*A Report On*

Trade Game

Submitted by

G. SuryaDev Reddy Yathish Reddy.M Srihari B T

14CO214 14CO254 14CO256

*In partial fulfillment for the award of the degree of*

Bachelor of Technology

In

Computer Science & Engineering



Department of Computer Science & Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, KARNATAKA SURATKAL

April 2016

Contents

1. Introduction To the project
2. About the Choice of Trade Game
3. Project Design Issues
4. Drawbacks of using SDLC in Trade Game
5. GDLC used in the Trade Game project
6. Implementation and explanation of each phase of GDLC
7. Conclusion

Introduction To Trade Game:

THE GAME IN BRIEF:

Trade Game is a variant of traditional famous board game Monopoly. Trade Game is the game of buying, renting or selling Properties so profitably that players increase their wealth – the wealthiest becoming the eventual winner. Starting from the "GO" space, move your token around the board according to your roll of the dice. When you land on a Property that is not already owned by anyone else, you may buy it from the Bank. If you do not choose to buy it, it is auctioned off to the highest bidder. Players who own Properties collect rents from opponents stopping there. The construction of Houses and Hotels greatly increases the rent you can collect, so it is wise to build on as many Sites as possible. If you need to raise more money, the Bank can give mortgages on Properties. You must always obey the instructions given on Community Chest and Chance cards. Sometimes you will be sent to Jail.

OBJECTIVE:

To be the only player left in the game who is not bankrupt

***About choice of Trade Game***

Trade game is a variant of traditional monopoly. We wanted to choose a project whose solution requires us to grapple with modules and interfaces, abstraction and information hiding, graphics programming, User-interface design, testing. The basis of the project is already familiar to us so that several unproductive weeks of problem familiarization is avoided, and specific design issues are tackled right from the beginning. We have found the Trade Game to be an excellent basis due to following reasons for this kind of group project.

(i)  ***Complexity*:**

· The board, which consists of squares that represent buildable lots, railroads, and utilities; and squares that require different actions to be taken;

· Players' tokens, and their movement around the board which involves rolling the dice and picking up chance and community chest cards.

· Protocols for building houses and hotels, mortgages.

· auctions, which must be conducted for property whenever a player lands on and refuses to purchase outright an unowned square,

· To show the built houses and hotels on the appropriate position on the board.

(ii) ***Program Structure and Data Abstraction***

The complexities above must be handled by modules that deal with each other along clearly-defined interfaces, with information hiding playing a central role

(iii)**Preciseness about the Functionality**

We insist that the game be implemented precisely according to the Official Rules, with all of the nuances of auctions, building shortages, mortgages, bankruptcies, sales and exchanges of property, jail and income tax. The rules, then, serve as the program specification. As with most natural language specifications, though, these rule s may seem to contain some ambiguities and omissions; their correct implementation is a real-life software engineering problem

(iv) ***User-Interface Design*:**

The users should be able to get good feel of the user-interface (how the board looks like, how to display the property info, player’s tokens movement on board). So owing to all these considerations, we are subjected to following constraints

· The interface must be primarily mouse-driven

· Colours of the displayed board must match those of the real board, pictures of tokens and buildings must appear on the appropriate squares,

· Each aspect of the game should be realized in a natural way: Pictures of tokens and buildings must appear on the appropriate squares, we should see the dice roll ..

(v)**Graphics:**

The problems of representing the board, tokens, cards, and buildings,

and the animation of tokens as they move around the board are further complicated by the requirement of a mouse-based interface . A good deal of graphics programming is required in realizing the pictures, the animation, and switching displays and viewpoints

**PROJECT DESIGN ISSUES**

The importance of the design phase in software development is often underestimated. We therefore performed modular decomposition of the program on our own, which we believe to be a valuable learning experience.

Medium-sized software projects such as Trade Game can be approached from **procedure-oriented** or **data-oriented** viewpoints . In the former, the task is divided into a small number of sequential subtasks, and communication between tasks is organized somewhat like a pipe . In applications that are characterized by homogeneous data, this approach will often lead to a highly-cohesive and loosely coupled structure. However, it is difficult to organize a Trade Game program in this way because of the variety of data objects: each "procedure" has the potential of modifying the entire state of the game .

In the latter approach, the organization of the program is centered around the data that are to be manipulated. In a Trade Game program we name modules after these objects: Players, Board, Buildings, Cards, Dice, Cardsdeck; we will also need objects for the Mouse, Keyboard. Each object is viewed as having a state, shielded from the rest of the program, which may never be directly read or updated, except by specialized procedures. For example, a player is characterized by his/her token, cash on hand, position on the board, e tc. Now, it would be unwise for a command like P .Pos 17 (moving player P to square 17) to appear anywhere in the program . Thus we hide the representation of a player, and export, for each component, procedures for reading and writing .A player's position can be modified only by a call to the

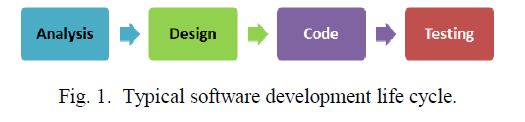
PROCEDURE MoveCurrentPlayer (NumSpaces : INTEGER )

Which, in addition to updating the player's position, animates the move , takes care of updating salary, and so on. In general, side effects of an action must be encapsulated in the update procedures. The identification of objects, the description of their state, and the careful design of update procedures allow information hiding to be realized.

**Drawbacks of SDLC in game development :**

For the game development, Requirements are not known well in advance and the game should be able to modified according to the user requirements and interests. While SDLC is a systematical process of engineering to develop software, game is not purely a product of pure engineering.If you create a game and release after entire completion of the game, there are risks that you game may not be inline with the user interest’s. We need to release a prototypes after each stage to assess the fun, features of the game which cannot be properly addressed by traditional SDLC’s because the game is not just a pure product of engineering but game is more like a craft, created from the combination of interleaving, multidiscipline aspect, from art of characters, programming, acting, and the management and integration of those aspects.

Regarding the Testing phase, Game testing is much complex than the normal testing. It is not something that can be left to untrained laymen. There is a place for live user test, or alpha and beta test, but those are just a few pieces in a much larger puzzle. Usability or user experience is something that is tested for all software, but Fun Factor Testing is something unique to games, since they are an entertainment product. Games are not only supposed to work intuitively and provide a good user experience – they also have to be fun to play. Creating balance between different options, as well balance of difficulty for different levels, is also something unique to games. Balance Testing can only be done in a good way with a vast knowledge of game design and how the target audience responds to different difficulty levels. It also requires many hours of actual gameplay of the game under test. Therefore, a game development requires specific guidelines which govern its development process and assess the quality of the game at each stage, which requires game development lifecycle (GDLC).We have analysed various GDLC’s and followed which suited the most to our project.

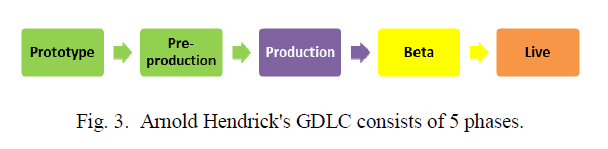


GDLC Used in our project:

**Arnold Hendrick’s GDLC :**

Arnold Hendrick defines five steps of developing a game, shown in Fig. 3. Arnold Hendrick's GDLC consists of 5 phases.

Starting point of creating a game is to create the initial design, concept arts, and several prototype in prototype(1) phase. The next step, pre-production (2), is to make the documentation in a form of game design document. Production (3) is related to the construction of assets, source code, and the integration of those aspects. When the build is ready, beta(4) testing is conducted to draw users’ feedback. Live(5) is when the game has already been passed the testing and ready to play.



**Implementation and explanation for each phase**

In GDLC we’ve followed the following stages which can be understood in analogous to Traditional SDLC models as mentioned in the brackets:

1. Intiation (Identifying the requirements)
2. Pre-production (Design Phase)
3. Production (Development Phase)
4. Alpha version (Testing Phase)
5. Live(Deployment phase)

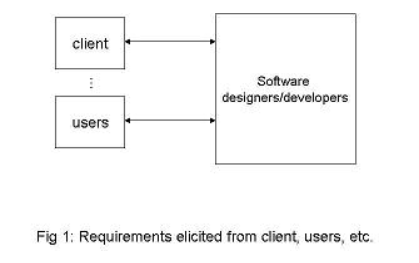
***Intiation Phase***

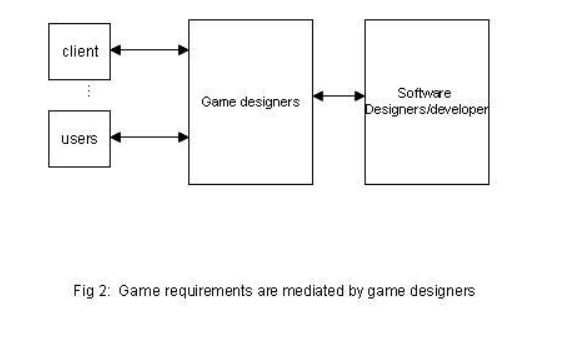
The Trade game concept is taken from the famous board game Monopoly and game description is made to our convenience. So We were well aware of game rules , Actors, game genre and intial game design.

***Pre-Production Phase***

Pre-production involves the creation and the revision of game design and the creation of game prototype.The process is slightly different for games. Game design focuses on defining game genre, gameplay, mechanics, storyline, characters, challenges, fun factors, technical aspects, and its elements documentation in game design document (GDD). Game design and development incorporates game design in addition to software design and development. Game design is often considered a "pre-production" phase that results in a Game Design Document (GDD); this document, describes the game in detail. It should answer any and all questions about the game and how it is played. It includes the story, game mechanics, character descriptions, concept art, market analysis, feasibility analysis, a production schedule, etc.

The requirement process in games, therefore, has two steps. The game designers elicits requirements from external stakeholders such as the client and users. The designers assimilate the results together with their game design and background research in the GDD. Software engineers then collaborate with the designers to convert the GDD into a software requirements specification (SRS).





***Production Phase:***

In programming the solution we identified low -level objects (i.e., those objects in the device, or computer, world), and high-level objects (those in the world of Trade Game). Each module is tackled by each one of us, and we partitioned the program into modules and designed their respective interfaces which had been the primary agenda for our early group discussions. We had to program object-actions using a data-centered approach, and had to pay special attention to information hiding. Here are a sketch of their approach and some technical details of the implementation:

five modules, and the main module, play Trade Game :-

* **Player** :A player is represented in the Players Module having related attributes like property bought , property mortgaged , money , position etc..Procedures and functions to read and update a player's state are exported. We hide all the information that is used to represent the playing order, the token pictures, and the animation routines. A routine for selecting a player (say, who wants to sell a house) is provided here.
* **Board**:The Board Module exports procedures for reading and updating the states of the squares. It also takes the positions of each square on the board. We also export a procedure for selecting a square (say, to be mortgaged) which, like Player selection, needs no menu.

**o DrawArea**: It exports all the information of the squares which are collectively stored in the board module and sets the positions for player tokens with respect to each square

**o Buildings**:In the Buildings Module we keep account of the number of available buildings, and provide procedures for buying and selling them.

**o Cards**:The Cards Module, exports information of the cards from a file It picks a card (which automatically triggers the card's action), and for replacing, transferring, and querying the owners of Get Out Of Jail Free cards.

**o Cards Deck**: It collectively stores the information of all the cards and it picks a card (which automatically triggers the card's action), and for replacing, transferring, and querying the owners of Get Out Of Jail Free cards.

**o Dice module**:From the Dice Module we export only a procedure for rolling the dice, and displaying it on a frame

**o Info**: It stores the title deed cards information of a player and getout of jail free cards if any.

**Game Testing**

**ALPHA Testing**

Alpha testing takes place at the developer's site by the internal teams, before release to external customers. This phase is used to test game functionality and usability. To check whether the game includes necessary rules and procedures to make the game fully functional is usually checked by Playtests. Game’s features (if they are operating well )are tested via the accomplishment of each playtest scenario .Fun is tested via playtest and direct feedback from fellow developers, whether it is boring, frustrating, challenging, etc. Accessibility can be tested via observing the tester behavior. If tester find it difficult to play and understand the game, it means that the game is not accessible enough.

The output of testing is bug report, change request, and development decision. The result will decide whether it is time to advance to the next phase (Beta) or reiterate the production cycle.

In the course of our game we’d tested the game for all the basic use cases, and after the development, game was tested several times for clear understanding of look and feel, and the we ensured the fun in playing is never spoilt and maintains the rapport between user and system

**Beta testing**

Beta testing comes after alpha testing and can be considered a form of externaluser acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team known as beta testers. The software is released to groups of people so that further testing can ensure the product has few faults orbugs. Beta versions can be made available to the open public to increase thefeedback,to maximize future users. In this case feedback was taken from our mentors on the functioning of game.

Beta testing still uses the same testing methods as the previous testing method but the tester , who tests the game can be of 2 types. (i)closed beta and (ii)open beta. Closed beta is only allow invited individuals to be the participant, while open beta allow anyone who register become the participant.

Live

This is the time where product has to be ready for release .This involves how to release game package, post-production activities, and planning for game package . Release involves product launch, project documentation, planning and maintenance and Game expansion(if required).

**Conclusion**

The combination between engineering and arts is the aspect that the SDLC do not consider as a important thing and so using SDLC becomes a challenge in developing a game.So We have followed the three key phases of game development life cycle . They are design and prototype, production and testing.

We found the course successful in that, We were able to learn how to structure a program, design and implement the modules in a data-centered fashion .The program was too difficult while handling graphical user interface of the tokens to be moved and houses, hotels to be built on the board. After structuring the program and implementing the modules, we did not find much difficulty in coding the game functionality. We were very much satisfied with the successful implementation of the mouse based interface. Unfortunately, no voice or sound routines were implemented .We require both logical and artistic nature to develop a game. The interface was tested in the intermediate stages, and no real problems arose during system tests except positioning of tokens on the exact positions of the squares

Finally, the application of the above GDLC guidelines has successfully delivered a good quality game ensuring all the quality parameters such as fun(engaging), balanced game(difficulty level is just fit), internally complete (game functionality), accessible(user’s comfortness in playing the game).