Computer Science Seminar

Writing a 3rd or 4th Year Project Report

Dr Maggie Charles, Language Centre

Outline of the Seminar

1. Parts of the Report
2. Introduction
3. Background and Requirements
4. Design and Testing
5. Conclusion
6. Abstract
7. Editing and Revising
8. Hints on Writing
9. Avoiding Plagiarism

Parts of a Project Report

- Title page
- Abstract
- Acknowledgements (optional)
- Table of contents
- Chapters 1, 2 etc.
- References
- Appendices

Introduction

- Objectives and what makes them worthwhile
- Achievements
- A road map of the rest of the report

Three stages or moves

1. General information
2. Establishing the problem
3. Specific statements about your own project

Introduction (1)

MOVE 1 BACKGROUND

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

MOVE 2 INDICATING A PROBLEM OR NEED

Introduction (2)

MOVE 3 PRESENTING THE PROJECT

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

Recycling

- **Moves and Steps may recycle**

  Moves

  Move 1 ⇒ Move 2 ⇒ Move 1 ⇒ Move 2 ⇒ Move 3

  Steps Within Move 3

  Step 1 ⇒ Step 2 ⇒ Step 1 ⇒ Step 2 ⇒ Step 3 ⇒ Step 4

- **Steps may be omitted**

TASK 1: Extract A (1)

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.

TASK 1: Extract A (1)

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.

TASK 1: Extract A (2)

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.

TASK 1: Extract A (2)

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.

TASK 1: Extract A (3)

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).

TASK 1: Extract A (3)

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

- Information necessary for the examiner to understand your project
- More specific than background given in the Introduction

Requirements

- Gives the program requirements
- These chapters prepare the ground for 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…

Extract B Requirements

3D Modelling in Java

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

Design

- How you broke the problem down into classes
- Interesting algorithms or data structures used
- Description of the user interface
- Why the design of your program should solve the problem
- Alternative designs considered, and why they were less appropriate

Testing

- Strategy used to test the program
- How the results compared with those expected

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…

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…**

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.

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

- Summary of your achievements
- Critical appraisal
- What you have learnt from the project

- Three stages or moves
- Moving from specific to more general statements

Moves in the Conclusion

Move 1: Summary

Move 2: Evaluation

Achievements and limitations

How far the aims of the project have been realised

Move 3: Future Work

Extensions of the project

May deal with the limitations noted in the project

Task 2 Extract E Conclusions (1)

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.

Task 2 Extract E Conclusions (1)

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.

Task 2 Extract E Conclusions (2)

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.

Task 2 Extract E Conclusions (2)

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.

Task 2 Extract E Conclusions (3)

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…

Task 2 Extract E Conclusions (3)

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

- Use 1 or 2 sentences to summarise each chapter of your report
- About 200 words is probably enough
- Keep language and sentence structure simple
- The reader should be able to obtain information quickly and efficiently

Moves in an Abstract

Not all moves will be present in every abstract.

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

Task 3 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)

Task 3 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)

Editing and Revising (1)

Communication

- What does **your reader** already know?
- What do you need to **explain**?
- Have you shown the **examiners** that you understand the material?
- Are your explanations **clear** and **understandable**?
- Is any necessary information **missing**?
- Are there any **redundant** parts of the report?
- Have you **previewed** the contents of each chapter?
- Have you given a brief **summary** of what you have said at the end of each chapter?

Editing and Revising (2)

Ideas and Organisation

- Have you **introduced** your subject appropriately?
- Have you **developed** it in a **well-organised** and logical way?
- Have you come to a **conclusion**?
- Are the **relationships** between ideas **clear** and logical?
- Have you **explicitly signalled** the connections to the reader?

Editing and Revising (3)

Language

- Check for **grammar problems** that could interfere with understanding
- Are your **sentences complete**, with a subject and a verb?
- Check that **subjects and verbs agree**.
- Check **vocabulary** for unnecessary repetition, misused words, colloquialisms and informal language.
- Check **punctuation** for incomplete sentences and missing or wrongly used commas.
- Check that **paragraphing** shows the organisation of your ideas.
- Check for **typographical errors** and use a **spellchecker**.

Hints on Writing

- **Plan** how to organise your report. Divide each chapter into sub-sections.
- **Start** with the **part you find easiest**.
- Get your **ideas down on paper** before you revise and edit.
- Use **multiple revisions**.
- If you have a **problem**, leave it and **return later**.
- **Print a draft version** of each chapter for revision.
- Be **critical**. Read your report as though it was the work of someone else.
- **Exchange reports** with another student and get their comments.

Avoiding Plagiarism

- **Source use** is the use of other people's work in the writer's own work.
- A **Citation** is a reference to the work of another person.
- **Plagiarism** is the use of other people's work and the submission of it as though it were one's own work.

Transparent Source Use

The Responsibility of Transparency in Writing

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

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.**

Two Types of Citation

- **Quotation** is the use of the **exact words** of another person.

  It has **quotation marks** (’...’ or ”...”) and

  a **reference** to the original source with a **page number.**

- **Paraphrase with reference** is a **substantial rewording** of an idea from another text.

  It should be **composed autonomously**.

  It has a **reference** to the original source.

Parts of a Reference

- **Depend on the type of text cited** (e.g. book, conference paper, chapter in an edited volume, journal article, website etc.)

The elements of a **reference** include:

- author; date; title of book or article; title of journal or other work; place of publication; name of conference; date of publication; page numbers; website address and date of access

Why Do We Cite?

- To **give credit** to the person(s) who first put forward the information that we are citing
- To have additional **support** for the points that we make
- To **strengthen our arguments** and make it more likely that our points will be accepted
- To **discuss current issues** in the field
- To **position our work** within the field
- So that **the reader** can **locate** and **read** the original source

The Consequences of Inappropriate Source Use

- The **benefits** of citation for the writer **disappear.**
- The **benefits** of citation for the reader **disappear.**
- The **benefits** of citation for the cited writer **disappear.**
- The **result** can look like **plagiarism.**

What You Can Do

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