Guidance on writing project reports

Martín Escardó

School of Computer Science, Birmingham University, UK

March 14, 2011

Resources and bibliographic references

This presentation is available at the following clickable links:

http://www.cs.bham.ac.uk/~mhe/project-guidance-2011.pdf

It is mostly based Peter Coxhead's notes:

http://www.cs.bham.ac.uk/~pxc/proj/ProjectReport.html

Warning

The report itself gets 30% officially.

But it is chiefly by *reading the report* that we give the marks to the other components of the project:

- Quality of Product (20%)
- Quality of Process (20%) - based solely on evidence in the report
- Quality of Demonstration (10%) - 2nd reader only
- Quality of Management (10%) - supervisor only
- Substantialness of Achievement (20%)

Projects can be very different

All comments made in these notes must be adapted to your particular project.

You must always consult your project supervisor.

Typical structure of a project report

Title
Abstract
1. Introduction
2. Background
3. Research (if applicable)
4. Project specification
5. Problem analysis
6. Solution design
7. Implementation (including validation and testing)
8. Evaluation
9. Summary and conclusions

Bibliography
Appendices (A. How to run the software B. Etc.)

Hints

- Go to the School library and look at examples of project reports.
- Produce a preliminary table of contents and discuss it with your supervisor.
- Fill the various empty sections with short summaries of what you will write.
- Write that.
- Read and rewrite that.

Hints

- And read and rewrite that.
- Move things in the wrong place to a better place.
- Repeat the process until you are satisfied.
- Don't be afraid of throwing away big chunks you have written with great pain.
- Start chapters or sections from scratch if you or your supervisor are not satisfied.

After you've written each component, ask yourself

- Does this follow from what I said 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?
- Have I made it easy for the reader to get a general feeling of what the general "picture" is?

Explain what you are going to say in later parts

- Your report isn't a mystery novel where the reader needs to look for clues as to what's going on!
- Have both forward and backward references to later and earlier chapters.
- Readers don't necessarily read the report in the order it is written.
- In fact often they don't.

Style

- Don't use itemized lists except in very few situations where it is appropriate.

- A report is not the same thing as a PowerPoint presentation.

Style

Your report should be written in "scientific English".
Use a formal rather than a colloquial style.

- Don't use contractions such as "don't".
- Don't use lots of informal words like "lots"

  E.g. you can use "many", "several", "a number of", etc.
- Don't use emotive words (e.g. terrible, awful, incredible) or slang.
- Don't use "I" unless you are referring to a specific individual choice you made.
- It common to use "we" in other cases.

Examples

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.

Examples

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.

Examples

So rather than

"After reviewing the possible design choices as discussed above, I decided . . . ",

you should write

"After reviewing the possible design choices as discussed above, it was decided ..."

or

"After reviewing the possible design choices as discussed above, we decided ..."

Examples

In general, expect to use quite a lot of passive sentences:

"the system was constructed"

rather than

"I constructed the system".

Or

"We 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 (XFDPK helps implementations using DDG to run significantly faster, especially in combination with RTXH).
- 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 do use common technical terms in the field. But explain it.

- 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 can be irritating.)

Spelling, punctuation and grammar

- Spelling, punctuation and grammar should be perfect.
- Use appropriate tools to check.
- Spell checkers are normally good (don't forget to set them to British English).
- Some people say that the grammar checker in Microsoft Word is of limited use (I don't use it).
- Try to get someone else to proof read your report: remember that other people are generally better at spotting your mistakes than you are.

  (E.g. your mother, sister, friend, colleague or flatmate.)

Plagiarism and Referencing

- Plagiarism is cheating by claiming someone else's work as your own.

  (Whether explicitly or implicitly by omitting a reference.)

- 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.

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

Quoting and citing references

- 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)").

Quoting and citing references

- Material which is paraphrased or otherwise directly used must also be referenced.

  Where you paraphrase ideas, or 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)".

- Peter Coxhead says "There is a distinction between a list of references at the end of a piece of work and a bibliography".

  But I haven't seen this distinction in my field of research.

- For one common style of referencing see

  `http://www.cs.bham.ac.uk/~pxc/refs/`. (Discuss with supervisor.)

Final note on plagiarism

- 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.

Extra plagiarism precaution

- 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.
- E.g. put your name as author in every Java class you have written yourself.

Ordering and content

- 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.
- This 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 we read project reports which we can't understand because the author has assumed that we 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.

Typical structure of a project report

Title
Abstract
1. Introduction
2. Background
3. Research (if applicable)
4. Project specification
5. Problem analysis
6. Solution design
7. Implementation (including validation and testing)
8. Evaluation
9. Summary and conclusions

Bibliography
Appendices (A. How to run the software B. Etc.)

Validation and testing

- 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.

Evaluation

This is very important. But sadly students tend to neglect it.

- 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.

Good luck