**3.1 Requirements Gathering**

The original product specification given by the formal product specification was to create “an interactive program, with GUI, that presents a go problem on a board, and interacts with the user as they explore the solution space of the problem. The program uses tree search algorithms to find good moves.” From this, some ambiguities were identified in the requirements such as what tasks, goals, and objectives would be explored in the program. So a requirements gathering interview was arranged with our project supervisor, Dr. John O’Donnell, to help specify the requirements further.

For this interview, any ambiguities in the project requirements were identified and questions intended to reduce them were formulated. Questions were also created that would identify more requirements, so that a more specific requirements document could be created at the end of this process.

**Interview Questions**

**Nature of problem**

What are the types of objectives the program will be asked to solve?

* It is hard for the program to determine the nature and goal of a problem as they are laid out in the problem book. The program would have to choose its own boundary for the problem and determine which pieces are irrelevant to the overall solution of the problem. This is too ambitious, so a boundary should be defined along with a nominated group of stones that should either be captured or saved.

What difficulty of problems (30 kyu - 9 dan) should the program be able to solve?

* It should be able to solve problems correctly at as strong a level as possible. At a minimum, it should be able to solve simple problems using brute force. The program should be tested with problems of an estimated strength against a competent human player to find the kyu ranking of the program. 20 - 25 kyu is around the right ranking that the program should be able to achieve. If the program ends up better than this, problems of better ranking could be used to improve it further.

How complex (number of moves) will the solved problems be expected to be?

* The number of moves to solve is a useful metric for calculating complexity when we use a brute force algorithm, however for more complicated heuristics it becomes less useful. A directory of collections of test cases should be created so that many tests cases are able to be ran over the program and calculate the complexity. However calibration is difficult as humans and computers find different things difficult. Humans can perceive “good” moves and can deliberately structure the tree deeper rather than wider by forcing the other player to make specific moves to save their pieces.

**Solutions**

How long should the AI have to respond?

* The program should have no time limit to solve, but it should be reasonable. ie. making a move in a few seconds or minutes compared to days and weeks. The interface should have a way for the user to define how long the AI has to take a turn. Thirty minute turns are too long in practice. To control time, a difficulty chooser could be implemented that would set time limits for turns or set depth limit for move searches. This should be a rough guide and not a set limit ie. if the time limit is set to 30 seconds, the AI can take from 10 - 60 seconds. As long as the times are reasonably similar.

Whom should the AI be able to play against?

* It should be able to play against itself or against humans.

What minimum success rate should the AI have:?

* The program should be able to solve 28 kyu problems with certainty. It may be unable to solve 15 kyu problems reliably but may solve them on occasion. For human players, if two players of the same rank play against each other many times, the win ratio should be 50:50. If one player is one rank stronger, the win ratio should be 66:33 in favour of the stronger player. Ie. for each rank, the stronger player is statistically ⅔ times more likely to win. The handicap make the game of “Go” even. Weaker human players are inconsistent, they may have a good opening game but be less proficient at life and death problems. Generally players of 10 kyu must be competent in all aspects of the game.

How do we know the AI has succeeded?

* **A section of our dissertation should be on this subject.**
* Maybe a main group could be elected to either be captured or be saved depending on problem. If the group is captured, or is unable to be captured then the program could be alerted that the problem has been solved. Pitting the AI against itself is not a good way to test the successfulness of our program, so the program should be tested with humans who know how to solve the problems so they can pick out problems in the program. There are two avenues that the testing process should pursue, the first being “Small data set; in depth evaluation,” where good “Go” players are asked to make good moves against the AI on a problem and then play again but make bad moves. Then they should be asked if they think the AI is good in the program. Ideally, there should be 5 (minimum) to 10 (ideal) testers for this step. The other testing avenue is “Large data set: low depth evaluation,” where a person, aided with the answers to the problems, is asked to test the first move the AI makes of many problems (several hundred). If the AI makes the correct first move, it can be assumed that the program is likely able to find the correct solution.

**GUI**

Is a textual user interface required in addition to a graphical user interface?

* Maybe start the program in TextUI, then have a command to start GUI but the ability to be able to switch between TextUI and GUI no needed. A TextUI is important for the portability of the program, as it can be hard to get GUI’s to work on different systems.

What information should be shown from the AI?

* The ability to go back through the moves showing: What moves were considered; What the heuristic evaluated the value of the moves as; Why it did not take moves that would be considered as “good”; Why it took moves that would be considered as “bad”. Also, make it possible to query the AI, to ask it what moves it is considering or show the value the heuristic gives for a specific position. This would help to find the bad decisions in AI so that fixes are more easily targeted.

Also, after a discussion with Dr. John O’Donnell, it was decided that the program would be created with the purpose of solving Life and Death problems as solving the entire game would be far too ambitious for a project of this length.