BlackJack Theory Work Oliver Walters

Defining the Problem:

The Problem:

The problem that this software needs to try and solve is to produce a revised digital version of the BlackJack game. This game needs to be not just functional, but also entertaining and aesthetically pleasing.

User/Client Needs or Objectives:

- The software needs to be bug free and perfectly functional
- Programming the software will require at most 6 weeks to fully implement
- The program should be able to simulate a realistic and proper game of blackjack with three computer opponents
- The player should be able to interact with the proceedings of the game
- Maintainability of the solution is also a key factor
- The software needs to have an AI model/difficulty setting in order to further the complexity of the game
- The objective of the task is to develop such a solution in the most efficient and maintainable way possible

Software Needs/Limitations:

- Requires a programming language to implement the solution (NodeJS)
- The program is limited to google's standards for development
- A maximum usage of memory and processing is possible as well as the limitation put on the transfer and storage of online information
- The program needs to be able to run efficiently on basic computers
- The boundaries of the problem can be described by the user requirements, such as the fact that the final product on necessarily needs to be blackjack with a customisable difficulty

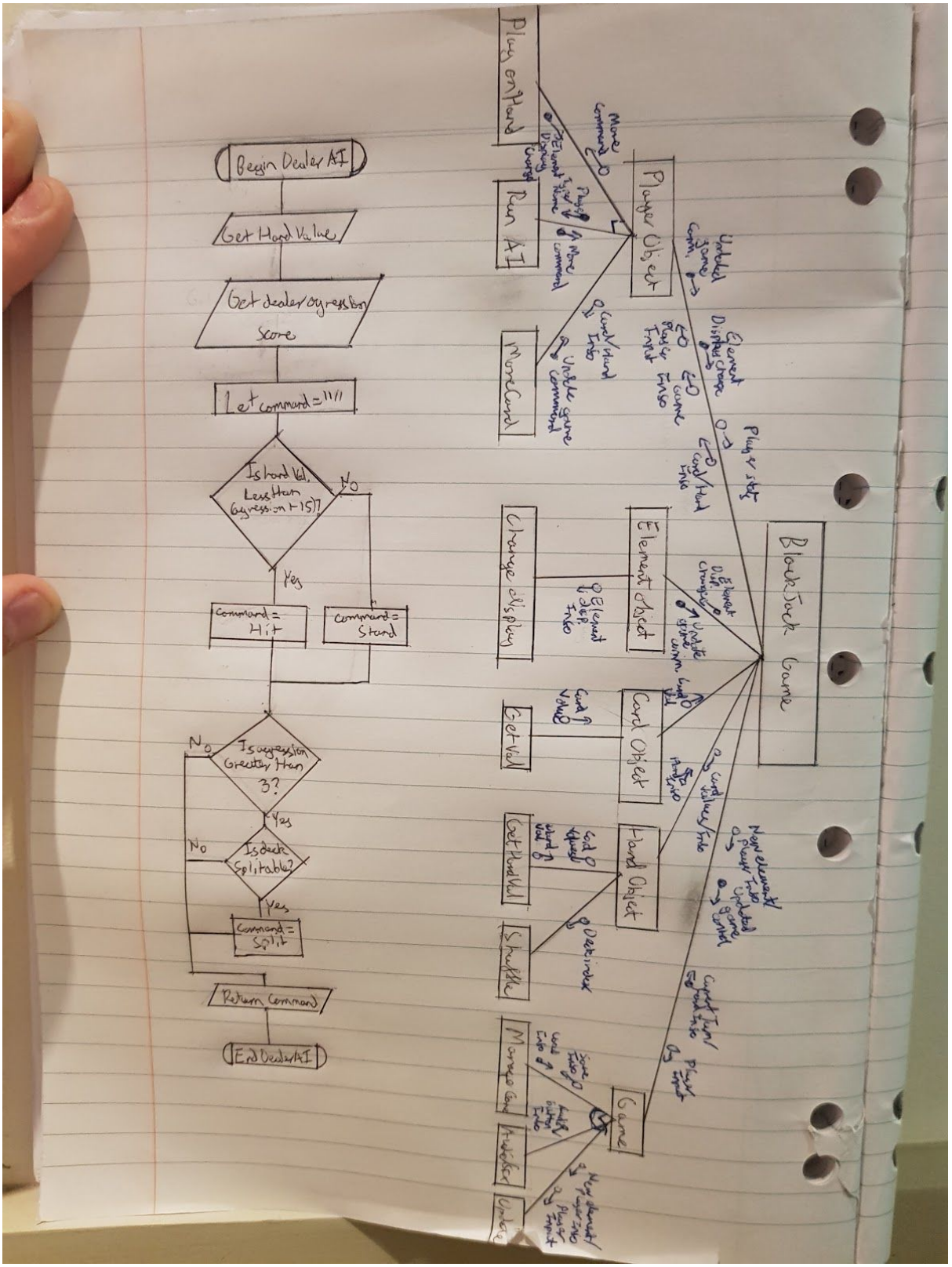
- **Issues relevant to the solution:**

- Developing the solution in such a short time will require the use of a faster development and implementation approach
- The solution must be compatible with the latest versions of the nodejs library and the HTML5 language
- Developing the solution requires that the software abides by the guidelines set by the internet browsing platforms as well as common developing codes of conduct
- The performance and memory consumption of the system should not hinder the user's ability to use other applications on a multitasking operating system
- Response times need to be kept at a minimum
- Ensuring that the developer's expertise is not overestimated - keeping the planned project in tune with the developer's skillset
- Considering the social and ethical issues pertaining specifically with disability-based inclusivity and quality of the final product

Description of the Design specifications:

- The program should apply the use of Object-Oriented programming
- Developing the program with the intent to make it both maintainable and modular as to ensure possible future amendments don't go poorly
- The program should be able to properly shuffle the deck and play through such cards in the deck. Involving methods to evaluate the point value of the hand
- The human player can play through a system of hitting, betting etc.
- Further actions that can be played in blackjack such as doubling down or splitting
- A suitable dealer algorithm that follows the rules of play
- Proper and aesthetic GUI applications and easy to use screen designs
- An AI system where the dealer and the computer can be customised as well as making decisions based on the situations encountered in game

Diagrams associated with Planning the Solution:



Development Approaches:

Structured:

Using the Structured approach to develop this project is useful in the fact that the report structure is already set on the different aspects of the actual assessment task. It is useful as each stage of the development process is well defined and as well, it's iterative nature ensures that the final product is error-free.

However, potential problems may arise with the fact that it may be set too much in its stages whereas a project like this will not be developed as such. Other issues involve the fact that there is a very small team size of one developer, a very limited budget of zero and only a month or two to develop the project. Which are all things the structured approach can't handle.

Agile:

Using the Agile approach may be useful to develop this program as it allows for fast development of the application with the ability to gauge feedback on incomplete versions of the product (via the forums/teacher input). This also helps as there is a very small time limit for the assessment.

One main issue is that the agile approach usually will require fast additions to the code as well as the fact that it still may require large budgets/teams which is not available. As well, the agile approach is only useful in fast-changing markets, which is not what the task will aim to operate in.

Prototyping:

Using a Prototype approach is useful in the fact that the solution requirements are designed to be vague in order to allow for proper room for creative activity. Prototyping is also useful as there can still be feedback on products. This also is an advantageous method to use as it allows for a more casual form of development with still proper stages of development.

However, the prototyping method may not work as there still is quite a small amount of time to develop the product so an experimental approach is not warranted. Using the evolutionary prototyping approach is also quite useful however it may take too long to develop working proper prototyped versions.

Rapid Application Development:

RAD is a useful methodology as it is able to develop the program in due time. It's speed and lack of formal steps mak be the most advantageous as I am able to develop different sections of the assessment task when I am able to. This approach also works well as the use of a higher-level language allows for the best application of RAD. It also is useful in the fact that it works in low-budget, small team and short on time scenarios.

That does come with a few caveats, such as the fact the often messy style of development may lead to serious errors that can jeopardize the entire task. This is an issue that needs to be dealt with if it is going to be used as an approach.

End User:
End User development is a useful ideology in that it fits the closest to the situation, as, the only people who really care about seeing it is myself and the teacher. This also is supported by the fact that it is a one-man programming team with zero budget.

There are problems with this approach such as the fact that it can't fit well within the constrains concerning the software requirements as well as the fact that the resulting code may take too long to develop with the end product being sloppy poorly made. This is not a good strategy.

Summary of Development approaches:
To conclude, the best development approach to choose is a mixture, that is, the best approach is the rapid evolutionary prototyping methodology. This combination of RAD and Prototyping fits the best as both approaches provide the best of both worlds with the swift and lack of formality of the rapid approach being beneficial considering the scope and scale of the project. The prototyping also helps benefit such by ensuring that some structure remains upheld and that errors can be properly dealt with. Evolutionary approaches also work best as it allows for a clean transition as the project nears its goal.

Implementation methodologies:

Note: these methodologies concern how the program is developed from prototype to prototype and eventually reaching the end of the project.

Direct Cut Over:
The direct cut over method of implementation may work considering the ease of such operation as well as the fact that there is zero budget to actually perform a more complex operation. It works better with the fact that the system may not be able to run new versions of the software. Also as it works quickly.

However, carrying out this implementation is risky as it can lead to serious errors that cannot be fixed without causing serious damage. This also is combined with the fact that there is no need to directly cut over as it also takes up too much effort to manage the change.

Parallel:
The parallel method of implementation works in the fact that it allows testing the new system in an A/B style format where if anything goes wrong, the system can be amended without the entire thing crashing and burning. It is also useful as it is still cheap and quick to run which means it can still be applied properly.

This method is difficult to deal with as it requires both more power and more storage to house both modules. This also brings the issue of having to maintain an older system that was probably amended specifically to rid it of errors. This can lead to those same problems transferring to the new system if they are allowed to communicate with each other.**Phased:**
Phasing the implementation of new modules is useful as it allows for a smooth transition from old to new. This means that any errors in the new system will be caught through the employment of testing and maintenance which will ensure the new system does not contain errors and can allow for the next stage of development.

This method has errors, however, specifically with the fact that it may take more time to implement the software as it is a slow process, which is not useful considering the development methodology. It also is difficult as it requires the editing of possibly error-ridden or not very maintainable code, which can slow the implementation even further.

Pilot:
The pilot method is also quite useful in the fact that feedback can be provided on the work performed and the prototypes mad of the new system. This also helps as there is a form of field test that gives a sense of what might actually need to be changed. Also, it allows for a sort of parallel style of implementation that isn't really as the new and old can be run on different devices at once.

There are some issues mainly concerned with the fact that it is often expensive and big-budget. As well as the fact that it requires a subsection of the user base to test first. This is obviously not probable as there is only ever 2 users for this project. As well, this can still

create issues as the users in the pilot may not be able to have their data from the old system and getting it from the old system can damage the new.

Implementation Methodology Summary:
In summary, the best implementation method that could be employed is the phased method of development as it allows me to slowly transition the code from one version of the product to another in a seamless manner. This also helps combat the messy error creation from the RAD approach as well as being able to test the solution as it's being produced and implemented. All of the other implementation formats either don't work or can't in this specific case. However, technically, project-wise the actual implementation is direct cutover as the solution did not exist previously.