# Human Computer Interaction - Coursework 1

s0903611 s0942340

October 26, 2012

## Introduction

The specification of the task is to design an 'Image Labelling Application' in which users can identify, mark and label objects in images. In order to complete the task, it was divided into subproblems and organised into stages in order to approach the task in a hierarchical and structured manner.

## Task Analysis

Task analysis was split into two stages: first analyse the task itself, then the previous attempts to solve this task.

### Hierarchical Task Analysis

Performing hierarchical task analysis requires the goal (the labelling of images) to be split into smaller subgoals which themselves are analysed, and if necessary are further split into subgoals. The results of this analysis are as follows:

### Application Critique

The next goal was to analyse and critique previous solutions in order to understand what ideas we can use and what functionality is missing.

#### LabelMe

The LabelMe application has a lot of features which work well. The labelling system is very simple and intuitive: once the image is loaded it requires no extra commands besides clicking on the image. The labelling box automatically appears when a shape is completed and it requires only one action to review a label.

There are a few small changes which might make task faster: being able to move back to images, as well as the current functionality (forwards). This allows for quick reviewing of the previous image in the situation where the user accidentally moves forward. However, in the application there are no methods

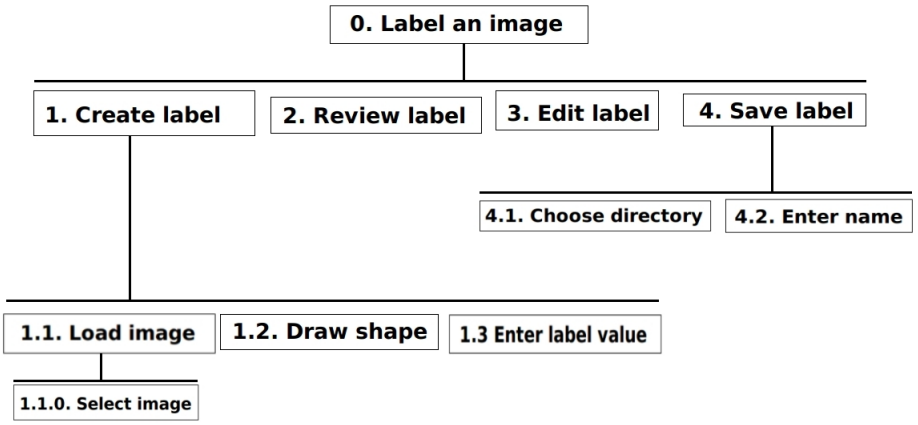


Figure 1: Results of hierarchical task analysis

to load an entirely new image into the application (if it hasn't already been added to your folder), which slows the process down.

### Demo

The only functionality the demo example provides is image loading and the drawing of the shapes. The drawing functionality is similar in function to the LabelMe application: clicking creates vertices, between which lines are drawn. However, shape ending is not as simple or obvious as LabelMe and this makes the first time use very confusing. The lack of buttons and of any guidance in the demo example was unintuitive and as an user, it was difficult to gauge whether or not the demo example offered any more features other than allowing new polygons to be created.

## General Considerations

The demo had implemented some simple functions but to allow more flexibility in design it was decided to start from scratch, using the drawing and image loading functionality as needed. Java & Swing were adopted as the programming language of choice for the application as familiarity with Java and the IDE Eclipse meant that implementation did not require learning of a new language or developing environment. Swing however, was unfamiliar for both of us and learning Swing implementation was a tough learning curve during the implementation. Since the code base was started from scratch, this made us learn the implementation basics of Swing thus further extensions were understood easier.

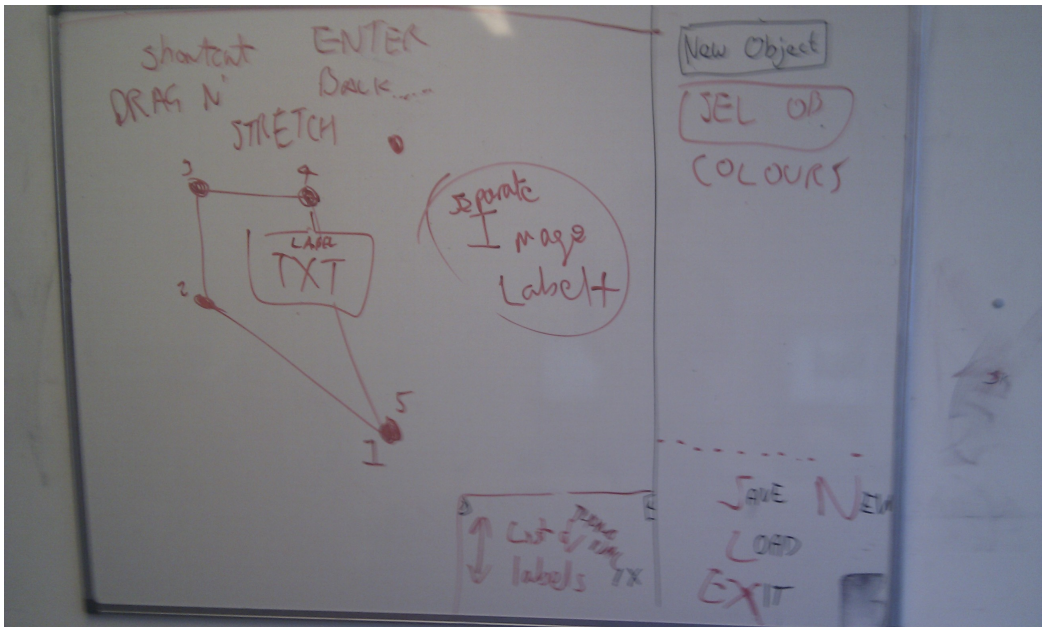


Figure 2: Wireframe prototype on a whiteboard

## Design

Wireframe paper prototyping was used to visualize the layout of the interface and to further realize other features that were required to be developed for the user. This allowed for rapid and inexpensive designing of the layout as thoughts and ideas could be illustrated on paper to illustrate design ideas. Once the layout was fully visualised and understood, the next stage was to design what functionality and how to bring the layout to a prototype.

Use cases aided the design process as they helped create scenarios where the user would expect certain events to execute if they performed an action. For example, if a colour palette was available to the user, we discussed what would be the consequences of choosing a colour and why this sequence of execution would be intuitive to the user. Once a use case scenario was developed, the task was transformed into a hierarchy of subtasks which are necessary to perform the root goal. For example, if the user wished to load an existing annotation file, sub tasks that could be included in the task hierarchy could be clicking on a 'Load Annotation' button, which pops up a window that shows a GUI file explorer and so forth.

This allowed exploring each detail and any possible actions with further breakdown of subtasks, to eventually atomic actions. A variety of different uses cases were discussed and developed, providing an initial criteria list and design scopes for the application. Example of design scopes included were allowing a colour scheme for the user to label icons with in order to categorise labels, a

label list with thumbnails displayed to the user and the choice to edit previous labels, be it the label name, position of vertices, colours or even delete an entire label.

During design and implementation we strived to comply to general HCI guidelines, specifically Shneiderman's Eight Golden Rules of Interface Design[1]. For example, we have tried to reduce the possibility of errors, like attempting to display a non-image file.

### Creating A Label

After testing out and getting to grips with the demo example, design ideas and expectations were discussed in order to design an intuitive interface for the task. Designing how the user creates a label was the first step as it is one of the main features required by the application and would be used frequently. In order to complete the current polygon, the demo example used the button 'New Object'. This would complete the polygon causing the final edge to be drawn. During experimentation we found this to be limiting. Following Shneiderman's Golden Rules we added the return key as a shortcut to end the current label.

The next stage of label creation is the label text. It was decided that this should be as simple as possible, since this is something which the user will always want to do and will be doing frequently. So a popup is created which prompts the user for the label name, with a cancel option if they completed the shape accidentally.

### Review Labels

Once a label has been created the user can view the polygon on the image, or by looking at the list of labels on the right with gives the label name and a thumbnail representing the location of the polygon. To ensure consistency, the thumbnail updates whenever the user moves or deletes a vertex, allowing them to quickly view the accurate positions of their labels. When using LabelMe it was difficult to know at a glance which shapes referred to which parts of the image. For this reason we decided a thumbnail system would allow the user to quickly understand which of their shapes correspond to different locations on the image.

### Edit Labels

To allow for easy error correction there are multiple attributes of a label which the user can edit: vertices, colour and the label name itself. We decided the simplest way to do this was to allow the user to click on drag the vertices to change their position (rather than, for example, deleting and remaking). This is intuitive and simple for the user because it allows them to see the shape changing as they move the vertex. If the user misclicks a vertex and subsequently creates a new vertex for the next label, they can press the Delete or Backspace key to

[1] http://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html

remove the incorrect new vertex. The other two aspects of editing are done via the label list by selecting the shape are clicking the relevant button.

### Saving & Loading Labels

The first decision to be made on file saving/loading is what data we want to be performing this on. To keep things modular it was decided to treat images and labels as separate objects: an image and an 'Annotation Session'. This also follows the user's understanding of the application: the user is adding metadata to the image and is not editing the image itself. Loading sessions would restore polygon positions with the vertices, colour and label name, as long as the annotation sessions were not drawn on a image of higher resolution. This prevents vertices being placed outside of the image frame which would be considered an error for the application. Through our own usage of the program it became apparent that you could very quickly lose a lot of information if you loaded a new image or annotation session. This would be potentially frustrating for a user and would require many steps before the user arrived at the same state. To prevent this, we decided to keep track of when the user had made changes without saving: then prompt them when they try to load image/labels without first saving. To allow the user greater control they can ignore this warning and proceed to load.

### Quick Load

One of our criticisms of the LabelMe application was the slow process of adding images. To enable fast processing of many images we added the two arrow buttons which switch between images in the working directory (this directory is defined as where the last image was loaded from). This means users can put all their images in oen directory and then don't need to deal with manual loading.

## Implementation

The implementation plan was to develop in iterative cycles, building from simple components and atomic actions, prototyping each stage of the application and if necessary, reverting back to designing if the implementation was not suitable. This eventually leads to integrating each part together to create a functional and intuitive interface to the user.

Developing in iterative stages allowed rapid development of ideas and more importantly, quick changes to design ideas if the original design was not as good as originally expected. For example, the original idea of using 'New Object' to complete the existing label and to allow a new label be created offered too many possible actions to the user and the button appeared to have multiple uses: ending the previous shape and starting the next. In order for users to move existing vertices, users had to click on a 'Move Vertex' button which only allowed users to drag existing vertices to new locations. To switch back to

drawing new vertices, the user had to click on 'New Object', even if they were still editing the same object.

## Implementation: Testing

After real users trying our application these problems came to light. Testers would be asked to demo the application by having the application on a monitor, with a keyboard and mouse, and try to figure out what the application was designed for and how to interact with the features available. The tester received little interaction with the demonstrators who were only there to observe the problems. After they had finishing demoing the app (their own choice when to stop) , we asked each tester what they thought of the layout, how simple it was to use and what difficulties they had. The primary issue was the 'New Object' and 'Move Vertex' buttons as it was unclear what they meant, and what the outcome of pressing them would be. Other comments included poor use of the colour scheme (during implementation, each panel was coloured differently to make clear distinctions between panels) and the file loading system was unclear in loading images or annotation sessions. The criticism was taken on board, and thus the 'New Object' and 'Move Vertex' buttons were removed and the vertex placing system was redesigned as a result of live testers.

## Implementation: Iteration

The vertex placing system had been redeveloped with the mouse and keyboard in mind, allowing the user to seamlessly switch between moving existing vertices and drawing new ones. The user can click on points where no other vertex lies to draw a new one, or immediately drag existing vertices to reposition them elsewhere on the image. The use of testers during the design and implementation stages was invaluable in refining and building the application. In addition, when the user is drawing vertices for a new label, they can complete the polygon by using the usual method of pressing Enter on the keyboard, or clicking on the first vertex to join up the edges to complete the polygon.

### Formal Analysis

In order to critique our own work, we decided to evaluate it in terms of pros and cons with some comparisons to the established LabelMe application.

### Pros

- The thumbnail in the label list works well and captures an accurate snapshot of the original image. Users can tell which labels correspond to which polygon.
- Vertices of a shape can be easily manipulated: be deleted, moved elsewhere or have their colour changed. The highlighting of the polygon allows the user to see what area is outlined by the shape. The user can manipulate

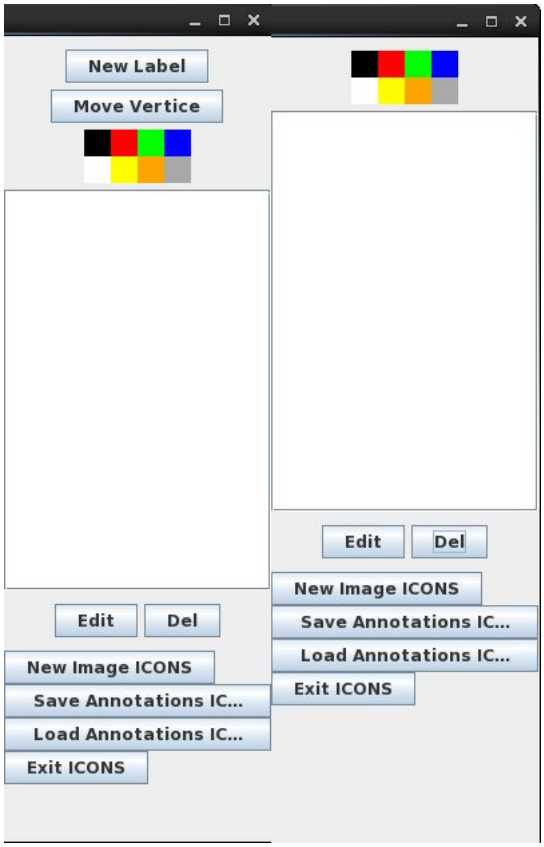


Figure 3: Example of iterative design

the existing vertices much quicker than using LabelMe, which requires the user to first enter the 'Adjust Polygon' mode before allowing vertices to be repositioned.

- Users are prevented from making errors to cause the application to fail as appropriate error handling has been applied where necessary. For example, if the user is prevented from drawing vertices in null space (if an image is too small to occupy the entire image region or has been scaled to fit comfortably in the image region).
- Users are alerted with a pop up box if they are about to perform an operation such as deleting a label, or if they are about to select a new image and have not saved their current labels.
- Use of keyboard shortcuts allow quicker access to shape completion unlike LabelMe which does not support keyboard use.

**Cons**

- If a user decides to return to an existing label to expand it, there is no method to allow new vertices to be added to the label.
- If an extremely wide image is used, scaling causes the image to be drawn in a skewed manner.
- There is no undo / redo mechanism in the application to allow the user to recover from large mistakes, such as deleting a label by accident. In this case, there is no easy reversal of actions.
- Although users are allowed to move existing vertices of incomplete shapes they are not allowed to move the first vertex. This is due to an implementation of what is considered to be a complete shape, as clicking on the first vertex will ask the user to enter the label name and complete the shape. Therefore users can only interact with the first vertex once the shape is complete.
- There are some known bugs that exist within the system that have been marked in the bug log.

**Bug Log**

Due to time constraints and being Swing novices, there exists bugs within the system.

- Using images with transparent backgrounds (and of course, being in a file format that supports transparent backgrounds) causes side effects when moving vertices in the (transparent) background. This causes the redrawn edges and vertex to appear and stay in the background.

- When minimizing the application window and then restoring the application window, the vertices are not redrawn on top of the image. Only when clicking on a completed label in the list, changing colour or completing a new label will the new vertices reappear.
- If the application window is dragged off screen when highlighting a shape then swing calls paintComponent in VertexPanel which causes repainting of the transparent polygon. However, since swing (for an unknown reason) doesn't also call paintComponent on ImagePanel then the polygon gets darker. This eventually leads to the polygon highlighting to become a solid colour and obscure the object.

## Future Work

Features that would have been liked to be implemented are the following:

- Addition of future vertices to complete labels
- Undo / Redo feature
- Aesthetic improvements (e.g. better line drawing)
- Scaling any images without any skewing effects