Writing a Project Report

Peter Coxhead

Projects can be very different: all comments made in these notes must be adapted to your particular project.

If in doubt, consult your project supervisor.

Read the Project Guidance Notes (start at http://www.cs.bham.ac.uk/internal/courses/projects/index.html)**. This document does not replace the Notes.**

Style

Your report should be written in 'scientific English'. What this means is not easy to define. Some of the suggestions below are taken from Avison (1981). Note that this handout is not in scientific style!

Use a formal rather than a colloquial style

- Don't use contractions: for example, write *is not* rather than *isn't*.
- Don't use informal words like *lots* (e.g. replace *this caused lots of problems* by *this caused many problems*).
- Don't use emotive words (e.g. *terrible*, *awful*) or slang.
- Don't use *I* unless you are referring to a specific individual choice you made.[1] Thus in the introduction, if you are explaining why you chose a particular project, then it's ok to write *I decided* because this was your own individual choice. On the other hand, if you review a number of alternative ways of solving a problem and then select the one which you have argued is best, I would avoid the use of *I*, since here you are claiming that anyone, not just you, would have made the same choice. So rather than *After reviewing the possible design choices as discussed above, I decided ...*, I would write *After reviewing the possible design choices as discussed above, it was decided ...* In general, expect to use quite a lot of passive sentences: *the system was constructed* rather than *I constructed the system*.

Good writing is simple writing

- Writing in a formal style doesn't mean that your report should be hard to read because it's full of long words and complicated sentences. Concentrate on getting over the information (facts, reasons) to the reader as clearly as possible.
- Don't use unnecessary jargon. Keep your intended audience in mind (for the Project Report this is a competent computer scientist who is not necessarily a specialist in the area of the project). By all means use common technical terms in the field, but avoid unnecessary jargon or long words which are just designed to impress. You won't.
- Don't 'pad out' your report. Markers won't be impressed by length.
- Long and complex sentences should be broken up. The same is true of paragraphs. (On the other hand, sequences of very short sentences and paragraphs are irritating.)
- Spelling, punctuation and grammar should be **perfect**. Use appropriate tools to check. Spelling checkers are normally good (don't forget to set them to British English). In my personal experience, the grammar checker in Microsoft Word is of limited use. Try to get someone else to proof read your report: remember that other people are generally better at spotting your mistakes than you are.

[1] Some people say never use *I*. Check with your project supervisor if in doubt.

Plagiarism and Referencing

Plagiarism is cheating by claiming someone else's work as your own. Don't do it! Penalties can be very severe: for example, in 2002/03, three Computer Science students failed to get a degree because of plagiarism in their projects. Fortunately nothing like that has happened since.

Accusations of plagiarism in a report can always be avoided by quotation and proper referencing.

- You **cannot** copy and paste **any** text from **any** source unless it is **quoted** to make it clear that it is copied and not your own words.
  Short quotations should be put in double-quotes like this example: *Avison (1981) states "Following these rules will ensure that you are not accused of plagiarism."*
  Longer quotations should be made into a referenced 'block quote' section: separate paragraph(s) indented from both the left and the right.
- Material which is quoted must be **referenced in the text**, that is there must be a **direct indication in the text** of its source, using some standard convention (e.g. numbers such as [1], or names and dates such as *Smith 2001*).
- Material which is **paraphrased** or otherwise **directly used** must also be **referenced**.
  Where you paraphrase ideas or use re-write material in your own words, you must still acknowledge the source, e.g. by writing *The outline presented here is based on Smith (1999).*
- There is a distinction between a **list of references** at the end of a piece of work and a **bibliography**. The list of references gives full details of the references in the text, whether these are numbers or names/dates. The bibliography lists sources you consulted that have contributed to the work in a general way, but which are not specifically referenced in the text. Usually the list of references should be much longer than the bibliography.
- For one style of referencing see http://www.cs.bham.ac.uk/~pxc/refs. Use this style unless you have agreed the use of a different style with your supervisor.
- You are not expected to re-invent the wheel; indeed you can legitimately be penalized for doing so. It's good software engineering practice to re-use code. But you **must** make clear which parts of your code are taken from elsewhere and which are original. Carefully commenting your code can achieve this.

You also need to take precautions against other people copying your project, leading to you being falsely accused of collusion. If your code and project report are on a networked machine, make sure the permissions do not allow viewing. Put your name as author in every Java class.

Ordering and Content

Remember that the project report should tell a story. This doesn't mean telling the reader what you did, step by step – we're not interested in an autobiography. It means making sure that the sections of your report follow logically one after the other and add up to something coherent. A key to this is **making your account linear**. YOU know what comes next; the reader doesn't. *Time after time I read project reports which I can't properly understand because the author has assumed that I know what comes later*. We all know that actual project work is cyclic, iterative, even confused. But the report mustn't be. While it mustn't falsify what you did, it has to present it in a logical order.

The diagram on Page 4 offers one model for the content of the body of a 'typical' project – it is **not** intended to be rigid guide, nor to define numbered chapters or sections. Read the Project Guidance Notes as well; some issues discussed there are not covered here.

- The first component (Introduction) sets broadly what you aimed to achieve and why. The final component (Summary/Conclusions) must relate closely to the first. In my view, it should be possible to read these two without reading the rest of the report, and know what the aims were and how far they were achieved. Don't be afraid to write a summary which repeats what is elsewhere: that's the point of a summary.

- The Background/Review should **only** review material where the Introduction makes it clear why it is included. Don't put in interesting but marginally relevant background material. The reader should always be clear as to why background material is being reviewed.
- It's generally useful to have a short component (I've called it the 'Project Specification') which basically says "well, my aims were originally these, but now that I've reviewed background material, this is more precisely what I'm going to do." It's quite acceptable to change or refine some of the original aims of the project, but the reasons should be made clear. (Note that this component isn't to do with software: it's not the requirements definition or the software specification. It's where you show how the original aims of your project were refined by your background reading.)
- Problem Analysis and Solution Design are what we want to read. Implementation details should be avoided unless some particularly tricky issue arises. Concentrate on explaining the decisions made and the reasons for them.
- Validation/verification in general and testing in particular are important. Describe your testing strategy, not all the details – by all means put details in an appendix. Convince the reader that you've thoroughly tested/validated/verified all the stages of your work.
- **The Evaluation is very important.** What lessons were learnt during the course of the project? Evaluate (with hindsight) both the product and the process of its production. Review your plan and any deviations from it. Don't forget to sell your work! Tell the reader what was good, what was achieved. Be honest about limitations – these can be suggestions for future work.

After you've written each component, ask yourself:

- Does this follow from what went before?
- Have I made it clear **how** it follows from or relates to what came before?
- Does it assume anything I haven't yet explained?
- Does it 'point forward' to what comes next?

Your report isn't a 'mystery novel' where the reader is supposed to look for clues as to what's going on!

Accreditation

To be acceptable for BCS and IET accreditation/exemption, your project must conform to their requirements. Most of these are also School requirements for all projects. In particular:

1. The report should demonstrate an appropriate level of professional competence in the practical development of a suitable application, tool or similar product.
2. Projects must give a clear description of the life-cycle stages undertaken (including validation and/or verification as required), describing the use of appropriate tools to support the development process.
3. Evaluation of your work is important. This must include not only the outcomes, but also the process and what you have learned from undertaking the project.

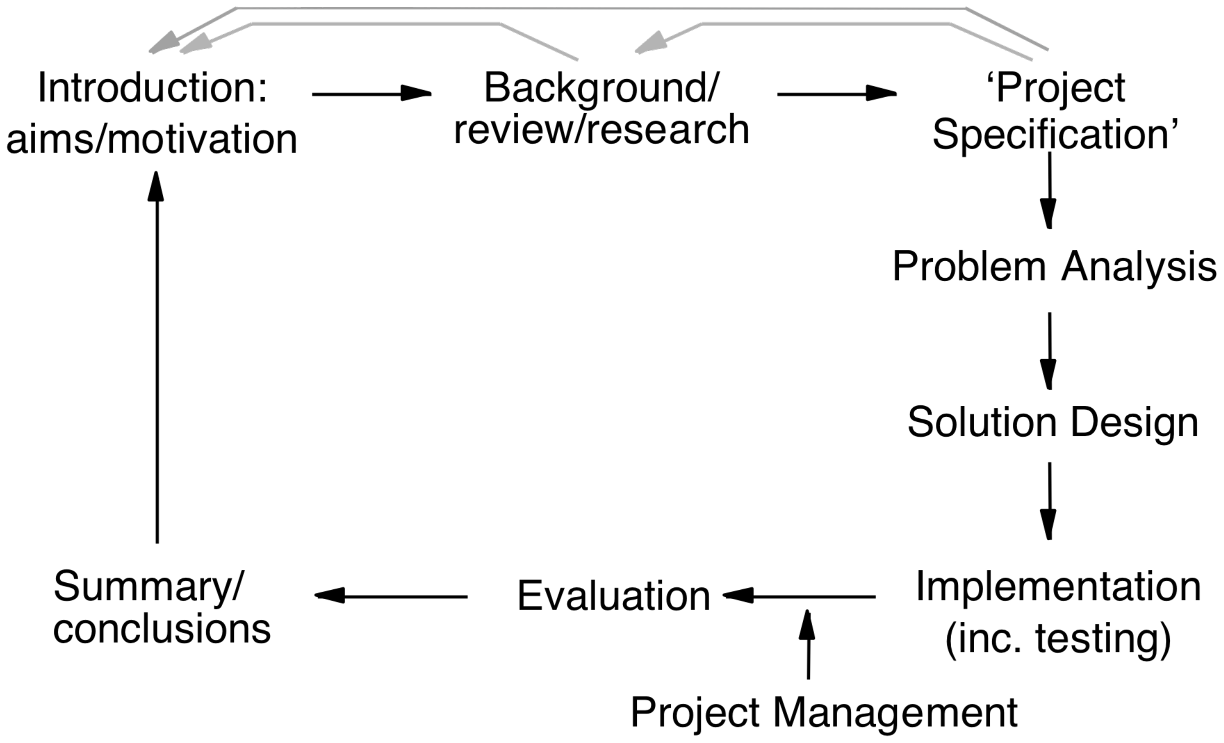
A project acceptable to the School (e.g. a theory project) may not be acceptable to the BCS or IET. If you want accreditation, pay careful attention to the points above.

References

D E Avison (1981) *The Project Report: a guide for students* (2nd ed.), Computer Centre, The University of Aston in Birmingham.

Components of a ‘typical’ project report

See the Project Guidance Notes for other required components.
Grey arrows show some backward links which may be missed.
Note: the diagram is not intended to be self-explanatory, but used in conjunction with the handout and lecture


Introduction: aims/motivation
Background/ review/research
‘Project Specification’
Problem Analysis
Solution Design
Implementation (inc. testing)
Evaluation
Project Management
Summary/ conclusions