COMPUTING LABORATORY SEMINAR March 8th 2011

Writing a 3rd or 4th Year Project Report

Dr Maggie Charles, Language Centre maggie.charles@lang.ox.ac.uk

1: PARTS OF THE REPORT

Title page

Abstract

A short summary of the whole report. Around 200 words is probably enough.

Acknowledgements (optional),

Some students put these at the end, which is more usual for published papers.

Table of contents

This should list ALL parts of the report, including References and Appendices

Chapters 1, 2 etc.

Areas to cover: Introduction, Background, Requirements, Design, Testing, Conclusions

These are the areas given in the Handbook, but you do not have to use these exact words in the chapter titles. Titles should be informative for the content of your own chapters. You may also find that you need more chapters, or that you can combine certain areas into a single chapter. Your choice of how to organise the chapters should follow from the nature of your own material. You will probably need about 1,000 words for the Introduction and the same, or less, for the Conclusion. The rest of the chapters will vary, depending on the number you include, but many will probably be around 2,000 words. Again the length depends on the content.

References

The list should include ALL the sources that you refer to (print and electronic), giving author, title and publication details. Formatting should be consistent.

Appendices

These give supporting information for your project. They provide extra, non-essential information.

2: INTRODUCTION

'a description of the objectives of the project, and what makes them worthwhile; an overview of the achievements; a road map of the rest of the report.'

The introduction is often one of the hardest parts to write. It may be easier to write it last, as you will then have a clear overview of your whole report. It gives the reader some general background to explain what problem the project is designed to solve and why it is worth doing. It shows how the problem is solved and should give an outline of the structure of the report so that the reader can follow it more easily. There are often three stages (moves) starting with more general information, then establishing the problem and finally narrowing down to specific statements about your own project. These moves may occur more than once.

Introduction

Move 1 Background

Possible Steps
1. Why the area is important
2. Giving background information
3. Reviewing previous research

Move 2 Indicating a Problem or Need

Move 3 Presenting the Project

Possible Steps
1. Purposes, aims, objectives
2. Work carried out
3. Justification or importance of the project
4. Outline of the structure of the report

TASK 1: Investigating an Introduction

- Extract A is part of an introduction to a project report.
- Identify the 3 moves and the steps that make them up. The moves occur more than once.
- Underline the language used to signal these moves and steps.

Extract A Introduction: Automatic Marking of Exam Papers Using Semantic Parsing

'Automated Essay Scoring' has been a large area of research since the 1960s. In such a process, a variety of 'features' are extracted from essays, such as word and sentence length and the structure of sentences, before the data is collaborated to provide a final classification [SHERMIS 03].

'Automated Exam Scoring' is a more objective mode of classification, in which answers are analysed for the presence of concrete facts or statements instead of using any continuous measure. This has been the subject of much research in the Computational Linguistics department at Oxford University, using an online study as the source of data, in which students completed a GCSE Biology paper [PULMAN 05]. The techniques employed are varied, but fall into two main categories. One simulating a human style marker defines the marking scheme via patterns inputted by an administrator, allowing for as many variants of an answer as possible. The clear disadvantage of this method is the hours of work required to painstakingly write these patterns, but this method yields high accuracy. Average accuracy in excess of 95% was obtained.

The latter method adopts a machine learning approach using a set of pre-marked answers for the training process. [PULMAN 06] experimented with a system in which the answers are treated as a 'bag of words' with no semantic structure incorporated. A technique known as 'k nearest neighbour (KNN)' was used…

This naive method is subject to a number of problems, as highlighted by Professor Stephen Pulman…

> *"You can't just look for keywords, because the student might have the right keywords in the wrong configuration, or they might use keywords equivalents".*

Thus if the answer requirement is a statement such as 'the cat chased the mouse', then an answer of 'the mouse chased the cat' would be accepted despite the clear semantic inequality, due to the identical set of words.

This project aims to extend this method by incorporating the semantic structure of sentences, so that for the above example 'the mouse chased the cat' would be marked as incorrect, whereas 'the mouse was chased by the cat' would be marked as correct.

CAndC Parser & Boxer

The CAndC (Clark and Curran) parser uses statistical methods and 'supertagging' to convert English sentences into a tree representing their structure, as detailed in [CLARK 07]…The output of the parser is a CCG (Combinatory Categorial Grammar) file, representing this sentence structure.

Alone, this representation is insufficient for machine learning use, given that the semantic interpretation of the sentences is our concern. We therefore use a tool called Boxer, which uses Prolog to convert the CCG into a form called DRS (Discourse Representation Structure). This is compatible with first-order logic, and thus can be used to make reasoned logical deductions (with its application extending to other systems such as Question Answering).

3: BACKGROUND AND REQUIREMENTS

Background should introduce any information that is necessary for the examiner to understand your project. The information here is generally more specific than the background given in the Introduction.
Requirements gives the program requirements. These chapters prepare the ground for the Design chapter.

Extract B shows how these chapters provide an important **link to the Design chapter**.

Extract B Background - 3D Modelling in Java

The 3D modelling system required for this project must be cleanly accessible from within our Java code, allow for dynamic changes to the 3D world, and provide a high-level intuitive interface for doing so. What we require is a system which can interface cleanly with the Eclipse window, and allow user-interaction with the underlying 3D objects.

One such three-dimensional modelling language satisfying these requirements is Java3D. The reason for this is that it provides a way to create a three-dimensional scene, completely in Java, and in a high-level manner…

Requirements

In designing any program, one must consider the requirements, in terms of fulfilling and achieving certain goals, whilst also adhering to the requirements in efficiency and usability enforced by an end-user. I will now discuss what these requirements are:

Accuracy… Efficiency… Usability… Extensibility… Integration…

This list prescribes themes which should feature throughout the design process, whilst giving an overview of what we plan on achieving. We will now continue to describe various aspects of the design which aims to meet these requirements.

4: DESIGN AND TESTING
These are the main body chapters of your report and will therefore vary substantially according to the exact type of work you are carrying out. They are likely to be the longest chapters, probably over 2,000 words.

Design 'maybe including: a description of how you broke the problem down into classes, perhaps supported by class diagrams and/or sequence diagrams; a description of any interesting algorithms or data structures that you used; a description of the user interface, perhaps supported by screen shots.
You should make it clear why the design of your program can be expected to solve the problem. You might like to discuss alternative designs that you considered, and why you decided that they were less appropriate than the one you chose.'
Testing: 'describing what strategy you used to test the program, and how the results compared with those that were expected.'

In Extract C, notice how the writer **first describes** how the plug-in is created and later moves on to **offer an alternative approach** to an aspect of the design.

Extract C Design
4.1 Preliminaries
This section will discuss the methods used in setting up a framework to allow for the dynamic placement of 3D visual objects.
4.1.1 Creating the Eclipse Plug-in
Creating an Eclipse plug-in is a straightforward process. Dave Springgay gives a good outline of the processes necessary [13]. However, essentially we are concerned with creating an Eclipse 'View'…

4.5.5 A Different Approach to Determining Object Size
As we have seen in the Divide and Resize algorithm, the visual objects size can play a vital role in the usability of the general layout. The halving method employed in the divide and resize algorithm seems rather naive, even if it works well visually. Given that the model has access to an importance score for each object, it would seem nonsensical for two objects to be of the same size, when one is vastly more important than the other. Hence, I suggest a sizing algorithm based solely on the importance of the object…

In Extract D, the writer not only **describes** the tests used, but also **evaluates** how useful they are.

Extract D Testing
We have already seen some screen shots of the working program; however, we provide two stringent tests for our program to ensure it works as intended, along with a test rig to fully analyse the program. In both test programs, I will run through the whole series of options available to the user, and ensure its correctness. However, I will also demonstrate its ability to visualise code, and hopefully provide valuable insights whilst debugging.
5.1 Simple Program -BFS and DFS using the Visitor Pattern
This test program begins by creating an underlying tree structure…
This kind of debugging is intuitive, and simple to do within this framework. If you have an intuitive understanding of what the underlying model in your program should look like, it is fairly straight forward to spot bugs like this in small code samples. Assuming a larger program is in use, the user must delve a little deeper into the part of the graph which they suspect the bug to exist in. This is obviously heavily aided by the JDT debugger itself. However, this test still shows the usability of the code in a small program, and shows that the code can cope with the different types of back links and cross links that can occur in a memory graph.

5: CONCLUSIONS
'a summary of your achievements; a critical appraisal, describing what worked well and what could be improved; a discussion of the lessons you have learnt from the project.'

In the 'Conclusions' you summarise your work and then stand back from the project and assess it as objectively as you can. This shows the examiners that you are capable of evaluating your own work according to the standards of the field. It is acceptable to mention areas that were not very satisfactory because this shows that you have learnt from the experience of doing the project.
There are often three moves starting with a summary of your work, then giving an evaluation, highlighting both its achievements and limitations and finally the possible future extensions of the work. The Evaluation move may also be carried out as part of the Summary. The Conclusion is the mirror image of the Introduction, in that it starts with the narrow concerns of the project and widens out to more general statements about further work.

Move 1: Summary
Summarises the most important aspects of the project work.
Move 2: Evaluation
Indicates what the writer considers to be the most valuable aspects of the project and its limitations. Provides an assessment of how far the aims of the project have been achieved.
Move 3: Future Work
Gives possible extensions of the project. These suggestions may follow from the limitations mentioned in the Evaluation move.

TASK 2: Investigating a Conclusion
- Extract E is part of a conclusion to a project report.
- Identify the 3 moves and underline the language used to signal them.

Extract E Conclusions: Efficient Local Type Inference
I have successfully developed a novel local type inference algorithm from an initial specification of the problem. I first provide an intuitive derivation of the algorithm and then offer a formal proof of correctness. I go on to offer generalizations to the algorithm, supporting more language features, and finally achieve local type inference in Jimple.
I have carried out careful experimental evaluation to compare the performance of my algorithm to that of Gagnon et al. [2], which is the only implemented alternative for local type inference in Jimple. Experiments showed a typical 4-fold to 5-fold execution time improvement across a wide range of benchmarks, and a much greater increase where very large methods exist. Experiments were not confined to code compiled from Java, and include code compiled from very different languages like Scala and Scheme. My algorithm is proven to always give a tightest possible typing. Experiments show that Gagnon's algorithm rarely but sometimes gives suboptimal typings.
The theoretical worst case complexity of my algorithm is exponential: whereas Gagnon's algorithm is polynomial. But experiments of execution time against method length show a typically linear trend whereas Gagnon's show a cubic trend. My algorithm is very much optimized for type hierarchies in which most pairs of types have a single least-common-ancestor, which probably includes most Java bytecode in existence today. Of course if a language appeared that made much greater use of multiple inheritance then my algorithm may not be appropriate. But as a practical 'workhorse' implementation I believe this is seriously worthy of consideration. And this is supported by the decision of the Soot framework's maintainers to replace their existing type inference algorithm with mine.
7.1 Future Work
The greatest scope for future work is extending the application of my algorithm from local type inference to global type inference. This involves inferring types for method signatures and public fields as well as local variables. The global type inference problem is currently an area of active research, most of which builds upon the work of Palsberg and Schwartzbach [6]. One could begin by treating method parameters, return values and fields in the same way as local variables, and then using my algorithm on the program as a whole.
7.2 Personal Report…

6: ABSTRACT
'a brief description of what you did; no more than a page.'
The abstract provides a summary of the project report. It may contain some or all of the following moves, often, but not always, in the following order. Only use one or two sentences for each move. Keep the language and sentence structure simple. The reader should be able to read the abstract and obtain the necessary information quickly and efficiently.

Move 1: Background to the project
Move 2: Purpose of the project
Move 3: Problem tackled
Move 4: Work carried out
Move 5: Results
Move 6: Conclusions or implications
Move 7: Achievements of the project

TASK 3: Investigating an Abstract
- Extract F is the abstract of a project report.
- Identify the moves used and underline the language which signals the moves.

Extract F Abstract: Visualising Memory Graphs: Interactive Debugging using Java3D

This report describes a new way of visualising Java run-time objects, and their associated memory graphs. Using the Eclipse debugging framework, alongside the Java3D platform, it aims to describe methods for extracting useful debugging information from a running program and displaying this information in a three-dimensional space. The focus of this report deals with how using a three-dimensional space can enhance the debugging experience, introduce interesting visualisations of programs, and create a basis for future debugging in this way. The result is a user-friendly, efficient system which can visualise large programs in a relatively small amount of screen real-estate. This report shows that three-dimensional visualisation can be a useful tool for debugging, program analysis, and a viable alternative to traditional solutions. (121 words)

7: AVOIDING PLAGIARISM

Plagiarism is defined as the use of other people's work and the submission of it as though it were one's own work. It may be intentional or unintentional.

In academic writing, it must be clear to the reader at any given point whose 'voice' is speaking. This is called **transparent source use**.

- **The responsibility of transparency in writing** means that the writer must use appropriate signals so that an experienced academic reader can understand the actual relationship between the source text and the new one.
- **The responsibility of transparency in language** means that language which is not signalled as a quotation is understood to be original to the writer. If the content is marked with a citation, the language is understood to be paraphrased, i.e., **substantially and independently reworded**.

The process of signalling the presence of another writer's voice in your text is accomplished through **citation**. Citation includes both **quotation** and **paraphrase with reference**.

A quotation has

1. quotation marks: single or double; they can begin and end where appropriate as long as the meaning does not change.
2. a reference to the original source with a page number.

and if necessary

3. ellipsis (3 dots): … to show that something has been omitted.
4. brackets: [added material] to show that something has been added.

A paraphrase is a **substantial rewording** of an idea from another text. The issue is not really how much has to be changed, but rather that the paraphrase should be **composed autonomously**.

Depending on the type of text cited (i.e., whether it is a book, a conference paper, a chapter in an edited volume, a journal article, a website etc.) the elements of a **reference** include:
author; date; title of the book or the article; title of the journal or other work; place of publication; name of the conference; date of publication; page numbers; website address and date of access.

Why do we cite?

1. Knowledge is constructed cumulatively: one person's work builds upon that of another (even if it conflicts with the previous work). What we know now about any field is the cumulative result of the work of many individuals over time. Thus we cite in order to give credit to the person(s) who came up with the fact, idea, observation, etc. that we are citing. The author of the earlier work deserves recognition.
2. It is useful to have additional support for the points that we make. We cite in order to strengthen our arguments and to make it more likely that our points will be accepted.
3. Sometimes a text needs to communicate that there is a difference of opinion or disagreement about a given issue. If so, that disagreement can only be reported by bringing the voices of the disagreeing parties into the text. We cite in order to discuss current issues in the field.
4. Any report of research needs to show that it is new, original, and important. One way to do this is to show what other researchers have done, and that your work fits into a gap left by previous research. In order to do this you will need to name the writers who define the borders of the gap. We cite in order to position our work within the field.
5. The reader may be interested in an issue and may want to know more. We cite so that the reader can locate and read the original source.

Unsuccessful Source Use
Source use is unsuccessful when **the relationship between the source and the new text is not transparent.**

The Consequences of Inappropriate Source Use
- The benefits of citation for the writer disappear, e.g. gaining support for your ideas; being able to position yourself in the field.
- The benefits of citation for the reader disappear, e.g. being able to go back to the sources and read more widely about the point.
- The benefits of citation for the cited writer disappear, e.g. gaining credit for original work.
- The result can look like plagiarism.

What You Can Do
1. Think about the ways in which your text owes a debt to other, earlier texts and acknowledge those debts.
2. Select the right signals to achieve transparency of source use.
3. Be meticulous about taking notes.
4. Know *why* you want to cite each source.

8: EDITING AND REVISING

Use the following points and questions to help you revise and edit your writing.

Communication
1. Think about what your reader already knows and what you need to explain. Are your explanations clear and understandable for someone who is not familiar with your work?
2. Are there any parts of the report where you should have included more information or explanation?
3. Are there any parts of the report which are redundant or which should be cut down?
4. The reader may need help in order to follow the flow of your ideas. Have you previewed the contents of each chapter? Have you ended each chapter with a brief review of what you have said?

Ideas and Organisation
1. Have you introduced your subject appropriately? Have you developed it in a well-organised and logical way? Have you come to a conclusion?
2. Look at the connections in your text. Are the relationships between ideas clear and logical? Have you explicitly signalled the connections to the reader through the linking words you have chosen?

Language
1. Are there any parts of the text where you think grammar problems could interfere with understanding?
2. Are your sentences all full sentences with a subject and a verb?
3. Check that subjects and verbs agree, both singular or both plural.
4. Check your use of vocabulary. Look for unnecessary repetition, misused words, inappropriate colloquialisms and informal language.
5. Check your punctuation. Look for incomplete sentences and missing or wrongly used commas.
6. Check your paragraphing. Have you used paragraphs to break up the text according to the sequence of your ideas? Make sure paragraph breaks are at appropriate points.
7. Check for typographical errors and use a spellchecker.

9: HINTS ON WRITING

1. Plan how you will organise your report. Divide each chapter into sub-sections. This will break up the writing task into parts that are short and manageable.
2. Start with the part you find easiest, perhaps a description of what you have done. Then move on to the more difficult parts as you become more familiar with your topic and experienced in writing.
3. Get your ideas down on paper first, before you revise and edit to improve the language.
4. Don't expect to get the whole report perfect at the first attempt. Ideas change as your work progresses and multiple revisions are the norm in academic writing.
5. If you have a problem or a block, don't keep attempting to solve it. Rather, leave it and return to that point at a later stage or try to approach it from a different angle.
6. Print a draft version of each chapter when it is as good as you can make it. Then leave it for a time (overnight is ideal), before reading it to check for problems or parts that are unclear. Looking at your writing in print enables you to notice points that you may overlook when reading on screen.
7. When revising, the aim is to be critical, to read your report as though it was the work of someone else. Of course, it is also useful to exchange reports with another student and get their comments.