Third Year Project 2013/2014

Report 1 User Interface design for a Sudoku solver Report.

A Sudoku Solver Using Constraint Satisfaction (& Other Techniques)

Thomas Paul Clarke

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Dave Cohen

User Interfaces are fundamental for a program that is going to be user friendly, the user interface will allow the program to show a window or a series of windows that will allow the user to see all the data that can be output as well as allow the user to input the required information. This process has become common practise for all programs.

The history of User Interfaces for programs often operating systems has changed so much since the first Interface was displayed in 1968 by Douglas Englebart[1], this project involved trying to display everything that a user will need and a way to make it understandable instantly, as well as allow the desired inputs and make it clear to the user most of all.

This practise also needs to be followed within the GUI for the Sudoku Solver, finding the simplest way to display the inputs and outputs as well as make it visually interesting and in an essence not dull.

For this project the user will need a very simple layout something that allows the buttons to be displayed in a simple and ordered way. The Sudoku grid it's self is going to be created in a recursive way that will allow each cell to be created individually and this means they will all be the same, but some of the design questions have been what should I use as the user enabled part of these cells, i.e. how can a user put a number in them.

This has been a way in which the whole structure of the grid and user interface will have to be made around to enable the user to edit the grid.
Filling the grid with buttons that once clicked on can change colour and shows they are editable, the user could then type in the desired number using the keyboard.

Another method is that there is a way to again click on a button and then show an editable state, this then allows the user to use some on screen number buttons that allows them to change the number, this would also be beneficial as it restricts the user so they can only place limited numbers (1-9) in the cells. Unlike a keyboard that gives a lot more options that will have to be programmed in to prevent errors.

If a button is not the best strategy then there is a way to allow the user to pick up and drag a frame that represents a number and drop it into the cell that will use an event

[1] http://arstechnica.com/features/2005/05/gui/ last Accessed: 31/10/13

handler, allowing the use of this smoother and maybe more user friendly system of selecting the numbers inside of the cell.

Technical Drawings have been made to draft the look and feel of the User Interface, all using different styles and ideas within to make it feel different as well as add increasing functionality.

1. This is a very simple design and was the original just to give an idea of what kind of inputs and outputs there will be and allow them to be visually displayed in a simple format. This design used simple buttons that would be linked to event handlers and through them methods to act on the desired function.
The grid for the Sudoku uses a very simple recursive grid layout that puts the buttons into a correct format. This keeps the design very simple and the desired input for the numbers would be via

2. This design includes different features for example a note box which will be an editable box that allows the user to put notes into if needed, there is also the 1-9 buttons that allow the user to select a cell and input the number from this sequence.

3. This design implements the drag and drop function as well as another window that allows the user to save multiple puzzles and when the load button is pressed it displays a window with the saved puzzles and allows them to be loaded, for simplicity this can also be done within a drop down box and this has also been pointed out in the design.

Each of these designs have been replicated within window builder to show and create a useable experience so that the best one or a combination of the features can be used within the final project. This also was a time to implement a structure that could test and show these features as well as to find out if they were practical.

To create this and provide functionality in an easy and more importantly an easy to build on process, I have used a model view controller design pattern with the GUI this will allow the methods and functions to work without being within the design code and keeps it separate this will also promote new functions as they will be kept separate and are processed independently though the controller and the project takes what it needs from the program and those computations and will output them.

Other ideas that have been explored are the ability to move the project to an android device as well as the original concept, this would have meant creating two user interfaces, but the interface would be one of the things that would be amended, as the rest of the functionality should have been standard within the program.

The ideas have also stretched as far in terms of user interface that it should have been something that was very different from the current structure of buttons and a large one window interface and use the task bar at the top instead of using buttons, keeping the window tight with just the Sudoku grid displayed and only show this and keep the options hidden in a menu to keep things cleaner.

Using a resource[2] which represents a process of evaluation of a user interfaces, these heuristics can be found on Heuristic Evaluation (Nielsen and Molich, 1990; Nielsen, 1994). I will summarise the relevant heuristic and relate them to the GUI of the Sudoku Solver.

1. Status of the system to the user. This means to inform the users of the programs inner workings in terms of time constraints, for example if needed there could be a loading bar on a splash screen as well as a progress bar when the user requires the puzzle to be solved.

2. How the system differs from a real life experience. The system originally was a game to be played with pen and paper and that allowed the user to make notes and edit their own mistakes, the mistake handling is possible however there are functions that can be implemented that are possible in real life and that includes a process of filling in squares that could be a possibility.

3. Allow a user to have the experience they want. Done by allowing the user to save and load previous puzzles as well as letting them solve the puzzle if they need or a help function that shows just a few steps.

4. Keeping the program consistent. This is vital to the way it handles and its appearance, buttons need to be the same size especially the ones within the grid to keep it a uniformed structure.

5. Error prevention. To keep the program from having errors or if It does provide safe guards, for example adding a number higher than 9 into the grid will result in an error, however if the user was made to choose a button between 1-9 then they will be unable to make that error.
6. User able to see all of the relevant information. All that the user should know should be visible, as well as everything that is trivial or complicated should be kept from the user. This allows the user to be able to focus on the things they want to for example, there could be time and date stamps on each saved Sudoku puzzle so that it is recognisable to the user, and even further a small visual example of what that puzzle actually looks like.
7. Easy to use. To make the user experience simple and without making them go through long processes to get to A often used function. This can be prevented in this project by instead limiting users to use the buttons as inputs to the cells they could also use the keyboard as will allow faster adding of integers, this can also be made faster by using a button like tab to move to the next cell.
8. Design. It is vital that only information that is needed is displayed and not in a manner that could confuse the user, for example if a program is done in a kind of order, like new game, no. players and hardness level, these should be ordered so that the user isn't selecting a new game after these options. The display needs to be minimal and without clutter, meaning non vital information, that could be displayed at a point in time, not all the time.

[2] Heuristic Evaluation, Usability Evaluation Materials by Darryn Lavery, Gilbert Cockton and Malcolm Atkinson at the Department of Computing Science, University of Glasgow
http://www.dcs.gla.ac.uk/asp/materials/HE_1.0/materials.pdf

9. Error handling. Error messages can be used to explain to user what the problem is instead of the program crashing, this will allow the user to correct their error if it was caused by them.
10. Help. A help function should be needed to explain the game and the user interface to a completely new user with no understanding of a program or the game it's self, this could be done by using boxes that can be taken away that explain what the user needs to do, a drop down box to show the game rules and process. Or an alternative is to right a manual, however these are not practical in modern day as users often want to get started and expect a tutorial.

After completing these heuristics it has forced several changes within my project and the design stages that I have currently made, there will need to be a few more functions to allow the user to get the best experience. However with these designs I made previously included the functions that I deemed most important and now these have changed due to the heuristics, although that is their purpose, to make the designer take a step back from what they want from a system and the real reason it is made.

I will be making changes to my final design to incorporate these and this will be in the final report.