Cardiff University
School of Computer Science and Informatics

A Short Guide to Writing Your Final Year Project Report Or MSc Dissertation

February 2011

Abstract

This guide is intended to help you produce a good final year project report or MSc dissertation. It gives advice on how to gather relevant material, how to organise it into a suitable form and how to then turn it into a written project report or dissertation. It also describes the conventions that should govern the structure of the report or dissertation, and suggests some descriptive devices that you can use to make it more effective. A summary of the guidelines is given at the end. The appendix lists the rules governing presentational details, including print quality, font sizes, etc.

Table of Contents

1 Introduction **3**

2 Gathering Material **5**

3 Arranging Material and Structuring the Project Report **6**
3.1 The “Introduction” 8
3.2 The “Background” 9
3.3 The “Specification & Design” 11
3.4 The “Implementation” 12
3.5 The “Results and Evaluation” 12
3.6 The “Future Work” 13
3.7 The “Conclusions” 13
3.8 The “Reflection” 13
3.9 The “References” 14

4 Writing the Project Report **16**
4.1 Potential Readership 16
4.2 Identifying Commonality 17
4.3 Sections and Subsections 17
4.4 Stylistic Conventions 17

5 Using Descriptive Devices **19**
5.1 Cross-references 19
5.2 Footnotes 19
5.3 Lists 19
5.4 Figures 20
5.5 Literal Text 21

6 Supporting Material **22**
6.1 The Title Page 22
6.2 The Abstract 22
6.3 Acknowledgements 23
6.4 The Table of Contents and Table of Figures 23
6.5 The Glossary and Table of Abbreviations 23
6.6 The Appendices 23

7 Sources of Further Guidance **24**

8 Conclusions **25**

Appendix A: Typesetting Rules for Report Presentation **26**

Appendix B: Bibliography **27**

1 Introduction

This guide is meant to help you produce a good final year project report or MSc dissertation. A good report is one that presents your project work *concisely* and effectively. It should contain various materials relevant to the work you have undertaken in respect of your project; it should be organised into a logical framework; and it should be supported by written material that follows well-established academic conventions in a consistent fashion.

The purpose of the project is, in the context of the degree you are studying, to integrate various aspects of the taught material and to demonstrate your (academic) research skills and your (professional) analysis, design and implementation skills. It gives you the opportunity to conduct in-depth work on a substantial problem to show individual creativity and originality, to apply where appropriate knowledge, skills and techniques taught throughout the degree programme to further oral and written communication skills, and to practise investigative, problem-solving, management and other transferable skills. The management and execution of the project is your responsibility, but you should seek and take advantage of advice from your supervisor.

When you choose a project, you should do so carefully, to reflect the focus of the degree programme you are enrolled in, your personal interests (the project needs to keep you interested for the whole the academic year) and the ability of the academic staff to support you throughout your project. Projects vary widely in the problem they address and the products they deliver at the end. While the main product of some projects is a piece of software or hardware, other projects produce a systems model or design, and yet others may address some research hypothesis using a theoretical or experimental approach. This means not every project produces a piece of software. In brief, the better defined the problem that your project addresses, the further through the systems lifecycle you should expect to progress in the course of your project. If instead you are addressing a research hypothesis, your main product may be the evaluation of some experiments or a theoretical result.

So, for example, a project that seeks to develop a logistics planning system for a small business or voluntary organisation would be expected to provide a fully operational, fully tested program that meets all the identified needs of the client. However, a project that aims to validate a government policy in a particular area might only achieve the development of a model to confidently simulate the main factors influencing that policy, and identify the research agenda in terms of specifying precisely the data requirements to allow a full investigation of the relevant factors. A scientifically oriented project may focus on the practical or theoretical evaluation of a new rendering approach and compare it with existing approaches, which may involve some implementation, but does not require fully functional software.

An important point to remember is that the report should describe *your* work. Large chunks of bookwork describing standard material are unnecessary. You should simply refer to such material where necessary – assume that your reader is a competent computer or information systems theorist or practitioner.

The guidelines here are arranged roughly in the order that you will need them. For undergraduate projects much of the information given here is based upon the ideas presented in the first year module on “Professional Skills” (CM0128). Therefore we recommend that you refer back to that module’s notes.

Your project supervisor will guide you on what it is reasonable to expect a project in your chosen topic to deliver. However, all projects are required to justify all decisions made at every stage of research and the development of appropriate deliverables, including the choice of approach.

2 Gathering Material

This section outlines the kinds of material you need to collect before you can begin writing in earnest. Most of the necessary material will consist of your own ideas and experiences gained while carrying out the project, and your approach to solving the problem you have decided to address. For the background study or literature review you will also need references to various resources such as key books and papers, policy documents, Internet resources, related software, etc.

While working on the project you may find it helpful to keep a notebook handy and record all relevant information. Typically such information will include:

- references such as papers, books, websites with full bibliography details;
- lessons learned, for inclusion in the "reflective" part of your report;
- notes from meetings or interviews with
    - your supervisor;
    - potential end-users and other stakeholders;
    - technical experts;
- and so on.

Also, we recommend that you keep a diary of all your project-related activities. This will show the progress made during the life of the project and will provide a record of how you spent your time. In particular, when you are validating, testing and debugging your work, keep a running log of your activities and their outcomes. You will then have a record of the unforeseen difficulties you met and, hopefully, how you resolved them. Summaries of these may well be worth including in the project report (see Section 3.4).

In general you should supplement the material you generate yourself with relevant material from other sources. A good project report will show that you are aware of relevant work that other people have done (see Section 3.2). You should include relevant references to such work in your project report. References to work in periodicals, i.e. magazines and journals, and conference proceedings *may* be more useful than references to textbooks, as periodicals and conferences are usually more specialised and up to date. References to technical manuals and national and international standards should also be included, where appropriate. You may also cite web sites as sources, if suitable. However, keep in mind that web sites may often contain incomplete or wrong information and in general textbooks or papers are a better reference and show that you have done a more extensive literature review than just searching for some keywords on the Internet.

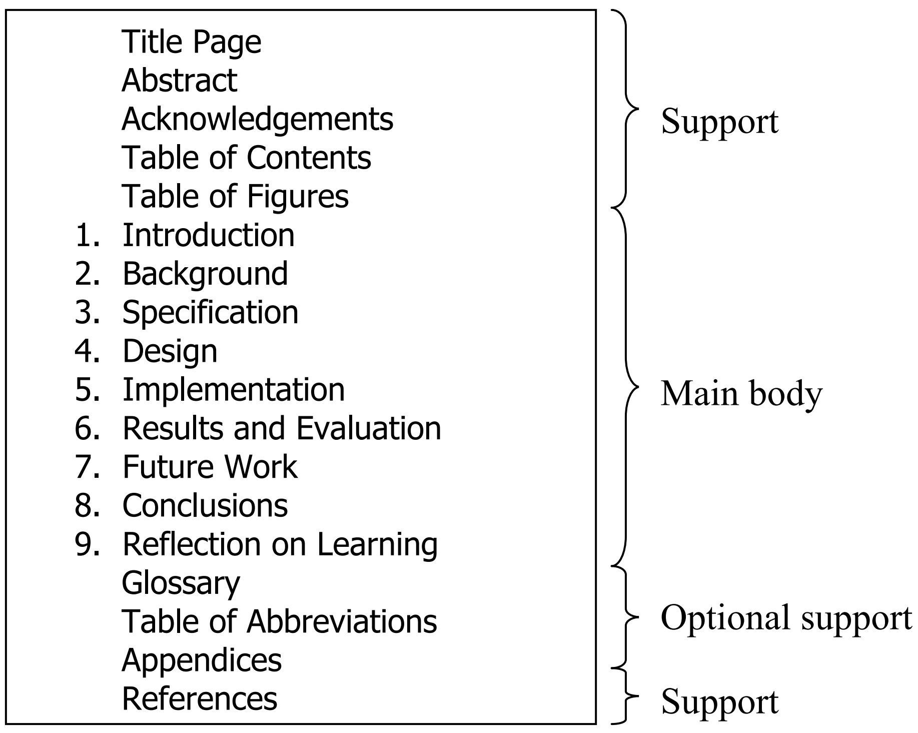
3 Arranging Material and Structuring the Project Report

You should consider, at the beginning of your project, what you need to do to solve the problem you have chosen to address. This will then inform choices about the structure of your report; your written report needs to be both a "narrative" (telling the story of your project) and an "argument" (providing a logical justification of the steps you have undertaken to solve your chosen problem). Once you have started to gather material you can begin to arrange it in a form which can then be refined into the final project report, though the outline chapter headings shown below will serve as a good guide in the early stages of your work.

All good project reports whatever their subject, follow certain well-established conventions and have a similar overall shape. They generally consist of a main body surrounded by other information (presented in appropriate formats) that support it in various ways. Some of these are mandatory, others are optional.

Figure 3.1 shows an example of the layout we suggest for a project which implements a piece of software. You should vary the titles of the sections if these are inappropriate for your project – your supervisor is the best person to guide you on this. For the moment we will concentrate on the main body of the report and leave the supporting information until later. We recommend that you do the same when writing your report, though you should have a plan for your final report which will guide you on what material your should be retaining for eventual inclusion.

Project reports describing projects whose aim has been to develop a particular software system tend to have a main body with a characteristic structure as illustrated above. For those which address a "softer" problem, these principles remain, though a more usual structure is shown in Figure 3.2.



Title Page
Abstract
Acknowledgements
Table of Contents
Table of Figures
1. Introduction
2. Background
3. Specification
4. Design
5. Implementation
6. Results and Evaluation
7. Future Work
8. Conclusions
9. Reflection on Learning

Glossary
Table of Abbreviations
Appendices
References

Figure 3.1: Suggested report structure for a project which implements a piece of software.

Title Page
Abstract
Acknowledgements
Table of Contents
Table of Figures
1. Introduction
2. Background
3. Selection of Approach
4. Application of Selected Approach
5. "Deliverables" from Selected Approach
6. Results and Evaluation
7. Future Work
8. Conclusions
9. Reflection on Learning

Glossary
Table of Abbreviations
Appendices
References

Figure 3.2: Suggested report structure for a project addressing a "softer" problem.

Title Page
Abstract
Acknowledgements
Table of Contents
Table of Figures
1. Introduction
2. Background
3. Description of Algorithms
4. Implementation
5. Experiment Design
6. Algorithm Comparison Results
7. Future Work
8. Conclusions
9. Reflection on Learning

Glossary
Table of Abbreviations
Appendices
References

Figure 3.3: Suggested report structure for comparing algorithms.

Title Page
Abstract
Acknowledgements
Table of Contents
Table of Figures
1. Introduction
2. Background
3. Problem Statement
4. Alternative Designs and Final Algorithm
5. Implementation
6. Experimental and Theoretical Results
7. Future Work
8. Conclusions
9. Reflection on Learning

Glossary
Table of Abbreviations
Appendices
References

Figure 3.4: Suggested report structure for the design and analysis of an algorithm.

If the nature of the project is not to design and implement some software, but is more of an investigative or research nature, for example to compare two algorithms, a more suitable layout could be the one shown in Figure 3.3. Another project might involve the design and analysis of an algorithm. Here, there might be a lot of analysis of the problem and its solution and little to say on the systems aspect or user interaction, for example. A possible report layout for the project is shown in Figure 3.4.

We look at each of the general sections of the report strucutre in more detail below. You can use this characteristic structure as a rough template for organising the material. However, often it may be of advantage to adjust the suggested structure to your particular project instead of sticking to the template. Consult your supervisor for advice. It is also a good idea at this stage to plan roughly how long each part should be, to make sure that the length and overall balance are about right. You can then construct each part to produce a first draft of the main body.

3.1 The "Introduction"

A good introduction should tell the reader what the project is about without assuming special knowledge and without introducing any specific material that might obscure the overview. It should anticipate and combine main points described in more detail in the rest of the project report. Also, importantly, it should enthuse the reader about the project, to encourage them to read the whole report. Normally it should include such things as:

- the aim(s) or goal(s) of the project;
- the intended audience or "beneficiaries" of the work done;
- the scope of the project;
- the approach used in carrying out the project;

- assumptions on which the work is based; and
- a broad summary of important outcomes.

3.2 The "Background"

The purpose of the Background section is to provide the typical reader with information that they cannot be expected to know, but which they will need to know in order to fully understand and appreciate the rest of the report (see Section 4.1 for details of who a typical reader might be). It should explain why the project is addressing the problem described in the report, indicate an awareness of other work relevant to this problem and show clearly that the problem has not been solved by anyone else. This section may describe such things as:

- the wider context of the project;
- the problem that has been identified;
- likely stakeholders within the problem area;
- any theory associated with the problem area;
- any constraints on the approach to be adopted;
- existing solutions relevant to the problem area, and why these are unsuitable or insufficient in this particular case;
- methods and tools that your solution may be based on or use to solve the problem;
- and so on.

The wider context of the project includes such things as its non-computing aspects. So, for example, if you are producing software or any other products, including business recommendations, for a specific organisation then you should describe aspects of that organisation's business that are relevant to the project.

Relevant existing products, documents or artefacts that you should mention could be ones that, for example,

- are similar to the one you are proposing;
- support your project;
- your project aims to extend or replace;
- demonstrate the "deficiencies" your project intends to address.

You need only describe things that will be unfamiliar to the potential reader, or are unique to the organisation or topic your project addresses. Your project, if it involves software development, will almost certainly use all kinds of existing software such as language compilers, subroutine libraries, etc., but you can assume that the reader will be fully acquainted with, for example, general purpose programming languages such as Java, C/C++, Fortran, Pascal, Python, PHP, etc,. Also, it may involve the better known specialised packages such as MySQL, ORACLE, OpenGL, etc. You should mention the particular variety and possibly version number, e.g. Java SE 6, but you need say nothing more than that.

If your project depends on any specialist or uncommon software such as specialised subroutine packages or a more obscure or specialised programming language, you should describe them briefly and discuss whatever features are relevant to your project. Often this can be done by comparing it to some well-established piece of software, for example

```
The Descartes language is like a restricted version
of Pascal but with the following extra features: ...
```

Again, long descriptions of details are to be avoided and references to suitable sources of detailed information should be given instead.

Other background information could consist of the sequence of events leading up to the present situation or the results of earlier investigations. You could also discuss such things as any cost or time constraints imposed on the project.

Your background section should end with a clear statement of the research questions problem your project is trying to answer. These will reflect the aim of your project, but will be different in that they explain the problem you are attempting to solve, e.g.,

Example 1:

Aim:
The aim of this project is to develop software for the improved planning of the routing of delivery vehicles to customer locations, that reflects the forecast availability of each customer to receive goods.

Research question(s):
In order to demonstrate the achievement of the stated aim, this project will identify route planning software currently in use and the underpinning algorithms, define appropriate performance metrics, determine how to express constraints on an alternative algorithm, develop an improved algorithm and demonstrate on what basis it is judged an improvement, and implement the improved algorithm in a usable and robust software package.

Example 2:

Aim:
The aim of this project is to develop a business strategy for organisation X that will improve the survivability of X in the face of increasing global competition.

Research question(s):
In order to develop a business strategy it will be necessary to identify key stakeholders and determine their vision for the organisation at the end of the strategic planning timeframe, assess the likely outcome, in terms of the organisation’s survivability, of maintaining the current strategy, and develop and assess an alternative set of activities to achieve the stated vision.

3.3 The “Specification & Design”

The purpose of the Specification and Design sections is to give the reader a clear picture of the system you plan to create, in terms of the capability required. A specification should tell the reader what the software system is *required* to do. The design then gives the top-level details of how the software system meets the requirement. It will also identify constraints on the software solution, that are important in guiding ecision making throughout the development process.

Describing what a software system does (specification) and how it does so (design) effectively usually means describing it from more than one viewpoint. Each viewpoint will convey some information about the system that other viewpoints omit. (You would use the same technique when describing any complicated construction such as a building, an aircraft, a novel or a painting). Possible viewpoints might be:

- the business model the software supports;
- the user interface;
- the dynamic behaviour of the system;
- how data flows through the system;
- what data types are implemented in the system;
- what algorithms are implemented in the system;
- the static architecture of the system, i.e. how the code is partitioned into modules, etc.

A common approach is to first define the user or business requirements, then describe the static architecture, identify modules and groups of closely connected modules, and then to apply other views to each of these groups. Fine details, specifically details of code, should be left out.

We strongly recommend that you make extensive use of diagrams, such as entity-relationship diagrams, UML diagrams, state charts, or other pictorial techniques (see Section 5.4 for more detailed advice on this).

As well as describing the system, it is important that you *justify* its design, for example, by discussing the implications of constraints on your solution and different design choices, and then giving reasons for making the choices you did. Typically these implications will relate to the aims of the project and to aspects of it discussed in the Background section.

The design of the system will almost certainly have evolved while you were developing it. Obviously you should describe its final state but often there are good reasons for describing intermediate states, too; for example, if you want to discuss the details of the design method used or to highlight learning that you later refer to in the Reflection section. If you do this, take special care to make sure the reader does not get confused between different stages of the design.

If you are not designing a system, but testing a hypothesis for a more scientifically oriented project, specification and design sections may not be required in quite the same form (see Figure 3.3 and Figure 3.4). The specification instead becomes a description of the problem and what is required of a solution. The design becomes a description of your approach to solving the problem and your suggested soltuion(s). For instance, if you are designing an algorithm to solve a particular problem you would have a problem statement section and then a section describing one or more suggested algorithms to solve the problem. Later in the

Results and Evaluation section you then describe how to design experiments to test how well the algortihm(s) solve the problem and present your experimental results with an evaluation of your suggested solutions.

3.4 The "Implementation"

The Implementation section is similar to the Specification and Design section in that it describes the system, but it does so at a finer level of detail, down to the code level. This section is about the realisation of the concepts and ideas developed earlier. It can also describe any problems that may have arisen during implementation and how you dealt with them.

Do *not* attempt to describe all the code in the system, and do *not* include large pieces of code in this section. Complete source code should be provided separately (see Appendix B and submission guidelines). Instead pick out and describe just the pieces of code which, for example:

- are especially critical to the operation of the system;
- you feel might be of particular interest to the reader for some reason;
- illustrate a non-standard or innovative way of implementing an algorithm, data structure, etc..

You should also mention any unforeseen problems you encountered when implementing the system and how and to what extent you overcame them. Common problems are:

- difficulties involving existing software, because of, e.g.,
  - its complexity,
  - lack of documentation;
- lack of suitable supporting software;
- over-ambitious project aims.

A seemingly disproportionate amount of project time can be taken up in dealing with such problems. The Implementation section gives you the opportunity to show where that time has gone.

3.5 The "Results and Evaluation"

In this section you should describe to what extent you achieved your goals.

You should describe how you demonstrated that the system works as intended (or not, as the case may be). Include comprehensible summaries of the results of all critical tests that were carried out. You might not have had the time to carry out any full rigorous tests – you may not even got as far as producing a testable system. However, you should try to indicate how confident you are about whatever you have produced, and also suggest what tests would be required to gain further confidence.

This is also the place to describe the reasoning behind the tests to evaluate your results, what tests to execute, what the results show and why to execute these tests. It may also contain a discussion of how you are designing your experiments to verify the hypothesis of a more scientifically oreinted project. E.g., describe how you compare the performance of your algorithm to other algorithms to indicate better performance and why this is a sound approach. Then summarise the results of the tests or experiments.

You must also critically evaluate your results in the light of these tests, describing its strengths and weaknesses. Ideas for improving it can be carried over into the Future Work section. Remember: no project is perfect, and even a project that has failed to deliver what was intended can achieve a good pass mark, if it is clear that you have learned from the mistakes and difficulties.

This section also gives you an opportunity to present a critical appraisal of the project as a whole. This could include, for example, whether the methodology you have chosen and the programming language used were appropriate.

3.6 The "Future Work"

It is quite likely that by the end of your project you will not have achieved all that you planned at the start; and in any case, your ideas will have grown during the course of the project beyond what you could hope to do within the available time. The Future Work section is for expressing your unrealised ideas. It is a way of recording that 'I have thought about this', and it is also a way of stating what you would like to have done if only you had not run out of time[1]. A good Future Work section should provide a starting point for someone else to continue the work which you have begun.

3.7 The "Conclusions"

The Conclusions section should be a summary of the aims of project and a restatement of its main results, i.e. what has been learnt and what it has achieved. An effective set of conclusions should not introduce new material. Instead it should briefly draw out, summarise, combine and reiterate the main points that have been made in the body of the project report and present opinions based on them.

The Conclusions section marks the end of the project report proper. Be honest and objective in your conclusions.

3.8 The "Reflection"

We believe in the concept of "lifelong learning". One of the principles applied throughout the assessment during your studies is that of the value of reflection. We believe that it is important that we reflect upon our performance in order to identify "transferable learning", that can be carried over into future activities. Reflection should focus on what Argyris calls "double loop learning"; this is where we identify, not relatively "simple skills", such as the mastery of a new programming language, but the impact of what we have done on the assumptions, concepts and ideas we used to make decisions about our work. For example, a "reflective practitioner" would try to identify the characteristics of the problem that has been addressed, and consider whether assumptions or decisions about the relevant approach to solving that problem had been appropriate, in order to make a better decision in relation to problems that might be encountered in the future.

[1] Remember to take into account Hofstadter's Law:
'Everything takes longer than you think, even when you take into account Hofstadter's Law.'

3.9 The "References"

In Section 2 we said that you should relate your work to that of other people. Other work explicitly cited should be listed in the Reference section and referred to in the text using some kind of key. It is important that you give proper credit to all work that is not strictly your own, and that you do not violate copyright restrictions.

It *may* be desirable to provide a Bibliography section separately from the reference section. In general, references are those documents/sources cited within the text. The bibliography lists documents which have informed the text or are otherwise relevant but have not been explicitly cited.

References should be listed in alphabetical order of author's surname(s), and should give sufficient and accurate publication details. For example,

> Chikofsky, EJ, Cross, JH. 1990. Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, 7(1):13-17.
>
> Date, CJ. 2000. An Introduction to Database Systems, 7th Edition. Addison-Wesley.

are acceptable references.

There are various conventions for quoting references. For example, you can quote the name of the author and the year of publication, e.g.

```
For more information see [Chikofsky et al, 1990]. A
more detailed description is given by Date [2000].
```

There are several other variations. For example, some authors prefer to use only the first three or four letters of the name, e.g. [Chi1990] or just to number the references sequentially, e.g. [3]. It can be helpful to the reader if, for books and other long publications, you specify the page number too, e.g. [Date 2000, p. 23].

Whatever convention you choose, **be consistent**.

Information Services provide a number of leaflets which describe in detail accepted ways of presenting references. For example, guidance on the Harvard Style of citing and referencing may be viewed at
`http://www.cardiff.ac.uk/insrv/resources/guides/inf057.pdf`.

Whatever style of referencing you adopt, it is critical that you are assiduous in acknowledging the sources you have used; failure to do so may lead to suspicions of unfair practice and an investigation into whether or not your work reflects the standards expected of academic research. Guidance on plagiarism and how to avoid it is available at
`http://learningcentral.cf.ac.uk/bbcswebdav/institution/INSRV/Study%20Skills/plagiarism2/new/index.html`.

Note that it is seldom sufficient to simply "cut and paste" material from other sources. When you take material from someone else's work, you are doing so because it helps support your argument, or justify decisions you are making. It is therefore essential to make it clear why you have included material from other sources; in other words, you need to critically assess the work of others, whether it is supporting your position or not:

- If the material you are citing from another source supports your position, you must explain why it should be trusted. For example, material from a published journal will, normally, have been peer-reviewed and can therefore be considered to have some validity, according to subject matter experts. Much of what is published on the Internet cannot be regarded in the same way, however.
- You will often find that there are conflicting views in the published material; in such cases you must explain which view you favour and why, before relying on the material to support your position.
- If other writers have taken a different position to the one you support, you must explain why the reader should accept your ideas rather than those proposed elsewhere.

In summary, you need to ensure that you have clearly assessed the relevance of referenced material to the development of your position, or your argument, and demonstrated that you are justified in taking this material to be authoritative.

4 Writing the Project Report

Once you have gathered and organised enough material you can turn it into written prose. To write effectively requires sustained concentration over long periods of time. Even with the incremental authoring possibilities that word processing offers, writing is best done in long uninterrupted sessions. Most people find it difficult and tiring.

There are rules you can follow which may make the task easier and which will certainly improve the quality of your writing, but unfortunately there are rather a lot of these and in a guide of this size we can only offer a few pieces of general advice:

- keep your potential readership in mind;
- identify commonality;
- use sections and subsections both to structure your work and to provide appropriate breaks for the reader;
- do not include "padding"; include only what is necessary to "tell the story" and justify your work;
- follow appropriate academic and professional stylistic conventions. We recommend that you read journal papers relevant to the general area of your project, as well as project reports held in the library; this is a normal research activity.

The project report's structure does not necessarily dictate the order in which you write it. If you want you can start by writing the Introduction, then the Background section, and so on, but this is up to you. Some people start by writing the Introduction first which gives direction to writing the other sections, but others prefer to leave writing the Introduction until last, as projects rarely turn out as planned. We recommend that you start with the middle sections, then write the Introduction (guiding the reader to what they will find in the report), then the Conclusions (bringing the report together at the end) and Reflection, and finally the Abstract (summing up the entire report). However you tackle the writing up, we recommend that you:

- write as you go along, rather than leaving all the writing until last (writing takes longer than you think, and is best done when the ideas remain fresh in your mind);
- leave time for someone you trust to proof-read your work, and for you to correct errors (it is not your supervisor's responsibility to correct your written English);
- read your work *out loud* to yourself. There are many advantages to this, not least the realisation that if you run out of breath your sentences are probably too long. Mainly, however, if you read "silently", you will tend to read what you meant to write, rather than what you have in fact written, and will run the risk of missing errors.

4.1 Potential Readership

Always keep your potential readers in mind and repeatedly review what you have written, putting yourself in their place. Look at the draft, sentence by sentence, and ask yourself: 'Will this make sense to the readers given their existing knowledge and what I have told them up to now?' You can consider the potential readership as

- your academic supervisor;
- your project moderator/internal examiner;
- the external examiner (usually a computing professor from another university),
- and quite possibly future students and others interested in the topic.

So, as noted earlier, do not explain things which are common knowledge to such readers.

Also, if your project report is of sufficient quality, your supervisor may consider submitting part of it to a journal for publication as a paper, in which case it may eventually be read by a substantial number of computing and other professionals.

4.2 Identifying Commonality

You can often both clarify text and reduce its bulk if you can identify generality or commonality among the ideas you are expressing. You can then revise the text so that the common factors are described first followed by details of how specific individual ideas differ from them.

4.3 Sections and Subsections

The main body of the project report should be divided up into sections, along the lines suggested in Section 3 or otherwise, as appropriate. Each section should, if necessary, be divided up into subsections, and so on recursively. Such nesting can be used to suggest some kind of hierarchical relationship between sections. This can become obscure though if the nesting gets to more than about three levels deep.

It is important that you start each section and subsection with a summary of the rest of the material in it, i.e. inform the reader of what you are about to tell them. This has the effect of "softening up" the reader so that when they move on to the body of the section they feel confident about the direction in which you are taking them. They are reassured at regular intervals when they encounter ideas that you have told them to expect. Without the overview the overall effect is like a mystery tour of ideas, with each new idea coming as a surprise. It is sometimes difficult to appreciate the need for this when you are the author because you are already intimately familiar with the whole route that the report takes.

Each major section should begin on a new page. All sections and subsections should be numbered and headed. Numbering should be like this: 3.10.7 – for subsubsection 7 in subsection 10, in section 3.

4.4 Stylistic Conventions

There are all kinds of stylistic conventions relating to technical writing that you should try to follow. For example:

- do not use shortened forms such as "don't" for "do not";
- avoid colloquialisms and slang words;
- use British English and write in complete sentences;
- divide your writing up into paragraphs;
- generally, you should write in the "third person". The "first person" can be used, to avoid the report becoming stilted, though it is recommended that its use be limited; for example, it may be appropriate to use "I" when stating an opinion rather than the common "It is the author's opinion…".

Writing where the language style or typography, e.g. font or character size, change arbitrarily looks amateurish and can be very distracting for the reader. Use typography to support the content. Other places where consistency should be maintained include:

- bullet points;

- use of hyphens;
- use of capitalisation;
- technical terms;
- abbreviations;
- use of symbols.

To some extent you can use your own judgement about what conventions to follow. Whatever you do though, you must **be consistent**.

5 Using Descriptive Devices

In this section we will mention some well-established descriptive devices which you can use in your project report to improve its quality.

5.1 Cross-references

Cross-references are just references to other parts of the same document. For example,

```
This module contains procedures for operating on
variables of type WINDOW (see Section 2.2).
```

Section numbers will change if sections are added or deleted. Good typesetting or word processing software provides suitable mechanisms to automatically number sections and create such references such that they will always refer to the intended section. Make sure you know how your chosen software does this and select the right software to make this simple. If you use software that does not support cross-references or uses an overly complicated system, it is a good idea to wait until the report is almost complete before putting in any cross-references.

Backward references to sections earlier in the project report can make explicit connections between parts of the document that may not be connected obviously. Forward references can be used, for example, to reassure the reader that you are not going to leave them stranded after you have introduced a new idea without explaining it. For example,

```
This procedure uses the Volestrangler algorithm (to
be described in Section 4.3).
```

Note that too many forward references are probably an indication that the report could be organised better.

5.2 Footnotes

Many word processors have facilities for handling footnotes. By all means use them, in particular when you want to make a comment which is not strictly relevant or which would upset the flow of ideas in the text. If the comment is closely related to the text you may consider including it in parenthesis instead.

5.3 Lists

Traditionally, collections of items are listed within the text using the adverbs ‘firstly’, ‘secondly’, etc. Often, though, it is clearer to tabulate these items, particularly if there are many of them. The simplest way of doing this is to use a “bullet” list[2]. Various examples of bullet lists appear throughout this guide. Sometimes there is a need to nest one list inside another. To distinguish the two lists, the inner one can be indented and have a different symbol. Lists with more than one degree of nesting tend to appear confusing and therefore we do not generally recommend them.

[2] “Bullet” is the name for the ● symbol, although other similar symbols are also available.

Listed items can also be keyed using numbers, letters, or other labels. Bibliography entries are an example of keyed items (see Section 3.9). However, keys should only be used when necessary.

5.4 Figures

A project report that uses figures (i.e. diagrams or other pictorial techniques such as tables) to illustrate ideas will probably be easier to digest than one that does not. We therefore recommend that you use figures wherever appropriate[3].

Be careful though. When drawing diagrams try to keep to a standard graphical notation that has been introduced during your studies, or that you have seen published widely, and use it consistently. Computer Science, unlike most other professions, has few established conventions governing the use of diagrams and this means that diagrams can sometimes make ideas more obscure rather than clarifying them.

If you feel you have to invent your own notation, remember that the best ones are usually the most *economical*, i.e. they use only a few different kinds of symbols. Also, you **must** explain the precise meaning of your symbols in a key. A very common mistake is to use arrows to illustrate some kind of relationship between items without declaring what that relationship is.

Graphics editors (i.e. picture processors) can be extremely useful, particularly if you have a great deal of drawing to do or if there is a lot if commonality among the drawings (because cut and paste operations can then be used with great effect). However, some artefacts are difficult to produce using standard software applications, and in such cases it is quite acceptable to present hand-drawn diagrams. To include these into your report you may use a scanner or even take a photograph (or multiple patial photographs that you merge afterwrds) of the artefact and include it in your report as any other computer generated image (help from staff for this is available and you may bring the original artefact to the viva).

All figures should be labelled and captioned, for example,

```
Figure 3.10: Sub-System Architecture.
```

The label can then be used to refer to the diagram within the text, e.g.

```
See Figure 3.10.
```

All diagrams must be explicitly referred to somewhere within the text.

Similar to sections and subsections the labels may change if you insert additional figures or change the structure of the report. Again good typesetting software will support automatic label generation and keeping the references to the figures consistent (see Section 5.1).

For some reports it may also be useful to distinguish between figures and tables and use separate labels for them (e.g. Figure 3.1 and Table 3.1 are two separte elements, sometimes

[3] If you have a graphical rather than a textual/verbal kind of mentality, a good way to write text is to express your ideas in diagrams first and then describe these textually.

also referred to as floats). Figures are diagrams, drawings, images, etc. while tables list information in a tabular layout, e.g. program running times for specific inputs.

5.5 Literal Text

It is important when writing about software systems to distinguish in the text between the ordinary natural language you are using and the program code or other literal text. If you are using a word processor which offers both proportionally spaced and fixed width character fonts then there is a straightforward way of doing this. Program code and other literal text can be written in a fixed width font such as "Courier New" while the natural language text can be written with a proportionally spaced font such as "Times New Roman". For example:

> The procedure `draw_circle (p:POINT, r:REAL)` draws a circle of radius `r` at point `p` on the screen.

Other similar kinds of text, UNIX commands for example, can be treated in the same way. Some typesetting systems also offer to include "verbatim" text, which you can use to insert small code examples, examples of the output of a program, etc. They are also typeset in a fixed width font. Using a fixed width font means that the code appears in the document much as it would do on a console. If you only have fixed width characters available on your word processor then put program code etc. into *italics* or **bold** text.

Note that using more than a few different character fonts, styles or sizes can make text look very untidy. Generally we recommend to use, e.g., a serif font for the main text (or a sans-serif font, if you prefer), a fixed-width font for literal texts as above, and optionally one sans-serif font for headings and captions (this can also be the same font used for the main text). Emphasis can be indicated by italics or stronger using bold text. If you use more fonts you should have a very good reason for this to support the content.

6 Supporting Material

In Section 3 we said that a project report consisted of a main body plus other supporting material that surround and support the body. There are well established conventions governing the purpose and format of these supporting structures which we will describe now. The structures include, in order of appearance in the project report:

- the title page;
- the abstract;
- the acknowledgements;
- a table of contents;
- a table of figures.

Then comes the main body of the project report, and this is followed possibly by:

- a glossary;
- a list of abbreviations;
- one or more appendices;

and finally

- the references and bibliography.

Each of the elements listed above should begin on a new page. All pages should be numbered, with page 1 being the first page of the Introduction. The pages preceding the Introduction should be given Roman numerals (i, ii, iii, iv, etc).

6.1 The Title Page

The title page should be the first page of the report and should normally include:

- the title of the project report;
- the name of the author;
- the name of the project supervisor;
- the qualification for which the project report is a part;
- the name of the school and institution, e.g. School of Computer Science and Informatics, Cardiff University;
- the date of completion of the project report.

The title itself should be short, yet should aim to describe the contents of the project report as accurately as possible.

MSc students are advised to consult current university submission regulations to determine the appropriate form of the title page and page of declarations required in an MSc dissertation.

6.2 The Abstract

This is a summary of the report. It must be less than 300 words long. It should give enough information to allow a potential reader to decide whether or not the report will be of interest to them. It should briefly describe the main ideas of the report, including the aims and conclusions. It should be both self-contained and self-explanatory, and it should not say anything not mentioned in the rest of the report (for this reason it is usually written last).

6.3 Acknowledgements

This optional section should be used to record indebtedness for the use of facilities or help from particular sources. You should mention any organisations who have helped you while you have been carrying out the project.

6.4 The Table of Contents and Table of Figures

The table of contents gives the reader a view of the detailed structure of the report, by giving section and subsection headings and associated pages.

If your project report contains many figures or it refers to the same figure many times you should consider listing them along with their page numbers in a table of figures.

6.5 The Glossary and Table of Abbreviations

If you use any abbreviations, obscure terms or esoteric acronyms in the project report then their meaning should be explained where they first occur. If you go on to use any of them extensively then it is helpful to list them all in a table at the end so that readers can quickly remind themselves of their meaning.

6.6 The Appendices

Appendices are where you present material which you want to include in the report, but which would seriously obstruct the flow of ideas if put anywhere in the main body. This could be extensive technical details or mathematical proofs, derivations of formulae, etc. required to support a point your are making in the report. Other documents you have written, such as user manuals, technical manuals or formal specifications should go here too.

The appendices should not contain any of the source code for your software. This will be submitted differently (see Appendix A and separate project submission instructions).

Appendices should be headed by letters in alphabetical order, i.e. Appendix A, Appendix B, etc.

7 Sources of Further Guidance

We have said several times that this guide is not complete. Textbooks usually demonstrate good technical writing especially if they are produced by a reputable publisher. They also provide illustrations of the use of descriptive devices, etc.

Appendix B contains a list of other books and Internet resources, specifically designed to assist in writing essays and reports. While writing this guide, we have tried to follow the guidelines it contains. Obviously though, a guide like this has a different purpose to that of a project report so total adherence has been impossible. We have tried to set an example, but it is not perfect.

Finally, there will be specific aspects of your project report that only your supervisor can advise you on. It is important that you discuss an outline of the project report with your supervisor before you begin to write up.

8 Conclusions

These are our main recommendations:

- Record all relevant information generated by the project:
  - use a notebook,
  - keep a diary,
  - log debugging sessions.
- Gather further material from publications or other external resources.
- Organise the material into sections agreed with your supervisor,
  - e.g.“Background”, and so on.
- Turn this material into written prose to form the project report’s main body.
- When writing the main body
  - keep your readership in mind;
  - identify commonality;
  - use sections and subsections;
  - follow stylistic conventions.
- Where appropriate use
  - cross-references,
  - references,
  - figures and other descriptive devices.
- Produce all required supporting structures according to convention, after completing the main body, and include this material in appendices to avoid disrupting the flow of your narrative.
- For examples to follow, look at
  - **textbooks** from reputable publishers,
  - the way this guide is written.
- Discuss an outline of the project report with your supervisor before you begin to write up; this will help you to plan your project. However, we strongly recommend that you write up your work as much as possible as you carry out your project, rather than leaving the writing to last.

We hope you will find this guide to be of value in completing your project. If you have any comments, or wish to suggest good sources of advice that you have found, please let your project supervisor know so that we can update these guidelines.

Good Luck!!! ☺

Appendix A: Typesetting Rules for Report Presentation

MSc students should check current university submission regulations, which take precedence over the rules given below.

Length:	Reports longer than 15,000 words, not including the supporting structures, will not be accepted. There is no minimum length but it is mainly through the report that your project will be judged so the report should adequately reflect the work done in the project.
Font Size:	Reports should be printed using 12pt typefaces.
Line Spacing:	Reports should have single line spacing such as this guide. The report should be economical on paper. It should not, for example, contain excessive amounts of white space. Only the major sections (as illustrated in Figure 3.1) need to begin on a new page.
Submission:	Reports should be typeset with some word-procecssing system, e.g. LaTeX, OpenOffice, LibreOffice or Word. The final project report should be presented as a PDF file with any other documentation in the appendices of the report. Artefacts produced for the project that are to be processed by other software such as a compiler or interpreter should be submitted separately in an archive file. Typically this will be the source code for software developed for the project. Detailed submission guidelines will be made available separately.

Appendix B: Bibliography

Bly, R. *10 Ways To Improve Your Technical Writing*, Center for Technical Communication. http://smartbiz.com/sbs/arts/bly10.htm [accessed January 2011].

Capital Community College Foundation. *Guide to Grammar & Writing*. http://grammar.ccc.commnet.edu/grammar/ [accessed January 2011].

Creme, P, Lea, MR. 2008. *Writing at University: A Guide for Students. 3rd* edition. Open Unniversity Press.

Contains help for all aspects of writing while at university.

Document Foundation, The. *LibreOffice Documentation*. http://www.libreoffice.org/get-help/documentation/ [accessed January 2011].

Duprė, L. 1998. *Bugs in Writing: Guide to Debugging your Prose.* 2nd edition. Addison-Welsey.

Contains tips in small, individual chapters to improve your writing; a good reference book.

Fry, R. 2004. *Improve Your Writing.* 5th edition. Delmar Cengage Learning.

This is a guide for students producing a written project. Easy to read, full of handy hints, and guides you through the whole process from carrying out research for your report through to producing the final draft.

Landsberger, J. *Study Guides and Strategies*. http://www.studygs.net/ [accessed January 2011].

LaTeX Project Team. *LaTeX - A document preparation system*. http://www.latex-project.org/ [accessed January 2011].

Oracle. *The OpenOffice.org Documentation Project*. http://documentation.openoffice.org/ [accessed January 2011].

Strunk Jr, W, White, EB. 1918. *The Elements of Style.* WP Humphrey, Ithaca, NY. http://www.crockford.com/wrrrld/style.html [accessed January 2011].

Excellent guide to good use of English, a classic reference book, meanwhile updated and republished many times.

Word MVPs. *The Word MVP Site*. http://word.mvps.org/ [accessed January 2011].