Object Oriented Design and implementation of A Web-enabled Beer Game for Illustrating the Bullwhip Effect in Supply Chains

N. R. Srinivasa Raghavan†, Bishal B. Shreshtha and S. V. Rajeev

Department of Management Studies

Indian Institute of Science

Bangalore 560 012, INDIA

Email: raghavan@mgmt.iisc.ernet.in

ABSTRACT

An important observation in supply chain management, popularly known as the **bullwhip effect**, suggests that demand variability increases as one moves up the supply chain. Empirical evidence suggests that the orders placed by a retailer tend to be much more variable than the customer demand seen by that retailer. This increase in variability propagates up the supply chain, distorting the pattern of orders received by distributors, manufacturers and suppliers. In this connection, the (popular) beer game [8] aptly illustrates the bullwhip effect. Various versions of this game have already come up starting from the original board-and-card version to computerized versions. Web-enabled versions allow for multiple players as well as a single player to play the game online. The software that has been designed and developed as part of this paper is one in that direction. Our analysis and design is based on sound principles of object orientation and the implementation is novel on several counts.

Key Words: Bullwhip effect, Beer Game, Object Oriented Design, UML

1.0 Introduction

The word 'game' conveys an entertainment content, which is one of the important characteristics of any game. But, the fact that games have been put to more serious pursuits is seen in the traditional use of chess in war planning.

† Author for correspondence

The principle underlying the use of games in training is that participants learn better through doing than through reading, hearing or observing. This principle of Discovery Learning has become almost routine in teaching and training. This method of training, with its focus on behavioral and attitudinal change, encourages active learning. Management games reveal a scientific tradition and a social science tradition, but predating them both is a pragmatic tradition of experiment and practice. More recently ideas have been drawn from an entertainment tradition. A common result is that anyone who arrives in the field of 'management games' thinks that their route towards it was somehow the 'natural' one [6].

1.2 Objective of Games

The purpose of games, regardless of their origin, is the same and can be classified as

- Promoting Knowledge and Skill
- Increasing Behavioral Skill
- Giving Pleasure and Promoting Teamwork

We briefly give more reasons for games to be applied in practice.

Reinforce Instruction - games and simulations about running an imaginary business are used to help players discover whether they have grasped, and are able to apply, what they have learnt theoretically.

Help players see the broader picture (integrate functional viewpoints) - Most business education is based on a functional/department split and many managers rise in an organization through a single specialized channel. Development practitioners therefore find a widespread need for a device that will 'Show them how it all fits together'.

To improve decision-making - the information presented to the players of a management game is often structured to create a 'time window' between making decisions and discovering the result. This makes it possible to review the decision clinically before the results are known. It is advantageous because outcomes do not always prove that a decision was 'good' or 'bad'. Both in real life and in games outcomes may be affected by chance, or by unforeseeable factors, so that a theoretically good decision produces a poor result (or the reverse).

1.3 The Beer Game

The Beer game is one such management game (see [10, 11, 14]), which falls into the functional type of games, focusing more on the Operations Management part of the wide spectrum of the business. There is a chain of people: retailer, wholesaler, distributor and brewer or factory, each of whom passes orders for cases of beer to the next person in the chain. The retailer gets orders from the `customer', who is external to the game. Each round corresponds to a week; there are typically 52 rounds. In any round each player must fill all current orders, place orders with his own supplier, and try to minimize back orders and inventory. There is a penalty for cases held as inventory and for having back orders unfilled. Players may not discuss tactics with each other; they merely place orders and receive cases of beer. The aim is to minimize the total cost for all players for the whole period of the game. The total cost includes inventory holding costs and the backorder cost, with either one being more than the other, thus requiring the player to make a trade off between overstocking and falling short of goods while making his decision on shipping and ordering. This game helps business people to better understand the system dynamics involved and to appreciate the bullwhip effect that creeps into the supply chain.

1.4 The Bullwhip Effect

The bullwhip effect basically portrays increasing variability in demand seen by each facility along the upstream direction of the supply chain from the retailer's end to the factory end. The causes for this effect are

- Demand Forecasting
- Lead Times
- Batch Ordering
- Supply Shortages
- Price Variations

The first aspect, i.e., *demand forecasting* has been covered in this implementation of the beer game using two forecasting methods – Moving Average and Exponential Smoothing (see chapter on Forecasting in Kostas [7]). It is at the player's disposal to choose whichever method he wants. The bullwhip effect is caused, in part, by the need to forecast the mean and standard deviation of demand. Although this insight seems to suggest that we

should eliminate demand forecasting, this clearly is not the answer. Forecasting, of course, provides valuable information to each stage of the supply chain. Without demand forecasting, we would be forced to simply "guess" the expected demand in each period.

1.5 The Objective of our work

The objective of the paper is to come up with a web-enabled version of the beer game designed using object oriented techniques, which allows a single player to play the game at any point of time. This game will help management students to understand the bullwhip effect and its implications on the supply chain, on how costs increase due to unavailability of correct information on the real customer demand. This implementation of the game will also provide an insight as to how the exponential smoothing method of forecasting leads to a greater increase in the variability than the moving average method, under the assumption that the demand distributions are independent and identical and that both the forecasts have the same variance of the forecast errors.

2.0 Relevant Literature

An important observation in supply chain management, popularly known as the **bullwhip effect**, suggests that demand variability increases as one moves up the supply chain. The bullwhip effect is a major concern for many manufacturers, distributors and retailers because the increased variability in the order process (i) requires each facility to *increase its safety stock* in order to maintain a given service level, (ii) leads to *increased costs due to overstocking* throughout the system, and (iii) can lead to an *inefficient use of resources*, such as labor and transportation, due to the fact that it is not clear whether resources should be planned based on the average order received by the facility or based on the maximum order.

The five main causes of the bullwhip effect, as pointed out by Lee, Padmanabhan and Whang [2 ,3] are demand forecasting, lead times, Batch ordering, supply shortages and price variations. Chen, Drezner, Ryan and Simchi-Levi [1] consider the impact of demand forecasting and lead times on the bullwhip effect. The authors present a lower bound for the amplification factor (variance of quantity ordered to the variance of actual customer demand) as a quadratic function in the variable L/p where L is the lead time and p is the parameter for moving average

forecast; alternatively, it is another non-linear function of order two in the variables L and α where α is the parameter for exponential smoothing forecast. System dynamics based models for illustrating the bullwhip effect are presented in Wickner et. al. in [4, 5]. We refer the reader to the above papers (and the references therein) for an elaborate analytical treatment of the bullwhip effect.

Regarding games that have been designed to demonstrate this effect, we are aware of "hardware" versions of the beer game, and some standalone software versions available for free/paid download from various sites. (See for instance, [12, 13, 16]). We did not find the details regarding the software architecture from any of the sites available on the Internet. Also, in our implementation, we provide for rich functionality not available in any of the sites that we are aware of, by (i) including in the game, the impact of the forecasting method on bullwhip effect, and (ii) summarizing the status of the game by plotting the graphs and displaying them to the user via the web itself. Our tool can be used for education purposes, be it in B-schools, or in the corporate sector, where supply chain decision makers need to appreciate the facets of inventory control. Because our implementation is in Java technologies, it can be seamlessly integrated across various hardware and operating system platforms.

A software implementation of the "Hulia Game" developed by Gilad Ravid and Sheizaf Rafaeli [15] is a simulation game developed using java applets. Our implementation is using java servlets that are more efficient. The website of MA-systems[16] provides a very good GUI for the beergame but does not provide the design and implementation details. The paper by Biswas and Narahari [17] is an illustration of applying object oriented modeling techniques for supply chain decision making.

A major contribution of our work is that we have used object oriented analysis, design and programming to implement a web enabled game. The architecture for the same is presented in the form of UML diagrams (including Use case and sequence diagrams) making it easier for even novice implementers to replicate our implementation. We now provide a brief description of the game that we have designed and implemented.

2.2 The Beer Game

We resort to the description of the game which is available at [14]. The purpose of this game is to introduce to the participants the concept of supply chain and the effects decisions along the chain have on inventory levels and costs. The reader may refer to the details of the game given at the website. For brevity, we do not mention it in this paper.

3.0 System Design

For designing the system, we used Rational Rose to create UML diagrams (see [18]) which were later on used in the implementation phase. The diagrams that were generated were Usecase diagrams, Activity diagrams and sequence diagrams for the various usecases.

3.1 Usecase Diagrams

The main Usecase diagram is shown Figure 1.

3.1.1 Flow of events for the *Login* Usecase

3.1.1.1 Preconditions

There are no preconditions to this use case.

3.1.1.2 Main Flow

This use case begins when the player logs on to the site and prompts him/her to enter the name, the role s/he wants to play, as well as set other prior data like choosing the demand forecast method for the entire chain, inputting a value for the smoothing constant (if the exponential method is chosen), setting the inventory holding cost as well as the back order cost and finally choosing whether to use the demand forecast method to help him/her place orders.

3.1.1.3 Sub flows

S-1: Enter name and role

The player enters his/her name and role that s/he would like to play and the system creates a separate profile for the player.

S-2: Choose Demand Forecast method

The player is given a choice between two methods – moving average and exponential smoothing. If the latter is chosen then s/he must enter a value for the smoothing constant as well. The system then notes this down in the player's profile, which will be used throughout the tenure of the game.

S-3: Set the inventory and back order cost

Here the player has been given the freedom of choosing the cost parameter values, so as to make the necessary trade-off between shortage and overage. The system uses these costs throughout to calculate the costs of each facility in the chain as well as the total cost.

S-4: Choose the option of using demand forecasting for helping to place orders

If the player chooses this option then the system automatically informs him/her about the expected demand during lead-time, which will help the player to get a fair idea of how much s/he should order.

3.1.1.4 Alternative Flows

E-1: If the player enters a name and role that has already been taken up by someone else or by him/herself, s/he is required to enter a new name (if he wants to play the same role) or a different role (if he wants to use the same name).

E-2: If the player enters characters or negative numbers in the cost fields then s/he is required to enter the appropriate data (positive numbers).

E-3: An integer value or a number greater than 1 or lesser than 0 or alphabets for alpha is entered. The system sends an alert message. The player is required to enter the correct value (a number between 0 and 1).

3.1.2 Flow of events for the Fill Decision Form Usecase

3.1.2.1 Preconditions

The Login event should be a success and should precede this Usecase. Only after the player has been successfully logged in will the system present a decision form to him/her to fill up regarding ordering and shipping quantities for each period.

3.1.2.2 Main Flow

This use case begins after the player has successfully logged on to the system and his/her profile has been created. Here the player is required to make decisions on how much to ship to the immediate downstream facility in the supply chain and how much to order from the immediate upstream facility in each period of operation. S/He is presented with a screen that shows the current status regarding inventory level, back order level, the corresponding costs incurred, the current demand, and also how much of goods is due in one week and in two weeks. Based solely on this information and the forecast for demand during lead-time (if the player chooses to use demand forecasting), the player has to make a decision on how much to ship and how much to order. After the form is filled the player clicks on submit after which the system takes this information and updates the status of the player.

3.1.2.3 Sub flows

S-1: Enter shipping quantity

The player enters the shipping quantity in this field. This decision is based on the current demand, inventory level and back order level.

S-2: Enter ordering quantity

Here the player enters the amount s/he wants to order from the upstream facility, taking into consideration the forecast for demand during lead-time, and also the goods due. After this decision has been made the player clicks on submit to forward the information to the system.

3.1.2.4 Alternate flows

E-1: If the player enters alphabets or floating-point numbers in the text fields or leaves any of the fields empty, s/he is given an error message. Also if the actor ships more than s/he is required to, an error message is again given.

3.1.3 Flow of events for Update Period Usecase

3.1.3.1 Preconditions

The Fill Decision Form use case should precede this one. Only after the player has filled the decision form and has submitted the same to the system, will the system present an updated status on a different screen to him/her and only then can s/he update the period.

3.1.3.2 Main Flow

This event requires the player to click on update period after s/he has submitted the decision form and has been presented by the system with an updated status for the current period. After clicking on this button, the player is again presented with a decision form for the next period, with all the necessary fields updated like inventory level, goods due in one week, goods due in two weeks as well as the new demand for the period.

3.1.4 Flow of events for View Summary Usecase

3.1.4.1 Preconditions

The Fill Decision Form use case should precede this one. Only after the player has filled the decision form and has submitted the same to the system, will the system present an updated status on a different screen to him and only then can s/he update the period.

3.1.4.2 Main Flow

Here, after the player has been presented with an updated status after making the decisions required of him/her, s/he has the option of viewing the summary of the progress of the game. After clicking on this button, the player is presented with four different tables showing the status of all the four facilities in the supply chain, with the player's table coming first. It shows the total costs incurred by each individual facility as well as by the entire chain.

3.1.4.3 Sub flows

S-1: Continue with the game

If the player chooses this option then an updated-period screen is presented to him/her with the decision form for the corresponding period.

S-2: View Variability

If the player clicks on this button then s/he can see the increasing variability at each facility in the linear chain, which in fact is an illustration of the bullwhip effect. In addition s/he is also provided with an option to view a graphical display of the results.

S-3: Quit Game

The player by clicking on this button confirms that s/he wants to exit the game and on doing so is presented with a goodbye screen.

3.2 More UML Diagrams:

The sequence and activity diagrams for the above cases are presented in figures 2-8. We do not describe them here due to want of space. The diagrams are self explanatory and are drawn for the use cases described in section 3.1.

4.0 Implementation

The implementation of the game requires the creation of a server whose principal task is to access the forms sent online by the clients and send the processed information back to the client. The game has been implemented in Java using the client-server architecture and involves three phases. The first phase involved designing the static pages in Dreamweaver. The second phase involved selecting an appropriate database to which the servlets could communicate. The third phase involved generating algorithms for the servlets – six in all.

Our work is available at the following URL and our pages are best viewed with MS Internet Explorer. The first page that comes up when the player types the following URL (http://sudarshana.mgmt.iisc.ernet.in:8080/enter.html) looks like the one shown in Figure 9. This form takes in all the relevant data required to create a profile of the player. If the player's entered name and role already exist in the database, then s/he gets the Relogin screen on clicking the "Next" button.

Figure 10 shows the a sample set of values for the Variance. Some sample graphs are shown in Figure 11.

4.2 Selection of the Database

We decided on MySQL as the database, since the software and the jdbc driver is available in public domain. Furthermore the implementation does not have any stringent requirements at the database end. It requires the creation of one table for serving the purpose of a log, which keeps the list of the names of all the players who login and also the roles they choose to play, and another table which holds the entire statistics of the player's session. The servlets talk to this database constantly, either sending queries or updating the tables according to the progress of the game.

The table which serves the purpose of the log has been named log and the attributes of the table are shown in Table 1. The table, which keeps the information on the player's status as well as the status of the entire linear chain, will be created dynamically by a servlet after a player successfully logs into the system. The name of the table will be "name_role" where name will be the name of the player and role will denote the role he is taking up to play the game. The attributes are as shown in Table 2. The coding of the game has been done using Java Servlet Programming (see Hunter et. al. [9]). The Java Servlet Development Toolkit is supported by JAVA 1.2 and above. The dynamic files created by the Summary.java Servlet for generating graphical display of results are stored in a temporary directory. To connect to the server site, the client has to enter this address http://sudarshana.mgmt.iisc.ernet.in:8080/enter.html.

Conclusions

We have seen that the bullwhip effect is due, in part, to the need to forecast demand. Although this insight seems to suggest that we should eliminate demand forecasting, this clearly is not the answer. Forecasting, of course, provides valuable information to each stage of the supply chain. Without demand forecasting, we would be forced to simply "guess" the expected demand in each period. Smoother demand forecasts can reduce the bullwhip effect. The magnitude of the increase in variability depends on the forecasting method and also on the demand process. Research has been carried out to show that for the same variance of forecast errors using both the methods, the increase in variability due to the exponential smoothing method is more than the increase in variability caused by the moving average method, which can be seen from our game too. Even though the same variance cannot be ensured for two games using different methods, we can see that the increase in variability is

more for the one using exponential smoothing than that using moving average method of forecasting. Similarly the other managerial insights pointed out above can also be illustrated by playing this game.

This implementation allows only a single player to play the game but its functionality can be extended to provide a facility for multiple users to play the game simultaneously and along the same chain. Also this implementation has only considered demand forecasting for portraying the bullwhip effect. There are other causes too, like, lead time, price variations, batch ordering and supply shortages which can lead to this effect which too can be captured in this game if needed and if given more time for developing the software. Work is underway to incorporate the effect of order batching within the beer game.

Acknowledgements

This work is partly sponsored by the Department of Science and Technology, Government of India, Grant No-SR/FTP/ET-08/2001

Bibliography

1. Chen Frank, Zvi Drezner, Jennifer K.Ryan and David Simchi-Levi, "*The Bullwhip Effect: Managerial Insights on the impact of forecasting and information on variability in a supply chain",* **14** (1998), 419-35.
2. Lee, H., P. Padmanabhan and S. Whang, "The Bullwhip Effect in Supply Chains," *Sloan Management Review*, **38** (1997a), 93-102.
3. Lee, H., P. Padmanabhan and S. Whang, "Information Distortion in a Supply Chain: The Bullwhip Effect," *Management Science*, **43** (1997b), 546-58.
4. Wikner, J., D.R.Towill, M.M Naim, "Industrial Dynamics Simulation Models in the Design of Supply Chains", *International Journal of Physical Distribution & Logistics Management*, Vol. 22 No.**5** (1992), 3-13.
5. Wikner, J., D.R.Towill, M.M Naim, "The System Simplification Approach in Understanding the Dynamic Behavior of a Manufacturing Supply Chain", *Journal of Systems Engineering*, **2** (1992), 164-78.

6. Elgood, Chris "The tradition of Management Games", http://www.*mbagames.com.*

7. Dervitsiotis, Kostas N., "*Operations Management*", McGraw-Hill, 1994.

8. Sterman, John D., "The Beer Distribution Game: An Annotated Bibliography Covering its History and Use in Education and Research", (1992), *http://www.solonline.org/pra/tool/bibl.html.*

9. Hunter, Jason & William Crawford, " *Java Servlet Programming*", O'Reilly & Associates, 1998.

10. Caglar, Deniz, Michael Li, and David Simchi-Levi "The Web Based Beer Game: Demonstrating the Value of Integrated Supply Chain Management", *http://beergame.iems.nwu.edu/guide.htm.*

11. "The Beer Game", *http://www.solonline.org/pra/tool/beer.html.*

12. "The Beer Distribution Game Management Flight Simulator", *http://web.mit.edu/jsterman/www/SDG/MFS/beerMFS.html.*

13. "Online version of the beer game", *http://beergame.mit.edu/.*

14. "Beer Distribution Game*",* http://www.pom.edu/beer.

15. "Multi player internet and java based simulation games", Gilad Ravid and Sheizaf Rafaeli, University of Haifa, Israel, (2002), http://citeseer.nj.nec.com/516223.html.

16. "The Beer Game", http://www.masystem.com/beergame.

17. "Object oriented modeling for decision support in supply chain networks", S. Biswas and Y. Narahari, European Journal of Operations Research, 153-3, (2003), 704-726.

18. "UML Resource Centre", http://www.rational.com/uml/index.jsp.

Tables And Figures

Table 1: The relational table created for keeping log of every player

Field	Type	Null	Key	Default	Extra
name	varchar(30)	YES		NULL	
role	varchar(15)	YES		NULL	

Table 2: The relational table created for every session initiated by players

Field	Type	Null	Key	Default	Extra
Role	varchar(13)	YES		NULL	
Period	int(10) unsigned	YES		NULL	
Inventory	int(10) unsigned	YES		NULL	
Back_Order	int(10) unsigned	YES		NULL	
Delay_1	int(10) unsigned	YES		NULL	
Delay_2	int(10) unsigned	YES		NULL	
Demand	int(10) unsigned	YES		NULL	
Forecast	int(10) unsigned	YES		NULL	
Shipment	int(10) unsigned	YES		NULL	
N_Order	int(10) unsigned	YES		NULL	
Inv_Cost	int(10) unsigned	YES		NULL	
BO_Cost	int(10) unsigned	YES		NULL	
Tot_Cost	int(10) unsigned	YES		NULL	

Fig 1: Main UseCase Diagram

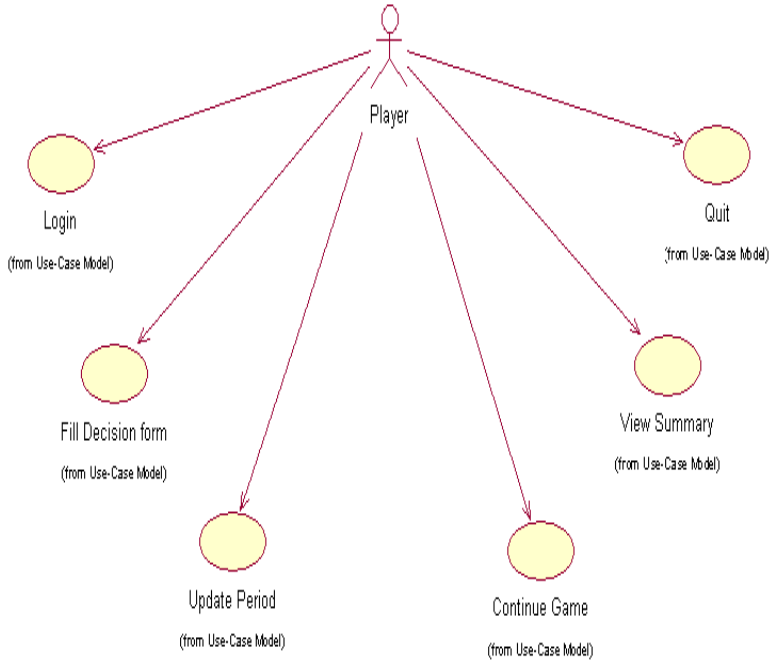
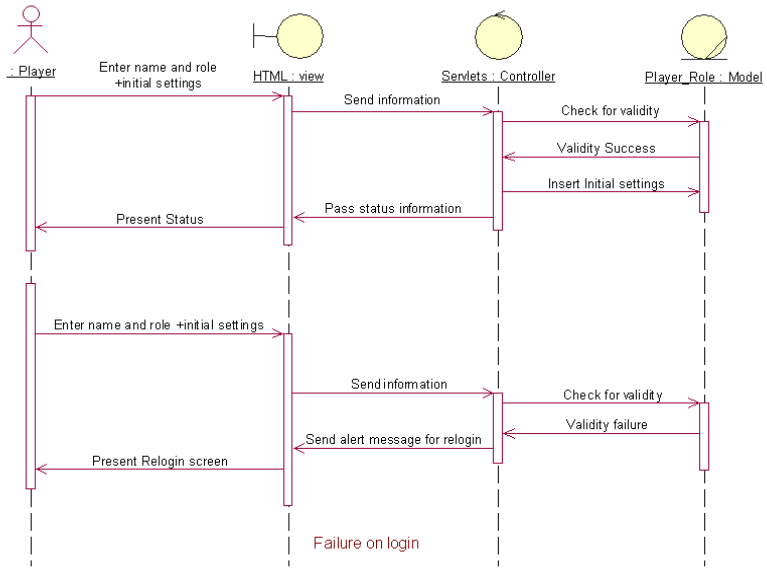
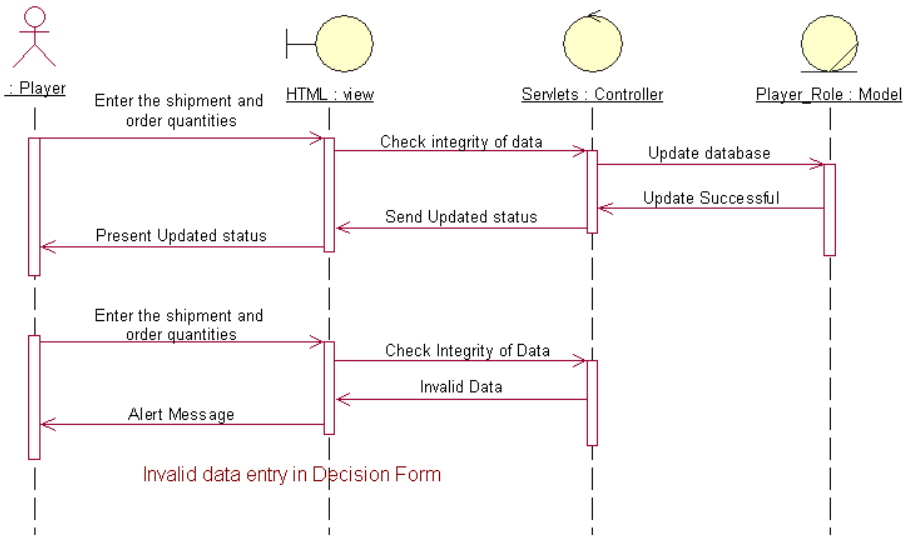


Fig 2: Sequence Diagram for Login Use Case



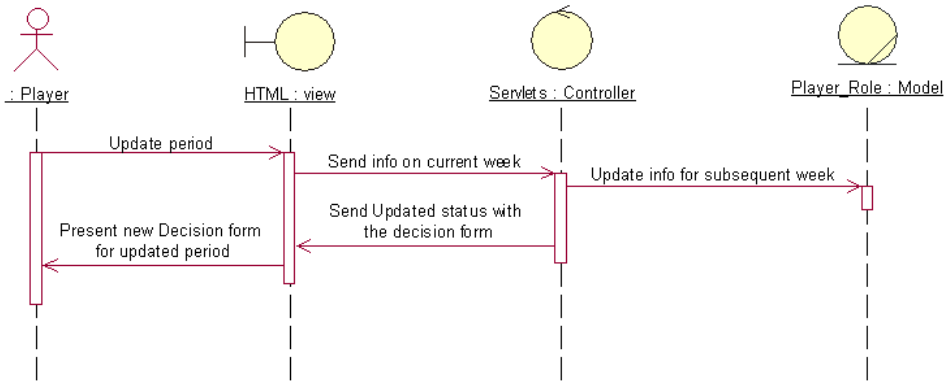
Sequence Diagram for Login Usecase

Fig 3: Fill Decision Form UseCase Diagram



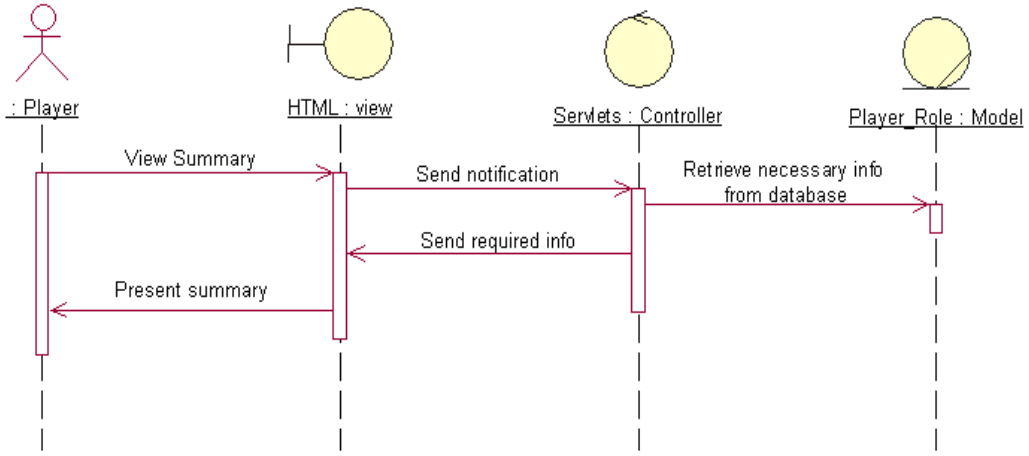
Sequence Diagram for Fill Decision Form usecase

Fig 4: Update UseCase Diagram



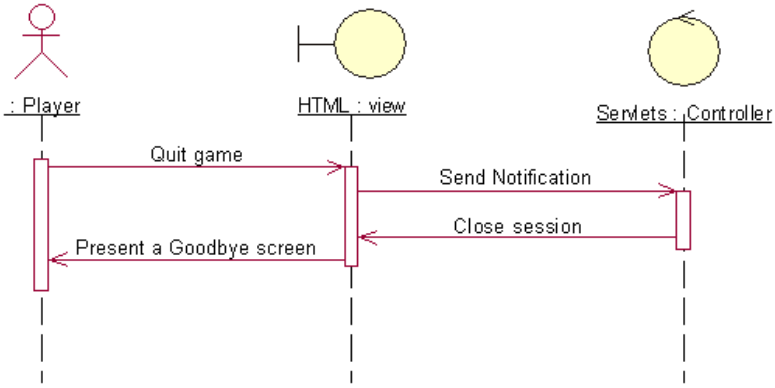
Sequence Diagram for the Update Period Usecase

Fig 5: View Summary UseCase Diagram



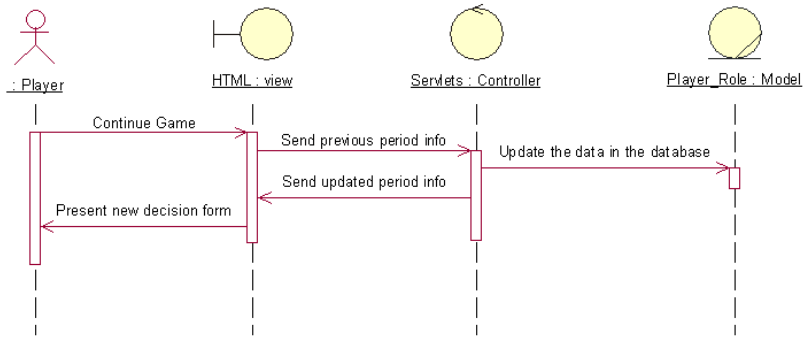
Sequence Diagram for View Summary Usecase

Fig 6: Quit Game UseCase Diagram



Sequence Diagram for the Quit Game Usecase

Fig 7: Continue UseCase Diagram



Sequence Diagram for Continue Game Usecase

Fig 8: Activity Diagram

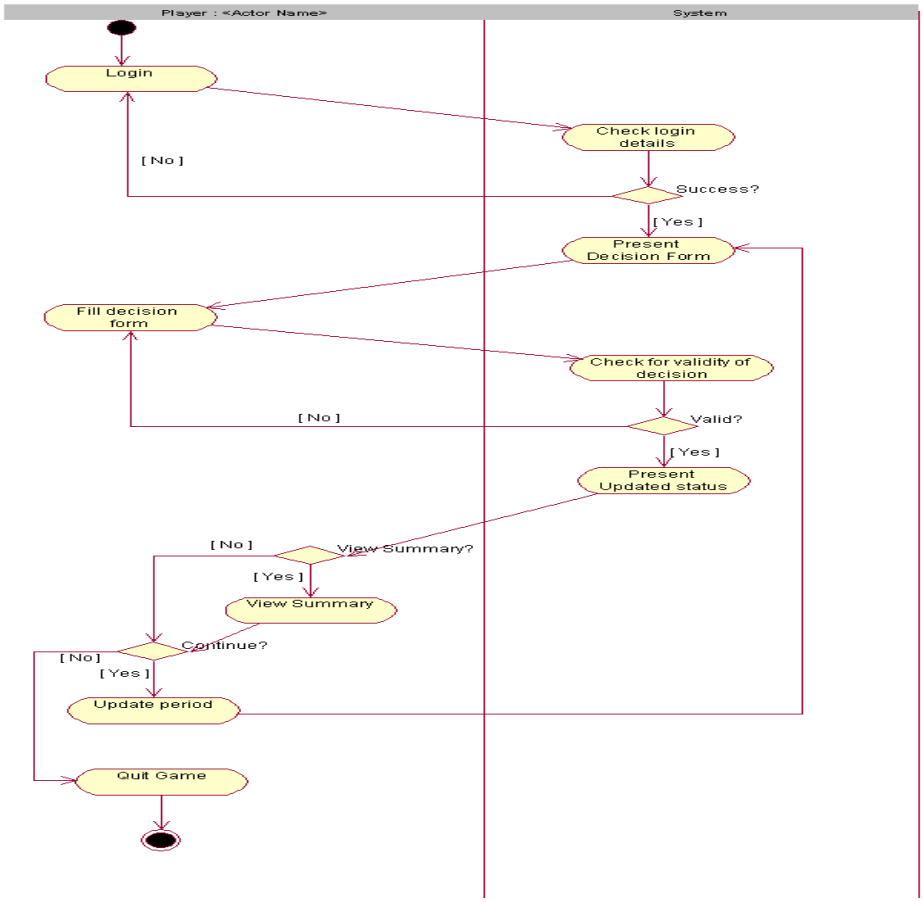


Fig 9: Screen shot of the home page of the Beer Game

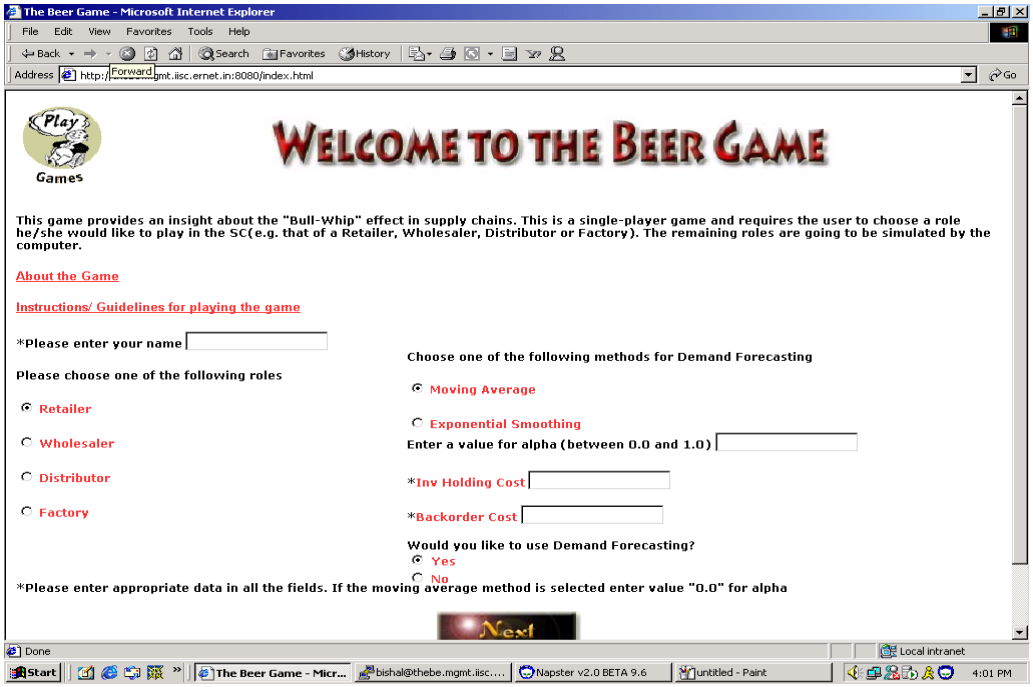


Fig 10: Variance

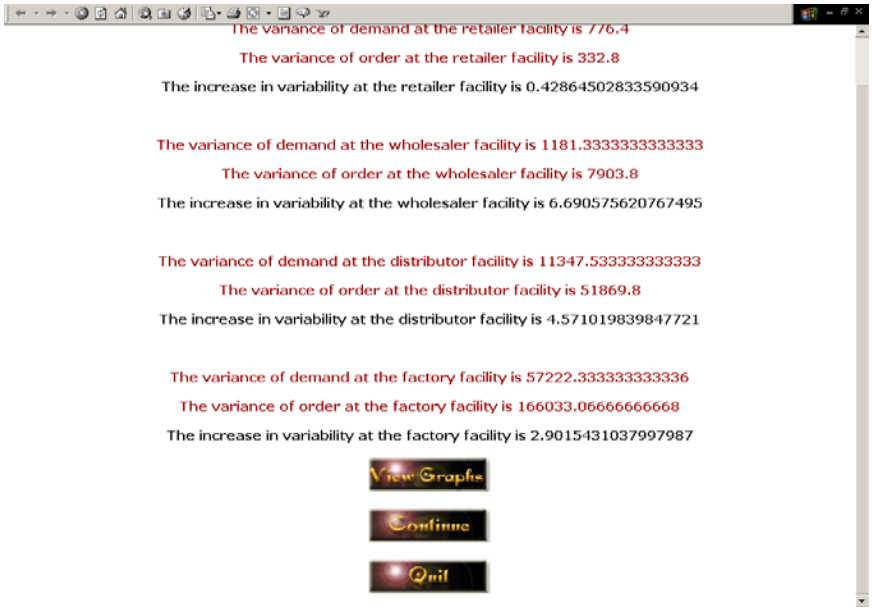


Fig 11: Graphs