experience of software design, we define the PF of us a much higher number : **29**

6. Duration

UUCP = UAW + UUCW = 6 +65 = 71

UCP = UUCP * TCF * ECF = 71 * 0.865 * 1.175 = 72.163 **approximately 72 use case point**

Duration = UCP * PF = 72 * 29 = 2088hours

5. Domain Analysis

5.1 Domain Model

To build a wholly detailed domain model we need to fully review all the use cases and requirement to find out the inner relations between different use cases and the responsibility holders to realize each use cases, that is, the so called concepts. And then we can get the corresponding attributes and associations later.

5.1.1 Concept definition:

Personally, we view the responsibility doer as a concept, that is, a section in program to realize a function that can, eventually, work coordinately to complete the whole use cases. Thus, a draft of how-to-work graph is made and then name the concepts one by one.

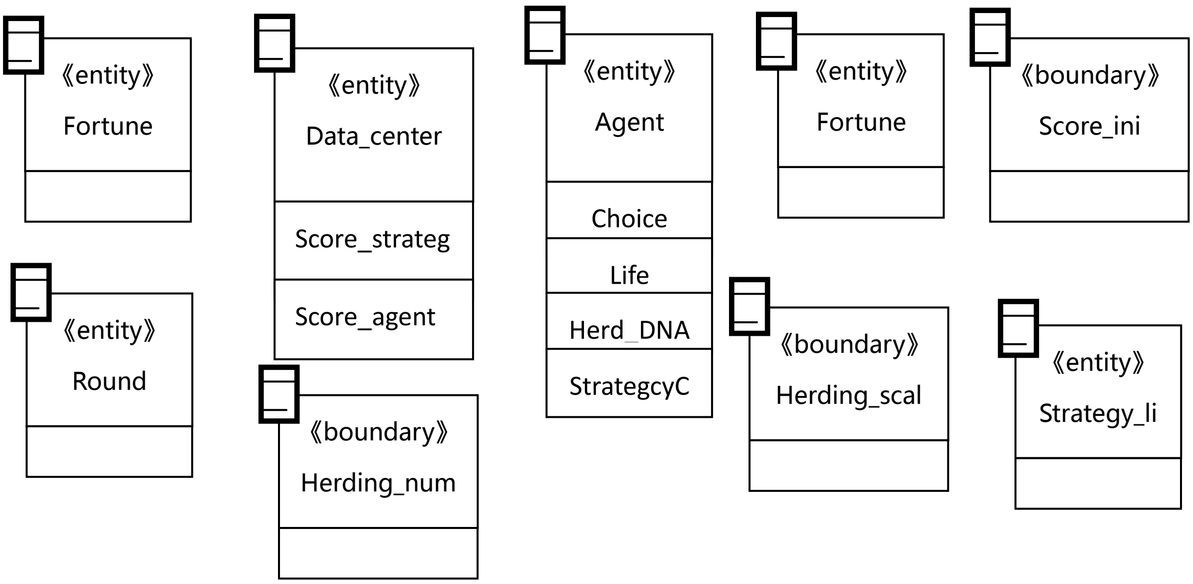
During making the graph, we firstly divide the actors into non-human and human actors and then according to what they act to draft the boundary concepts. As in this case, the only actor is the user, and he or she, just sets the initial parameters and let the software to simulate the model. For the initial parameters needed to set, we look through the use case and compose scenario, agent_num, round, herding_num, lifeopt……

1. Boundary concepts

Responsibility Description	Type	Concept Name
Background or situation choice for users if he or she does not like to set the parameters or certain models user would like to see.	K	Scenario
Container for user's choice of numbers of agents participating in the game	K	Agent_num
Verify whether the user type in valid numbers of agents number within certain limits	D	Agent_num_checker
Container for user's choice of how many rounds it would simulate	K	Round
Verify whether the user type in valid numbers of round times within certain limits	D	Round-checker
Container for how many groups the herding effect would cause if the user choose the option "herding"	K	Herding_num
Verify whether the user type in valid numbers of herding	D	Herding_num_checker

number within certain limits		
Container for how many agents a group would include if the user choose the option “herding”	K	Herding_scale
Verify whether the user type in valid numbers of herding scale within certain limits	D	Herding_scale_checker
Container for whether the user choose the option “Life duration”	K	Life_opt
Container for the initial score of every agent at the very beginning of the game	K	Score_init
Verify whether agents_num is equal or bigger than herding_scale*herding_num	D	Parameter_checker
Stop the simulation immediately	D	Sim_stopper
Switch to the past simulation graph	D	Past_sim_viewer

The property of these concepts includes the types, namely, the K type or the D type as shown on the domain model graph later, the “smile” or the “document”symbol tagged on each concept. From the definition on the textbook, Professor Marsic compares K and D to things and workers: Workers get assigned mainly doing responsibilities, while things get assigned mainly knowing responsibilities. The following are the concept diagram divided by K or D while K symbolized by document and D symbolized by smile.



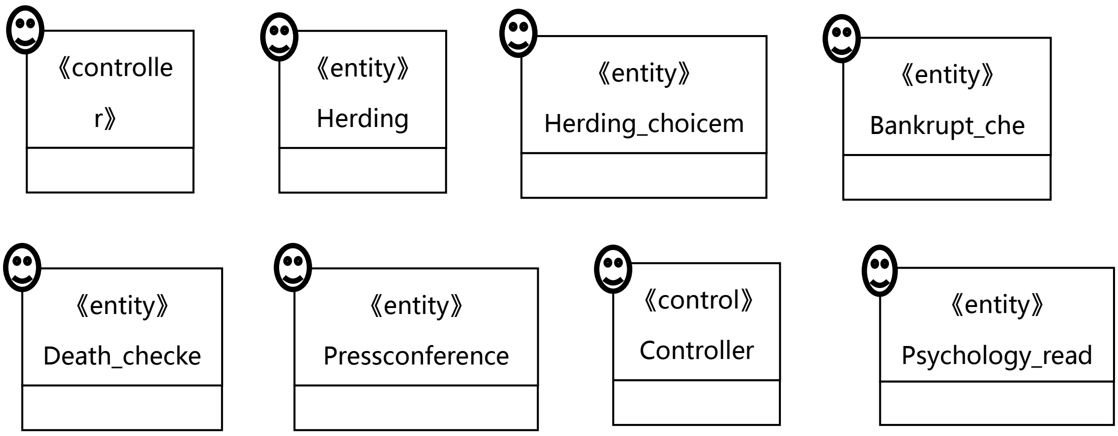


Fig 5 concept diagram

2. Internal concepts

As a matter of fact, the core part of our software mainly lies in our internal concepts; the boundary concepts, mostly take the responsibility of accepting initial parameters and setting scenarios.

At meantime, the types of concepts are mainly D type as inside the software concepts need to communicate and coordinate with other concepts to fulfill the overall use cases eventually.

Responsibility Description	Type	Concept Name
Container for total score of every single agent	K	Fortune
Verify whether the score of any agent be equal to 0 after each simulation	D	Bankrupt_checker
Container for the score of each strategy and each agent	K	Data_center
Update the score of each agent and strategy after every round of simulation	D	Data_updater
Reset the score and strategy of certain agents if they are checked bankrupted or checked death	D	Life_maker
Container for all strategies each agent may equip with for making decisions	K	Strategy_lib
Container for all the statistics of each agent, including choice,life_duration,strategy,herd_DNA	K	Agent
Making the choice of each agent and feed the choice back to Agent	D	Choice_maker
Form the groups of agents willing to herd	D	Herding
Make a uniform choice of a herd	D	Herd_choicemaker
Check if round number has met life_duration of each agent	D	Death_checker

Find out the top 3 agents ranking in score and broadcast their strategy at that round	D	Pressconference
Figure out the probability of each strategy an agent would like to choose the next round	D	Psycology_reader

There is one thing we need to notify that the concept “Agent”associates with some sub-concepts as life, choice, strategy and herd_DNA as mentioned in the attributes.

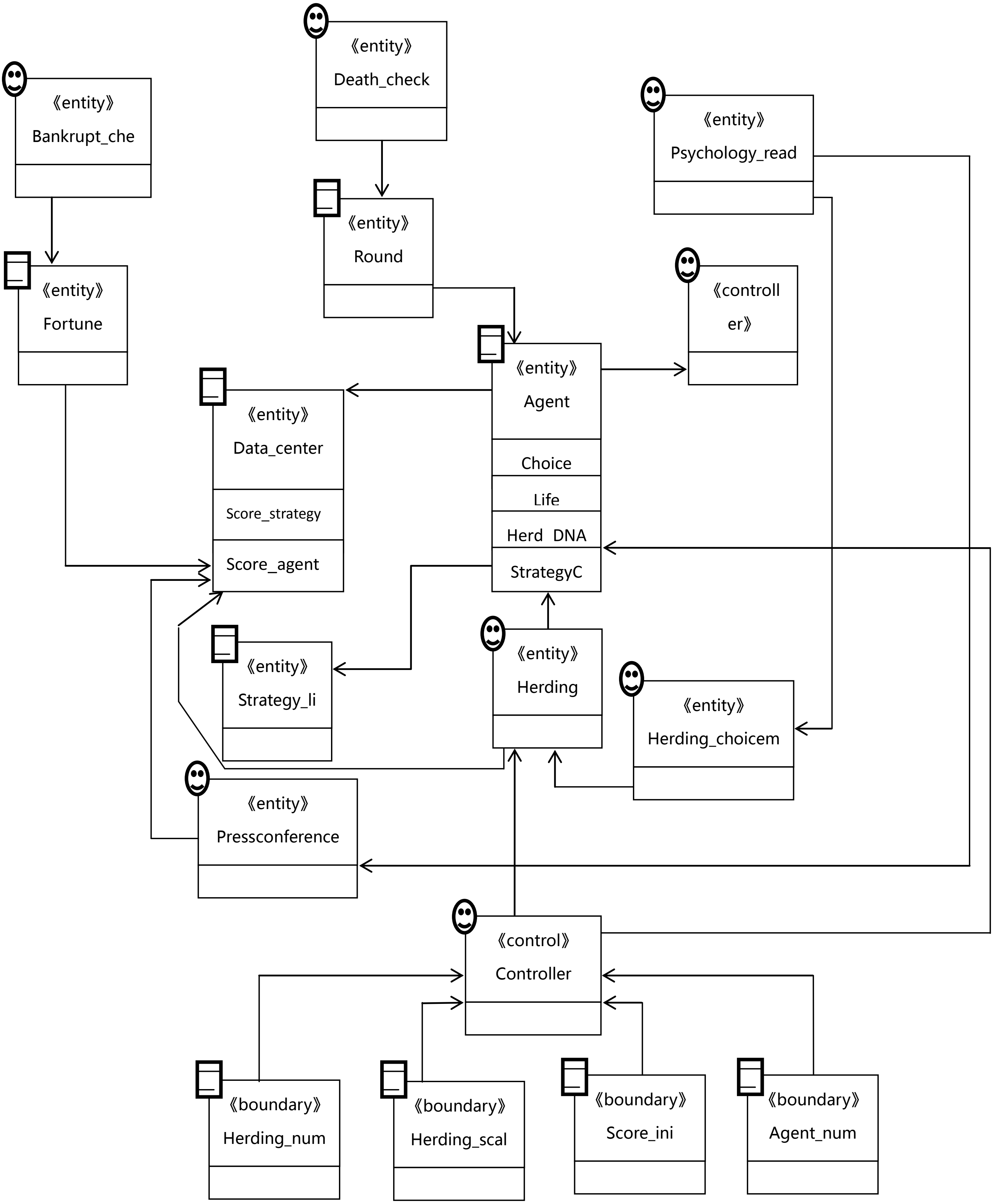
5.2 Association definition

Associations with different concepts are mentioned below. These arrows indicate the relationship between each concept and mainly for conveying information and saving related information. As this software has a very obvious boundary for system and user, we mainly focus on how it operates in the inner side. And one thing is of great significance that there is strictly defined sequence of doing association especially for lots of associations on one concept. Let’s take the example of concept Agents: firstly, after each round, Agent will acquire the updated information from Data_center and get the scores of strategies an agent owns. Then, for non_herding agents, they will draw the conclusion by the scores of strategies with the help of psychology_reader, so that’s the association between psycology_reader. And for those have DNA of herding, they would first get the strategy by the advancer of the herding and the broadcasting of the top3 giants, so that’s the association with herd_choicemaker and conference.

Concept pair	Association description	Association name
Fortune↔Death_checker	Death_checker passes requests to Fortune and receives back each agent’s total score	Conveys requests
Fortune↔Data_center	Fortune passes requests to Data_center and receives back and save scores of each agent after a round is done	Requests save
Conferencepress↔ Data_center	Conferencepress passes requests to Data_center and receives back overall scores of each strategy after each round	Conveys requests
Agent↔Data_center	Agent passes requests to Data_center and receives back and save scores of strategy each agent owns	Requests save
Herd_choicemaker ↔ Data_center	Herd_choicemaker passes requests to Data_center and receives back the strategy with highest score in the very herding	Convey requests
Strategy_lib↔Agen	Agent passes requests to strategy_lib at first round	Requests save

t	and receives back N random strategies for every agent.	
Simulator↔Agent	Simulator passed requests to Agent and receives back choice and strategy at the beginning of every round	Convey requests
Data_center↔Simulator	Data_center passes requests to Simulator and receives and save scores of that round of each agent and the strategy used by that agent	Requests save
Death_checker↔Agent	Death_checker passes request to Agent and receives back and save lifeduration of each agent	Requests save
Death_checker↔Round	Death_checker passes request to Agent and receives back how many rounds has processed	Convey request
Herding↔Agent	Herding passes request to Agent and receives back how many agents have DNA to herd	Convey requests
Data_center↔Agent	Agent passes request to Data_center and receives back and save each score of strategy he or she owns	Requests save
Herd_choicemaker ↔Agent	Herd_choicemaker passes request to Agent and receives back the sterategy of that agent at the next coming round	Convey requests
Herd_choicemaker ↔ Psycology_reader	Herd_choicemaker passes information to Phycology_reader of each agent in that herd the strategy of the advancer	Requests save
Psycology_reader ↔Agent	Phycology_reader passes request to each agent and receives back strategies and its scores of that agent	Convey requests
Choice_maker↔Agent	Choice_maker passes final choice to each agent let it save	Requests save
Conferencepress↔ Psycology_reader	Conferencepress passes strategies of top3 agents to each Phycology_reader	Requests save
Herding↔Herdign_choicemaker	Herding_choicemaker passes request to Herding and receives back herding information and save it	Request save

The following diagram is the domain model diagram:


《entity》
Bankrupt_che
《entity》
Death_check
《entity》
Psychology_read
《entity》
Fortune
《entity》
Round
《controll
er》
《entity》
Agent
Choice
Life
Herd DNA
StrategyC
《entity》
Data_center
Score_strategy
Score_agent
《entity》
Strategy_li
《entity》
Herding
《entity》
Herding_choicem
《entity》
Pressconference
《control》
Controller
《boundary》
Herding_num
《boundary》
Herding_scal
《boundary》
Score_ini
《boundary》
Agent_num

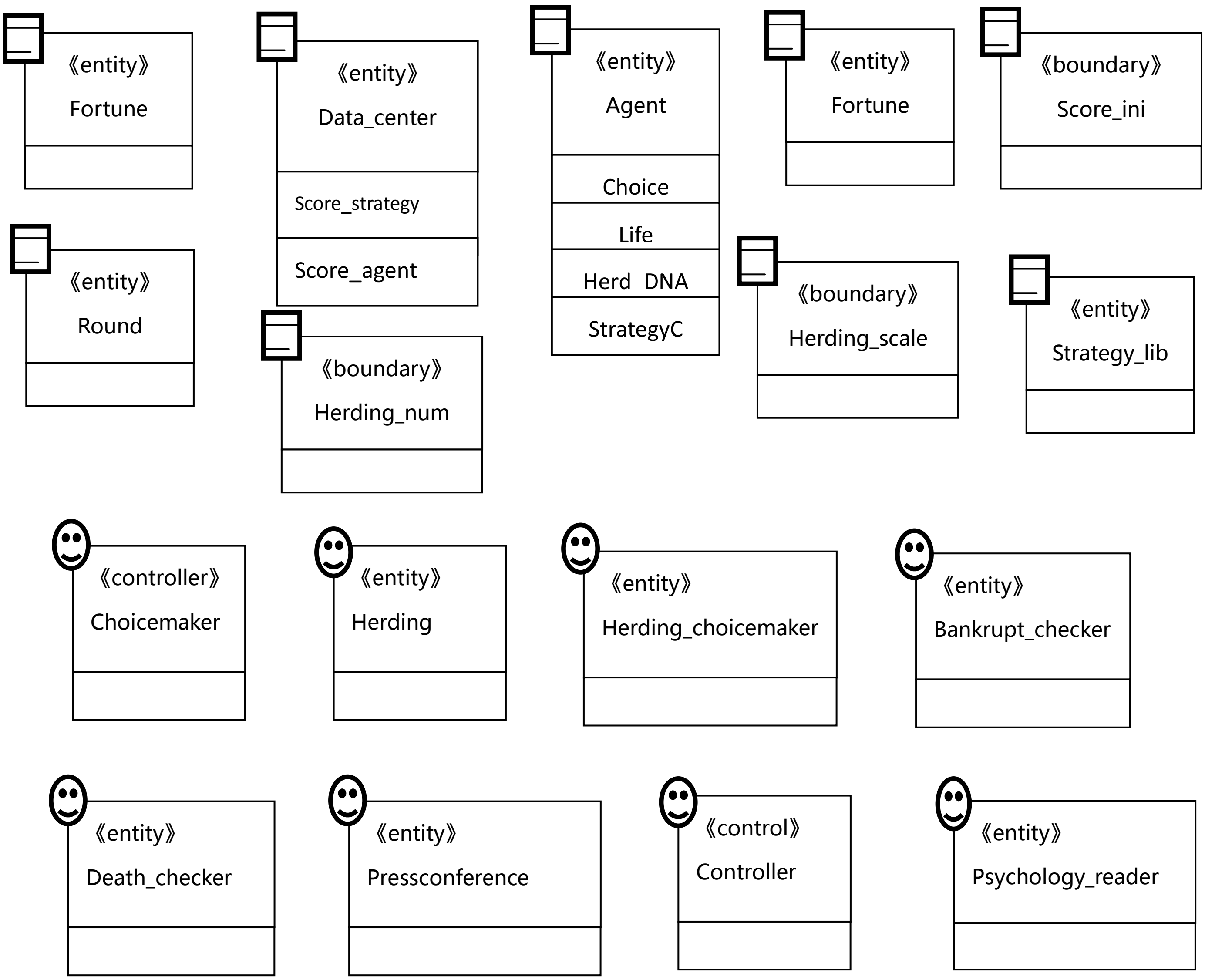


Fig 6 domain model diagram

5.1.3 Attribute definitions

The Minority Game has 7 concept attributs: Agent, poorcheck, container, herdconstitution, herdchoice, database and input variables

Each agent will choose their own strategy depend on their memory or the group advancer's suggestion. Furthermore, agents who have Herd DNA will set up groups and share the strategies. However, they are also constrained by life duration, which means how long they can exist in this game system. Thus, the concept of Agent has attributes of Choice, Life Duration, Herd DNA and Strategy_C.

After several rounds, agents whose scores are less than 0, they will excluded the system. So, Death_checker has attribute of Exclude

System needs a container to record and contain the scores of each strategy and agent. Poorcheck and herdchoice will call this data. So NumS_a and NumS_s are attributes for Container.

When we select herding button, we need to determine the number of groups and group scale. Furthermore, we also need to determine which strategy each group will choose. In this case, Herding will have attributes of Herding_scale , Herding_num, and Advcancer.

The system also needs a database to store some initial data, such as strategies, life model and herding model. User should input initial information such as numbers of agents and rounds. So Database and InputVariables also have some attributes.

Attribute Definition shows below.

Concept	Attribute	Attribute Description
Agent	Choice(memory)	The strategy a agent will choose each round
	Life Duration	How long they can exist in this game
	Herd DNA	Agents who will hold together with others
	Strategy_C	The number of Strategies contained in agent's memory

Death_checker	Exclude	Agents whose scores are less than 0
DataCenter	NumS_a	Record and contain the score of each agent
	NumS_s	Record and contain the score of each strategy
Herding	Herding_scale	The total number of groups
	Herding_num	The number of agents each group have
	Advancer	Agents who have the highest score in a group
Database	Strategies	The initial strategies storage
	MortalityType	contains enumerated type of mortality model being using
Inputs Variables	Num_a	The number of agents participating in the game.
	Num_r	The number of rounds played before the end of the game.

5.1.4Traceability matrix

The traceability matrix for is shown in Figure-7. It shows how the system use cases map to the domain concepts

UC	PW (1-5)	Scenario	Agent num	Agent num checke	Round	Round-checker	Herding num	Past sim viewer	Herding scale	Sim stopper	Life opt	Score init	Fortune	Data center	Life maker	Strategy lib	Agent	Choice maker	Herding	Herd choicemaker	Death checker	Pressconference	Phycology_reader
UC1	5	X	X	X	X	X	X		X		X	X											
UC2	5												X	X	X	X	X	X	X	X	X	X	X
UC3	4	X																					
UC4	3		X	X	X	X	X		X			X											
UC5	1									X													
UC6	2							X															

Fig 7 Traceability matrix

5.2 System Operation Contracts

1, What are the Sections of a Contract

Operation:	Name of operation and parameter
Cross reference	Use cases this operation can occur within
Preconditions	Note worthy assumptions about the state of the system or objects in the Domain Model before execution of the operation. These are non-trivial assumptions the reader should be told.
Postconditions	This is the most important section. The state of objects in the Domain Model after completion of the operation.

2, Our System Operation Contracts

Operation:	Agent_num_checker
Cross reference	UC-1
Preconditions	User inputs the number of agents
Postconditions	Data valid, system runs

Operation:	Bankrupt_checker
Cross reference	UC-2
Preconditions	1,Data-center contain each agent's score successfully 2,Succeed in receiving data from Data-center
Postconditions	1, Agents exclude from system 2, New agents being added in

Operation:	Herd_choicemaker

Cross reference	UC-2
Preconditions	1, Agents hold together with others 2, Data-center contain valid data
Postconditions	A strategy is chosen for the whole group

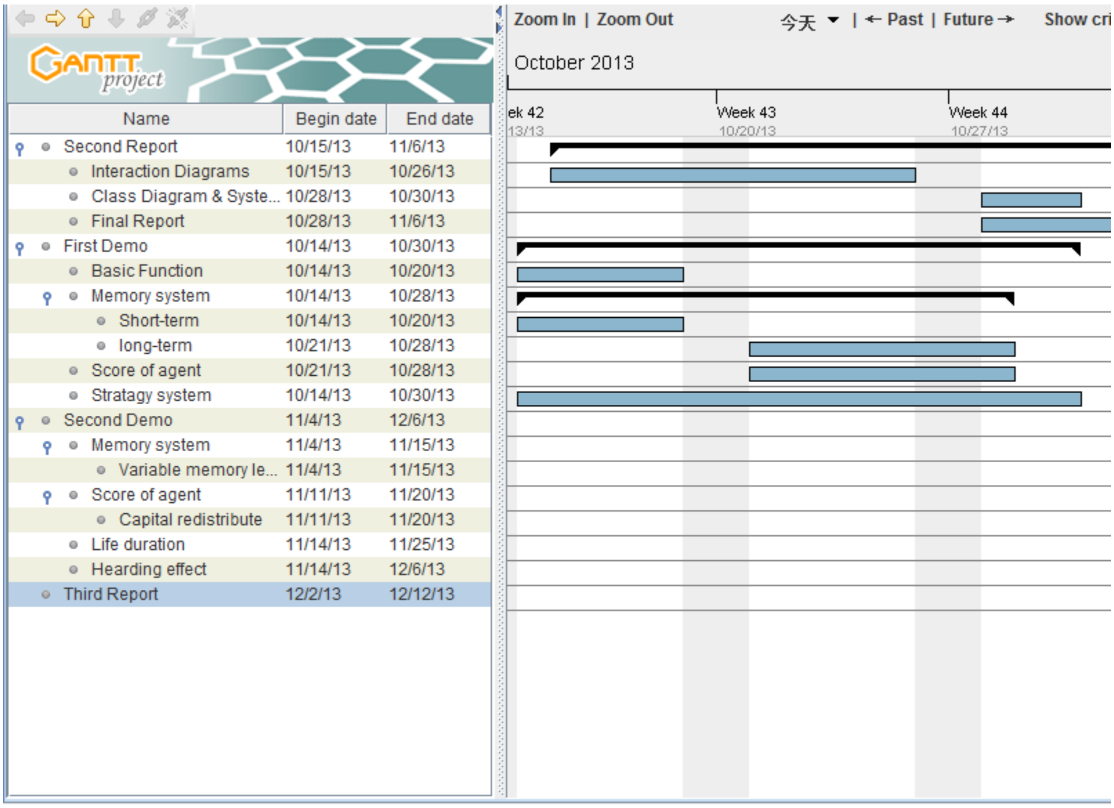
Operation:	**Past_sim_viewer**
Cross reference	UC-6
Preconditions	1, System runs successfully 2, All data is true
Postconditions	Analysis graph and data

Operation:	**Sim_stopper**
Cross reference	UC-5
Preconditions	system is running
Postconditions	The game is finished

Operation:	**Agent_maker**
Cross reference	UC-3
Preconditions	All data valid Change initial conditions, such as Agent_Num, Life_duration
Postconditions	System runs with another speed

6. Plan of Work

There will be two demos of this project. The first one will be presented in November which will show the basic function of the system. The second one will be presented in December which will add vivid elements to the system, like situation option, herding effect of agents, broadcast， etc. And there will be two reports in the future. The first one will describe the design of our system and the second one will be the final report. The plan diagram will sow below.



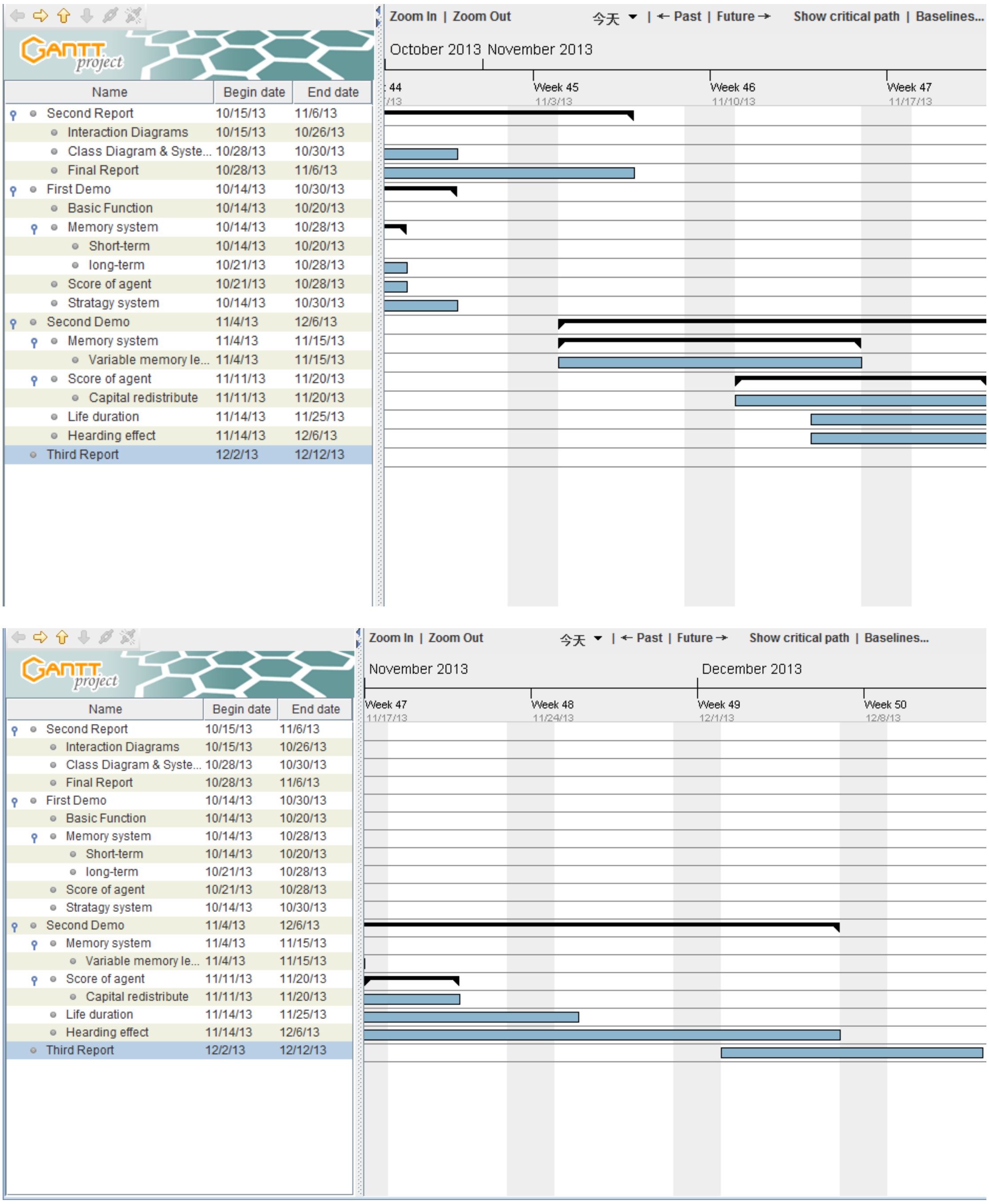


Fig 8 plan diagram

7. Reference

1. Software Engineering by Ivan Marsic

2. El Farol Bar Problem - http://en.wikipedia.org/wiki/El_Farol_Bar_problem

3. El Farol Bar Problem and the Minority game Project Description

-http://www.ece.rutgers.edu/~marsic/books/SE/projects/MinorityGame/

4. Project #3, group #4, Spring 2011 -

http://www.ece.rutgers.edu/~marsic/books/SE/projects/MinorityGame/2011-g4-report3.pdf

5. Project #3, group #7, spring 2012 -

http://www.ece.rutgers.edu/~marsic/books/SE/projects/MinorityGame/2012-g7-report3.pdf

6. Project #3, group #10, spring 2012 -

http://www.ece.rutgers.edu/~marsic/books/SE/projects/MinorityGame/2011-g4-report3.pdf