MapMyNotes

Final Report for CS39440 Major Project

Author: Ryan Gouldsmith (ryg1@aber.ac.uk)
Supervisor: Dr. Hannah Dee (hmd1@aber.ac.uk)

4th March 2016
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Acknowledgements

I am grateful to...

I'd like to thank...

Abstract

Include an abstract for your project. This should be no more than 300 words.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Background

Handwriting notes is still considered to be an important aspect of note taking. Smoker et al [7] conducted a study comparing handwritten text against digital text for memory retention and out of 61 adults 72.1% prefered to take notes using pen and paper, rather than on a computer. Smoker et al concluded that recall rates for handwritten text was greater than that of typed text proving that handwritten notes are better for a user's memory retention.

However technology has advanced and we're moving into an era where we track and view everything digitally, from email to calendar entries. Therefore, there is a need to transfer the productivity from handwritten notes to digital notes - so they can be located easily.

1.1.1 Taxonomy of notes

When a user creates a note they will often come in differing forms: some are semi-structured and some are back of the evenvelope kind of notes. When thinking about an application to analyse notes, first there has to be consideration for what a note will consist of. A taxonomy, by definition, is a biological term for a classification of similar sections, showing how things are linked together.

Notes can be throught of as a collection of similar classifications, whether this is the pure textual descriptions of a note or whether this is purely pictoral form or a mixture of both. However, the notes are normally split into three distinct categories:

1. Textual descriptions
2. Diagrams
3. Graphs

In figure 1.1, it shows a taxonomy of the different aspects which may comprise to make a note. Textural descriptions form the core part of a note, this is essentially the important content that a note-taker is trying to remember and write down. Different note-takers form their notes in different ways, for example the headings may be underlined or hashed - if they were adopting a mark-down style approach. These sections helps to show that there's a break in the content, and should be

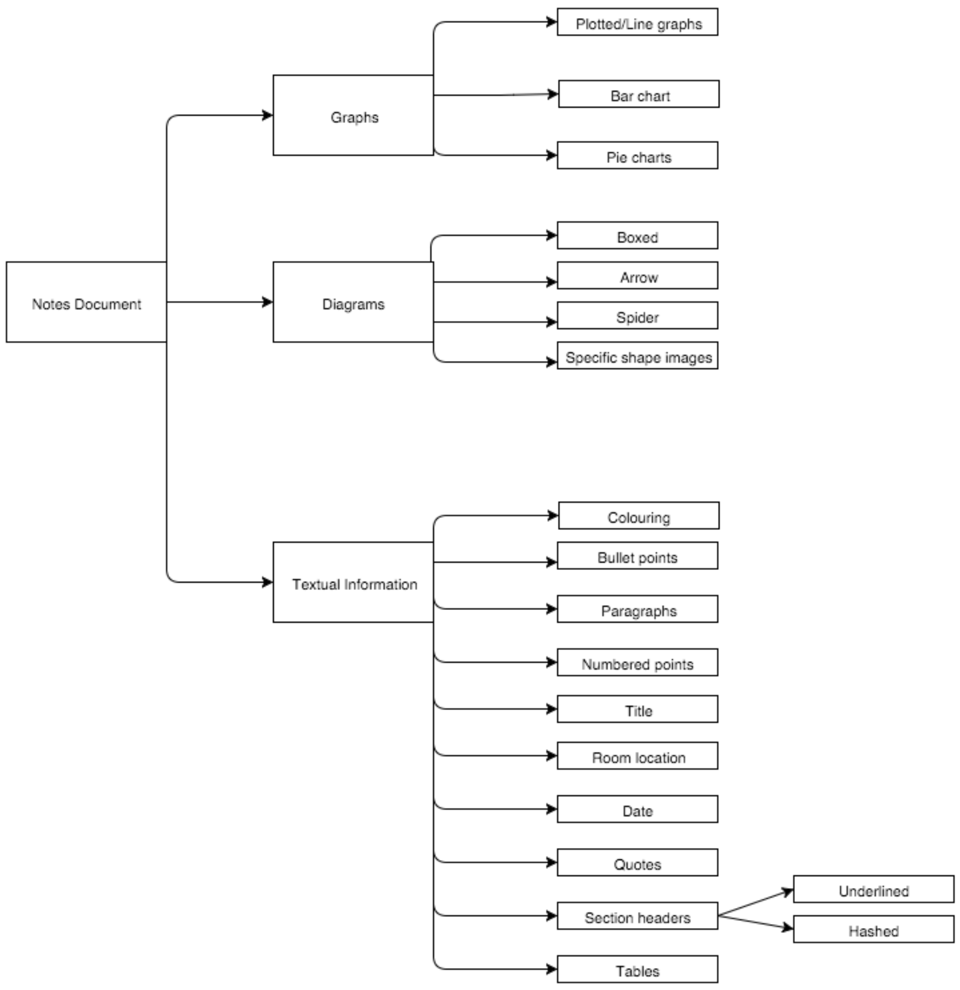

Figure 1.1: A taxonomy showing the structure and classification of different types of notes and what is contained in a note.

sub-sectioned. Textural points that are short, but important, are often characterised by a colon or a bullet point; these are the most common form of concise note building, in the classisifcation. Coloured highlighting if often used for a variety of reasons: it stands out on the page and for some people colour retention is better. Finally, tables help to represent textual content in tabular form. This is often good in notes for comparisons.

Graphs are great visual tools for user's in helping to convery textual information in a graphical form. Naturally, they have their limitations such as they come in different shapes and sizes, such as a line-graph, pie chart or a bar chart. Coupled with graphs, notes often consist of diagram drawings. In figure 1.1, there are different sections and classsifications of a diagram: boxed, arrow etc. Each one has its own purpose and arrowed and box can overlap; UML diagrams are a case of this. Spider diagrams are probably the hardest to represent, due to the varying sizes and whether the user draws circles or clouds. Furthermore, specific shape diagrams are conceptutally hard to think about as it depends on the domain in which the user is drawing the note. For example, a person in biology may draw a stick person, whereas someone in Computer Science may draw a PC.

Overall, identifying the taxonomy of notes is important as it gives context to which the application can potentially parse and identify these different classifications. Additionally, it acknowledges where there may need to be constraints due to the variations in notes and how it can deal with semi-structured documents.

1.1.2 Similar Systems

With note-taking on digital devices becoming more of the normal then software tools have been created which help users to write their notes down, edit. These are mainly WYSIWYG (What you see is what you get) editors - which a great deal of flexibility. When evaluating existing systems two were commonly used and they were:

- OneNote
- EverNote
- Google Keep

1.1.2.1 OneNote

OneNote is a note-taking and organisational application made by Microsoft [**?**]. Commonly, this is a WYSIWG allowing you to add text, photos, drag and drop photos. This offers a wide-range of flexibility when making notes. In recent times, OneNote has developed functionality to analyse a user's handwriting, from say a stylus, and interpret the text they entered [CITE]. [CITE] In OneNote you can insert a note into document and then it would interpret the text from the note. They offer a wide range of product support from mobile based applications to web versions of their software. Office Lens [CITE] can be used in conjunction with the OneNote to help to take photos and automatically crop the image and then save them to OneNote. This feature is important and should be considered when thinking about *MapMyNotes* application. The process requires you to have a microsoft account, which you sign in with. Once in they format their documents as a series of "notebooks" which group together similar notes. When editing notes in the input boxes then it does automatic saving behind the scenes, so you never have to click a save button. When using OneNote it feels very much like Microsoft Word - with the similar layout that gives most users a similar user experience feel.

1.1.2.2 EverNote

EverNote is a note-taking and note organisation app, it is both supported on the web and in app form. EverNote is widely used and would provide a lot of the functionality a user would need to upload their note. They have realeased development articles [CITE-2013 one] stating that they are able to do OCR recognition on images. This would allow the user to upload an image and it would give a list of words, which it thinks is the word identified. This differs from MapMyNotes where the author aims to develop an application which would give a 1-1 word comparision, rather than a list of words. Like OneNote, the notes are collated into Notebooks, offering a WYSIWG editor, giving you full control of the content that you type in. When uploading an image, to the web version, it gives you the option to edit the PDF and images, however it seems as though you have to download another application, specific to your platform, to do this functionality. According to the website, it does to OCR recognition, however whilst using the application I could not find an easy way in the user interface to get the any text or event find out what a word was. This is something which is important and key for *MapMyNotes*. Additionally, there seemed to be no way to save the note to a calendar item - only the option to send via a link.

1.1.2.3 Google Keep

Google Keep [CITE] is a note taking application made by Google with mobile and website support. Google keep allows a user to attach an image to their note and attempt to extract the text from an image and save this in the body of the text. It allows you to tag a title, and add an associated body. This is note a WYSIWYG editor, it's a plain old text box which a user can input information into. They have the option of a remind me functionality, which will get synced to their calendar as a reminder - but there's no easy way to add it to a calendar event. Google keep seems as though it's more for TODO lists and jotting down quick notes, rather than an archiving tool suitable for note taking. Although, the tagging with labels is a nice feature and the filter by image is a smart tool - to only show notes with images. The simplicity of the user interface and the ease in which text could be extract provides good thought going forward.

These three existing products are widely used by the every day note taker. They have been developed to a high quality and give the user a full control experience of what their notes can consist of. The automatically cropping of an image is an important process and should be considered when for the application going forward. *MapMyNotes* aims to try and give the user full control of their lecture notes content, so that they can find their notes easily again.

1.1.3 Personal Interest

The author has a motivation for a project which can integrate with a calendar and automatically store notes. Notes are often written up but discarded soon after the lecture and not found until weeks before an assignment or examination is due. There is then a frenzy to try and find all the notes for a given module which are often dispersed throughout many pages in a notebook. The tool would be useful to take a photograph on a normal phone, upload it to the software service, and then attach any associated meta-data and save it to a calendar. This would help to reduce the chance of lost notes and notes which are unreadable or split due to page breaks. Additionally, with it being in the calendar all notes will be associated to an event so quickly searching for associated notes on a calendar item would return the correct notes easily. This would reduce the time spent searching for notes and actually finding the note again and getting on with the purpose in hand.

1.2 Analysis

Due to the project originally being proposed by Dr Harry Strange, meetings were arranged before he left the department, to discuss what his initial ideas were of the application. It was here that it was confirmed with the author's personal interests, that this was a problem that needed to be solved. Looking at the existing systems it was clear what a good system would entail. The problem highlighted that there needed to be an application, which would allow you to upload a note to the application and view it and ensure that it saves it to the calendar.

By analysing the problem a taxonomy of notes was collated. This helped to breakdown exactly what was in a set of notes. This taxonomy gave a comprehensive analysis of what information would be available to parse from the test, as well as giving the information parsed semantic meaning. It was quickly brought up that I should potentially create a specific layout for the note so that the data can be parsed sensibly, to reduce the complexity. Additionally, this highlighted all the possible things which the application could parse. Due to text being the main component of a note, it was agreed that efforts would be focused on parsing the text. Images could have been extracted from the notes but this would not form the core aspects of the notes and was placed as a task that could be implemented in the future. This lead to the task of investigating how to extract the text from an image. It was agreed to look into OCR tools and find a suitable tool for the application - there could have been the approach of writing my own handwriting recognition tool using Machine Learning, but that would not solve the problem it would only help to solve part of the problem as a result it was agreed the application was more importnt than this research, so an OCR tool was agreed to be used.

From the Google Keep example, it was clear that analysising the text would be an imperitive part of the application, this lead to the task of having to work out of a way to analyse characters on the image. Initially, the idea was to analyse all the characters on the page and parse them, like Google Keep, however this would be far too complex. Due to the complexity this had to be scaled back and we would have to analyse the meta-data from the note, i.e: Title, lecturer, date etc, this became the primary aim of the tool. Additionally, due to analysing handwriting being a complicated process to begin with then it was agreed that it would only be trained on the author's handwriting, with the possibility of extending it in the future - this was chosen due to help reduce the complexity of the application and simplify the process within the timeframe.

In addition to this, it was clear that when trying to analyse the image a certain structure would have to be applied to the notes to help to aid suggestions. As a result a set of rules would have to be collated to ensure all notes were structured in this way when using the application. As already mentioned, this would help to ease the complexity of parsing the notes and reduce the need for Natural Language processing, which would increase the complexity of the application again beyond the ability to complete it in the timeframe.

During the meeting with Dr Harry Strange, it was clear that the tool would need to be used on the go, potentially where a laptop access was reduced. Trying to make the application available to everyone a web application as a software as a service, would be created. This was chosen instead of a mobile application due to the accessibility which a web application can give you - especially if it's responsive. Additionally, from the background research conducted all three of the afformentioned products have accompanying web applications.

Next it was discussed what the web application should actually do. From looking at the applications, it was clear that a view all notes, search and add-edit , delete (the usual CRUD application) would be needed to help to make a full system; all the examples looked at contained this functionality so it was important that I acknowledged that this was an important thing. Reflecting on the premise of the application it was analysed that the best way to search for notes would be by module code, as most University students would want to find specific module notes. This created the tasks of being able to search for the module code and show all codes for that user.

One thing that was noticed very quickly, from the research, was a lack of calendar integration. Reflecting on the conversations with Dr Harry Strange, this was an important feature for him. The challenge of deciding which calendar to integrate with was an important decision. From a survey

[CITE SURVEY] Google calendar is considered to the most popular app - so it was decided that Google calendar integration was needed. Other calendars such as Microsoft's calendar was considered but realistically there was only time to implement one calendar integration. Therefore, the application had to save a url of the saved note to the calendar event for the given day the user enters.

1.2.1 Objectives

As a result of the analysis of the problem, the following high-level requirements were formulated:

1. Investigate how to extract handwriting text from an image - this will involve looking into ways OCR tools can interpret handwriting.
2. Train the OCR to recognise text of the author's handwriting.
3. Produce a set of a rules which a note must comply to.
4. Produce a web application to form the core part of the product. This includes allowing a user to upload an image, display the image. Add appropriate tagging to a note such as module code.
5. The user must be able to search for a given module code, shwoing the fill list of notes based on the module code entered.
6. The backend of the application must conduct basic OCR recognition, analysing the first 3 lines of the notes.
7. The backend must integrate with a calendar to archive the notes away later to be found again.

Ideally, having image extraction and making it into a WSIWYG editor, like OneNote, would have been the desired outcome from the application. This would incorporate the analysis of the text from the image as a whole and display coloured text. After discussions with my supervisor, Dr Hannah Dee, it was clearly stated that this was far too complex for the timeframe so a re-think had to be conducted. Initially, my supervisor thought that the web application , excluding any OCR recognition, would be suitable for a minimal product and a dissertation. Although this was sensible,

During the first few meetings my supervisor, Dr Hannah Dee, it was discussed what was actually plausable in the time frame. Acknowledging that the project was large and could be expanded we opted to rein in the features and get a minimal project. Dr Hannah Dee suggested that the just the application without the Tesseract parsing would be enough for a minimal product, however I wanted the ability to try and recognise handwriting recognition. So we comprised but mentioned that the handwriting recognition would be a background process, and not the main aim.

1.3 Process

Software projects tends to have a degree of uncertainty regarding their full end outcome regarding requirements. *MapMyNotes* has a great deal of uncertainty regarding how long certain tasks would take, as well as thinking of new functionality from customer meetings and user testing feedback. Tasks such as training the author's handwriting could not be truely estimated due on a Gantt chart due to a degree of uncertainty regarding how long it would take. As a result, a plan-driven approach, such as the Waterfall model, would not be suitable for this project where the requirements may change from week to week. Instead, an Agile approach was followed throughout the process.

Scrum[CITE] is normally a methodology for teams to help to guide a team to be the most productive as possible. Due to this being a single person project the Scrum approach has to be modified to work in this scenario. Work to be completed is separated into sprints, which shows what work needs to be completed in the sprint. Sprints can vary from 1 - 4 weeks in length but a shorter sprint allows feedback to be gotten back to a team quicker, allowing more change to occur. Scrum uses user stories to help to make sure the team is always doing client-valued work - these stories are collated and put into a backlog, which forms a list of all the user stories for the project. At the start of each sprint user-stories are pulled from the backlog and placed into the sprint, each one estimated. At the end of the sprint a review and retrospective is conducted to analyse the sprint, identify what was good and what was not so good so the next sprint can be improved.

For the single person project, this methology was used and adapted. Sprints in the project were a week long, which coincided with the weekly supervisor meeting on a Thursday. A backlog was collated at the start of the project to reflect the initial stories. These were summarised as epics (a high level interpretation of a story). Each story was estimated accordingly not on time, but on the level of complexity it would take to complete the task. The tasks were estimated against a "goldilock" task, which is a base task which all other tasks are estimated around. I.e, if you estimate that the complexity of the research Flask's routing system is a 5, all other tasks will be compared against this for complexity. During the planning stages I adopted the planning poker [CITE] technique; user-stories are estimated on a scale of 1, 2, 3, 5, 8 and so on. If a task was usually 15 or above then it would be reflected upon to ensure that I fully understood the scope of the task and where it needs to be broken down into smaller tasks which would reduce the complexity of the overall task. At the end of the sprint a review and retrospective was conducted, instead of face to face with a team it was a reflection in the form of a blog post [CITE WEBSITE]. These were used to reflect on what has been achieved in the sprint, what went well and should be continues and what could be improved upon in the next sprint. Furthermore, I used this reflection to do some pre-planning, after my meeting with my supervisor, to decide what would be brought into the sprint currently. At the end of every sprint, a conversation between myself and my supervisor was utilised to determine what needed to be completed in the next sprint. Once we had decided the user stories which matched, at maximum, the last sprints story points were brought forward to the sprint. For example, if 20 story points were completed in sprint 3, then 20 story points would be allocated for sprint 4 and the user stories brought in would have to add up to 20 points. The project was managed on the open source management tool Taiga.io [CITE] which was invaluable, and provided built in functionality such as burndown charts per sprint, which show how well you're completing story points and tasks. These were used in the sprint to analyse how well work was being completed.

Coupled with Scrum, there was a modification to the development process to incorporate Extreme Programming principles[CITE EXTREME PROGRAMMING]. Principles such as: merciless refactoring, continuous integration and test-driven development were used. Test-driven development (TDD) is the art of writing the tests prior to writing the code. Using this principles ensures that tests were written for functions and methods, as well as helping to think about the design prior to the implementation. [CITE] agrees that tests an additional form of documentation. With TDD it follows three cycles: Red, a failing test, green, a passing task, refactor, refactoring the design. The aim is to do the minimal design needed to pass the test; this approach was adopted throughout the project.

Continous Integration tools were a core part of the process and life cycle on this project. When the code was pushed to the repository a build script was triggered in the continous integration software to build and test all the system to ensure they worked in an isolated environment. Finally, CRC cards [CITE] were used during the design section to consider how implementation between systems would interact and how different classes would look and feel. This principle from Extreme Programming helped to keep the design simple and not convoluted for the task, focussing on the current task.

Chapter 2

Image pre-processing for Tesseract

2.1 ImageMagick

In order to produce the best possible results with Tesseract, there had to be a pre-processing stage beforehand. Initially, the image was converted to grayscale using ImageMagick[CITE] but this showed that training was difficult and was not yielding the success rates which was needed.

Pursuing using ImageMagick for the pre-processing stage the image was converted to Monochrome. This managed to be able to pick up more characters in the image, but in doing so picked up too much noise around the image. This distorted the characters and made it hard to recognise characters with the correct confidence that Tesseract required.

Dr Hannah Dee suggested that maybe OpenCV[cite] image manipulation would be useful for the pre-processing step, instead of using ImageMagick.

2.2 OpenCV

OpenCV is an open source image processing library. There was a choice of using the C++ or the Python bindings. Due to the applications core infrastructure it was agreed that Python would be easier to implement.

An investigation began by looking into the OTSU binarisation method and other versions of the adaptive thresholds. The need for a binarisation method, instead of using the default image, is that it reduced the ambiguity of the light sources on the image and any shadows which may occur during the capturing of the image.

After doing some investigations into different thresholding algorithms it came to light that Tesseract OCR uses OTSU thresholding as its underlying image processing algorithm. This quickly explained why a lot of characters were not being identified from images, when looking at the output of the experiment that were conducted. From this it could be seen that using OTSU as a pre-processing step for Tesseract would not return the best results.

Looking at the experiment of different thresholding techniques it was clear that adaptive mean and adaptive gaussian returned the image clearly binarised. There is no shadow or dark patch over the image and the text on the page is legible.

2.3 Line removal

After using Tesseract to train my handwriting on plain paper it showed that some sentences it could parse accurately, but those which are on a slant or slightly skewed, because of the lack of lines, returned a worse recognition rates.

To try and straighten the lines of text normal lined paper was used, but quickly found this was not good when binarising it as it often left unconnected lines which were represented as a series of dots. If the erode function was used on the image, it would lose the quality of the characters correctly identified.

This led to a creation of my own lined paper, one which would have thick lines and one which the lines could be filterable. The premise of using the blue lined paper was to correctly identify the blue lines and remove them from the image, leaving uniform lines of text which could be analysed. This process went through a couple of iterations.

To begin with it was decided to just extract the text from the image which fell between a black to gray threshold colour - excluding the blue lines from the image. This was followed up by the morphological operation, erode, to remove some of the remaining pixels left over by the lines. This worked to an extent and showed the binarised image with some loss of pixels on the characters and some were completely undistinguishable.

Although this worked well, it wasn't the most suitable solution as often line pixels were left in the image and would be picked up by the OCR resulting in an erroneous output from the system. So, again, it went through another iteration.

The next solution would binarise the image more elegantly and would actually remove the lines from the image and clear up the image to just leave the image binarised, with no blue or black lines.

[CITE REFERENCE USED TO DO THIS] Firstly, it would read the image as grayscale and then apply a median blur to the image. This was needed to try and ease the noise and smooth the image. Much like the first iteration adaptive threshold with a gaussian mean was used to binarise the image. Afterwards, a mask was collected containing the horizontal black lines, using the structuring elements "MORPH_RECT". This was eroded and dilated to try and remove the lines as much as possible. Intermediate masks so that black text was passed onto a new canvas - aiming to get as much of the text from the images across as a possible. This naturally carried some of the disjointed lines across, and suffered the same issues the previous iteration had to deal with. However, this iteration includes the use of finding contours.

The contours were considered using because a way was needed to find connected components and filter out the lines. Naturally, if the lines have been dilated and eroded then they will not be connected closely. Therefore, by choosing connected components on the image it will correctly get the characters from the image. Once the connected components of the characters were identified these were transferred to a blank mask, with a final erosion to clear up the image.

This technique has worked well in removing the blue lines from the page, whilst keeping the character quality. It clearly shows a binarised image with the text on the page clearly identified, and it worked on multiple examples. What is more impressing is that because of the adaptive threshold and different morphological operations then it can clearly identify characters in terribly lit photos.

Chapter 3

Handwriting training with Tesseract

[CITE THAT GUYS THESIS]

Handwriting recognition is still an active research project and one which is constantly evolving. There are many complexities with handwriting such as whether to analyse cursive or non-cursive text. The project could have been taken in a couple of direction: write my own handwriting recognition system or use an OCR tool. Since the premise of the application is to provide a software tool for a user creating my own handwriting recognition system is not viable. As a result, an OCR tool has been used.

Consideration went into the different OCR tools out there, with commercial vs non-commercial and there was one open source technology which seemed very reliable, Tesseract.[CITE PAPER ON COMPARISON].

Tesseract is an open source C++ library for analysing handwriting, the current stable version at the time of writing the report is 3.04. It is mainly interacted through a command line application, and that has what has been used in this project.

3.1 Training

To begin to analyse the characters on a page, there was a decision that the training process would involve analysing non-cursive handwriting. Although this limits the user experience, analysing handwriting is a challenging task in itself, so by making it non-cursive there was a better chance of good recognition.

Secondly, another caveat with training is that the training process would be consisted on the author's handwriting, and not on the general public. This again, limits the application to the user, but that could be expanded in the future.

In order to successfully, train the handwriting then the pre-processing steps described in chapter 2 are required. Outputted from the image pre-processing is a tiff file - which then used for the Tesseract training.

Firstly, the image has to be in the following format: ¡language¿.¡font¿.exp¡expnumber¿.tiff, for example the following file would be valid: eng.ryan.exp1a.tiff.

Chapter 4

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

4.1 Overall Architecture

4.2 Some detailed design

4.2.1 Even more detail

4.3 User Interface

4.4 Other relevant sections

Chapter 5

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 6

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on real users? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

6.1 Overall Approach to Testing

6.2 Automated Testing

6.2.1 Unit Tests

6.2.2 User Interface Testing

6.2.3 Stress Testing

6.2.4 Other types of testing

6.3 Integration Testing

6.4 User Testing

Chapter 7

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

Appendices

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [2]. The library is released using the Apache License [1]. This library was used without modification.

Appendix B

Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

Appendix C

Code Examples

3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [6].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*---------------------------------------------------*/
  /* Minimum Standard Random Number Generator          */
  /* Taken from Numerical recipies in C                */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity     */
  /* Always call with negative number to initialise    */
  /*---------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
    *idum += IM1;
  }
  k = (idum2)/IQ2;
  idum2 = IA2*(idum2-k*IQ2) - IR2*k;
  if (idum2 < 0)
  {
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
```

```
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```

Annotated Bibliography

[1] Apache Software Foundation, "Apache License, Version 2.0," http://www.apache.org/licenses/LICENSE-2.0, 2004.

This is my annotation. I should add in a description here.

[2] ——, "Apache POI - the Java API for Microsoft Documents," http://poi.apache.org, 2014.

This is my annotation. I should add in a description here.

[3] H. M. Dee and D. C. Hogg, "Navigational strategies in behaviour modelling," *Artificial Intelligence*, vol. 173(2), pp. 329–342, 2009.

This is my annotation.s I should add in a description here.

[4] S. Duckworth, "A picture of a kitten at Hellifield Peel," http://www.geograph.org.uk/photo/640959, 2007, copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

[5] Microsoft, "Microsoft OneNote — The digital note-taking app for your devices," https://www.onenote.com/, 2016, last checked 13 April 2016.

Used to look at and compare how similar note taking applications structure their application. Used the applicatation to test the user interface and what functionality OneNote offered that may be usedful for the application

[6] M. Neal, J. Feyereisl, R. Rascunà, and X. Wang, "Don't touch me, I'm fine: Robot autonomy using an artificial innate immune system," in *Proceedings of the 5th International Conference on Artificial Immune Systems*. Springer, 2006, pp. 349–361.

This paper...

[7] W. Press *et al.*, *Numerical recipes in C*. Cambridge University Press Cambridge, 1992, pp. 349–361.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content.*

[8] T. J. Smoker, C. E. Murphy, and A. K. Rockwell, "Comparing Memory for Handwriting versus Typing," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 53, no. 22, pp. 1744–1747, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1177/154193120905302218

Used to show that there handwriting is still an important part of memory rentention with note taking, compared to digital text

[9] Various, "Fail blog," http://www.failblog.org/, Aug. 2011, accessed August 2011.

This is my annotation. I should add in a description here.