Third Year Project 2013/2014

Report 2: The use of Layout Managers for resizable GUI applications solvers.

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

Layout managers are used to prevent user interfaces from changing dramatically when the window size is reduced or increased, they are needed so that the placement of a button or object position is not effected when the window is changed, what can happen is that there is a possibility that the button can be lost behind the window frame or hidden part of view, another problem could be if the buttons that are organised in a way that allows them to be in an order could be changed.

The way the layout managers work, despite there being many different strands of managers, they use code to dynamically position the object that will be held there, some are done by grouping, setting and linking components to keep them in a relative position.

Swing will automatically use the absolute layout, this means that it will allow the object to just be set statically but setting the position of the object using integers to specify location. This can cause problems with resizing the window later.

The types of managers are used to help the user in many different situations, for example in my project I will be focusing on the Grid layout and the group layout, I will go in depth on these later, into why they are effective and will help the project.

The general layouts are:[1]
Boarder layout is used for simple allocation of components to areas within the window, this will include 5 main areas, the title top bar (page start), the 3 main information blocks that are allow the object to be set to the line start, centre and end, there is also the bottom page end with the last bar, this is all done by pane.add(button, boarderlayout.PAGE_START);.

Box layout used to display buttons in the same axis but making their centres or a set position to be inline, this allows buttons to be in a row or within the same line. The box layout arranges all the components inside from the top to the bottom by making each match the same size that is what is the desired size of the object to fill the space. This is the code to set an object to a box layout pane.setLayout(new BoxLayout(newPane, BoxLayout.X_AXIS));.

[1] http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html Last accessed 7/11/2013

Flow layout puts the components within a line that follow one after another, it is used to allow a series of buttons and means they are tightly organised and follow the same flow, this is done by simply adding a new object to the flow layout.

The group layout can be used to group components like buttons together by connecting the content by adding them into groups, this is done by using different groups for items and generic settings effect all of the items in that group. Eg. Here there is a content pane from a user interface built by window builder, this creates a group that then uses the .addGroup function to add items to that group and present it using the .addGap function.

```
.addGroup(contentPane.createSequentialGroup() //creates a group that follow in a sequence.
.addGap(100) //the size  of  the gap between components.
.addComponent(btnSolve, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)// adds the button and puts it into the group layout using default sizes.
```

Finally the grid layout, used in this project as a way of keeping the buttons in a visual grid, this is done by adding components to the grid layouts 2D array that allows it to display everything in the correct position including to specify which grid location. To great the grid a new instance of the layout is created and then the components are added to that.

```
GridLayout sudokuGrid = new GridLayout(0,2);
sudokuGrid.add(button("1"));
```

This was the best way for me to create a Sudoku grid, filling in a 2D array of buttons that the user can then use to select and inter the integers used to play the games.

There are a some extra features of layout mangers that allow customization for the user, these methods are used within selected managers: [2]

A property that all components can have is a preferred size, which sets a size that allows the user to alter and when possible the object will be set to that size. This is idea as it allows the user to specify I size but the button is not limited to those specifications.

Glue is used within layouts such as the box layout, allowing the user to add in a glue object between two components and this provides a gap between the components but with a link that is unlike a rigid link that is unable to change, unlike the glue function that allows the components be kept in a distance but not effected if they are moved to other positions.

There is a function that allows the alignment of the components to be set to allow it to be viewed in a different way this is done by

```
button.setAlignmentX(Component.LEFT_ALIGNMENT);
```

Layout mangers are used so widely and are used more commonly than absolute placement do to the many advantages that it provides both in the design and creating the GUI stage but also within the program and its resizable windows. The layout managers allow the buttons to be dynamically changed if needed so that they are kept at the same size and within relative distances to each other given the boundaries.

The layout system is very easy to use even for a new programmer to get used to, it is possible to simply add the component to the layout and this will then mean it is

[2] http://docs.oracle.com/javase/tutorial/uiswing/layout/box.html

connected to its controls. However it is also possible to use programs like Window Builder Pro that allow the user to create it using a faster and more visual experience where the layout restrictions can be visible instantly.

Layouts allow provide as much of a visual aid in terms of grouping the components and allowing them to be set quickly and within the correct placements showing them in the order they are added as well as positioning them with the correct glue in between each component.

One of the biggest advantages of layout managers is that they provide a simple template for each window that can be exactly the same on each window no matter the size there will always be the same positioning on each window. So this really helps to create a consistent and well presented user interface with minimal effort as a lot can be handled by the layout manager.

With those positive outlooks on managers there are also problems that occur when using them and these limitations include the what each layout manger can actually allow, there are different forms that can be used for many different ways but the use of one could hinder the developer as there might need to be a use to keep a component somewhere that the manger does not permit, this can be resolved with absolute positioning but then does not get the benefits of the layouts resizing capabilities.

Through my experience when using layout managers adding another component later on can really change the rest of the layout within a button system for example, when there were 6 buttons and they were perfectly arranged with the correct spacing in-between, I later added another button and a object next to the buttons the spacing between the buttons changed but not in a way that looked presentable, and the buttons were shifted much lower than expected, however this was easy to remedy by changing the code and the gaps.

Within the GUI concept program I have only used two of the layout mangers grid and group layout, this however will not be the case within the final project as I will be expecting to use a few others to allow the program to look and feel consistent.