Vision and Scope Document

CIS 641/642 – Software Engineering Project 1

Team Name
SE3

Team Members
Shicong Liu
Quan Kong
Byron Wheeler
Tanmay Varma
Zac Throneburg (TL)

Team Project
Sudoku

Problem Statement

a. Project background

For our CIS 641/642 project Dr. Gustafson tasked our team with developing a computer Sudoku game. The game has two basic requirements. First, it must allow the user to import image files of clean Sudoku games for gameplay. Secondly, the game will be developed using the C++ language.

b. Stakeholders

Dr. Gustafson	(Professor)
Joseph Lancaster	(Teaching Assistant)
Shicong Liu	(Team Member)
Quan Kong	(Team Member)
Byron Wheeler	(Team Member)
Tanmay Varma	(Team Member)
Zac Throneburg	(Team Leader)

- All stakeholders require a functional computer Sudoku game that meets all requirements by the end of the 2013 fall semester.
- The Professor and Teaching Assistant require the team to give progress presentations and keep a current project directory.
- The team leader requires weekly updates including progress and hours spent on the project from all team members.

c. Users

- Sudoku game player – The player will require a functional game that meets all project requirements.

d. Risks

Delays due to busy schedules
Delays due to project part merging (image scanning and gameplay)
Version control issues

e. Assumptions

All game images will be clean. This means images will be properly oriented in the x-y-z planes.
All game images will be of the same file type, image size, color, and font.
All game images are considered to be an original game. That is a game with no extra numbers added to it.

Vision of the Solution

a. Vision statement

Our goal is to implement a fully functional Sudoku program. Image processing will be used to import clean Sudoku game images into a user interactive GUI. The GUI will allow user gameplay. The project will be coded in C++.

b. List of features

- Image importing to game file – The program will take in a clean game image file. The usable format will be .bmp files. In the future, we may add implement for other file types. The output game file will be a .txt file that is made up of a series of zeroes for blank boxes and valid numbers that are already filled in. This game file can then be loaded into the GUI. The game file names will be incremental starting with game "000.txt."

- Game file importing to GUI – The game file that is generated from an image can be loaded into the interactive user GUI. This will initialize the game and enable the user to play through the game.

- Possible move hints – Show available moves in an individual box. The moves will be shown based on the current state of the game, but it will necessarily mean that each available move is the correct move for a game solution.

- Valid move checking – When the user enters a move, the GUI will check the move validity. If the move is invalid, it will not allow the move to be made. Invalid moves will be reported to the user. Valid moves will be stored in the GUI.

- Game file saving – Outputs the current game to a .txt file. The format will include the original game file of zeroes and the original numbers that existed in the image. The remaining part of the file, will contain the game where the user last left off. The game save files will be the name of the original file followed by an underscore and an incremental count, such as "000_save1.txt."

- Game file loading – Loads the game file. The created game will be initialized using the original game content, and then the GUI will be updated to reflect the current status of the game where the user left off.

- Game generation (optional) – Generates a new game from scratch. This will possibly allow the user to choose game difficulty. The generated game will be saved to an output file in the same way an imported image is saved.

- Game solver (optional) – Uses brute force method to solve the current game. It will ignore inputs the user has made so far. The brute force method will solve from the original game file. The solution will be output to a text file based on the original game name, such as "000_solution.txt."

- Undo and redo (optional) – The GUI will allow the user to undo and redo moves. This is not necessary because we only allow the user to make valid moves, which they can choose to change later if a conflict is found. If we have time this feature will still be considered.

c. Scope of phased release

Over the course of October we will develop a basic GUI and game framework. We will also work on our image processing code.

During November, we will complete the basic GUI and user game play. This will include file loading and saving. Image processing will be the priority from this point on.

During November and heading into December, we will complete the image processing. If this code is finished early, we will consider implementing optional features.

d. Features that will not be developed

- Using images that are not ideal.
- Using images that are of certain file types, example .jpg.
- Using game images with different fonts.
- Other features we do not want will be determined as they become available.