Awesome Projects in Computing!

How to choose research, conduct and write excellent projects in Computer Science, Computing, Multimedia and related subjects

By Sarah Mount and Peter Every

Sarah Mount. School of Technology. University of Wolverhampton
Peter Every. Department of Computing. Coventry University

Chapters

Introduction

Sarah and I spent nearly five years sharing an office together. Sarah has a background in Computer Science and Mathematics. My background is in Linguistics and Cultural Studies. We always had interesting conversations about the differences and similarities of research within the Sciences and the Humanities.

We talked about student projects a great deal, mostly what was wrong with them. Out of these conversations Sarah started blogging excellent articles of advice about projects in computing which can be found at: http://projectsuccess.posterous.com

Everything presented here is by Sarah (with some commentary and a research primer by me). One day this will become a book.

Peter Every 20/10/11

Chapter 1

How to choose a good BSc project

A critical part of the success or failure of any project is the initial choice of what to work on. This is a surprisingly difficult part of any project, in some ways the most difficult part, and it's something that we see students struggle with year on year. Nothing is so disappointing than marking a project and coming to the realisation that, with some better decisions at the beginning of the year, a failing project could have passed. This is a trap to avoid, and by avoiding it you will not only improve your chances of passing your project, you will greatly improve your chances of getting a first. In fact, projects are pretty straight forward to do well in, so long as you fully understand what is expected of you. This chapter takes you through what you need to focus on and avoid right at the start of your project journey.

Do something you are interested in

A final year project is a six month, single person project and in most Universities in the UK. Students will have to study several other modules concurrently. This is a long time to be working on a single piece of coursework, so it is important to choose a project which will hold your attention for that length of time. Moreover, you will be working on other things at the same time, so ideally you need to choose a project that is compelling enough that you *want* to work on it, in preference to doing other things.

How to know what you are interested in

This might seem like a rather unnecessary topic - what is "interesting" is very personal and individual. However, estimating what you might find interesting in several months time, when you are under pressure to meet deadlines is not easy. One trick to weight the odds in your favour is to choose a project which you do not, at the start of the project, entirely know how to complete. Like Einstein said: "If we knew what we were doing, it wouldn't be called research". Obviously, don't choose something that is completely outside your area of expertise. If you have spent two years studying bioinformatics then don't suddenly decide to try a dissertation in fine art, but equally, if you know exactly how to complete every part of the work that you will need to do for your project then your idea is not "big" enough in scope. This really is the key to finding a project and much of the rest of this chapter expands upon it: **a thesis or dissertation is not simply a long piece of coursework**; it is an individual, self-contained work which should stand on its own. Think of it as a sort of "first job". When you leave University and apply for further study or a job, then the results of your project will be part of the professional portfolio of work you can use to convince a future employer to take you on.

Project difficulty: a difficult project is an easy project!

By far and away the biggest mistake that we regularly see from students writing project proposals is choosing a project which is far, far too easy to complete. The train of thought seems to go ... projects are difficult, I want to make the project easier; therefore, I will

choose a simple idea to work on. The classic examples of this in Computer Science are "a website with a database" -- usually for a family member or friend who runs a small business - or occasionally a website or database on their own. What's wrong with this? Well... so many things:

1. By the time a student has reached the final year of their degree, they will likely already have written several databases, websites and at least a couple of websites-with-a-database. Therefore, the project is something that the student has already been awarded credit for. This means that the student will not be demonstrating that they can learn independently, and go beyond what has been taught in lectures, which is one of the main purposes of the project.
2. Because the proposal is about the same size and quality as an individual module coursework, it is not large enough in scope to gain many marks.
3. An individual website using ASP, PHP, or similar, for an SME is a very old problem for which there exist a large number of "turn-key" solutions - that is, off the shelf products that can be used to create the product. These include templating systems such as Joomla, cloud-based solutions such as Google Sites, Posterous, Tumblr and so on, wikis, and a number of other technologies. A straightforward website-with-a-database is, therefore, in no way a demonstration of the student's ability to work at the cutting edge of their field.
4. A website-with-a-database is not a problem; it's a solution to a problem. A project proposal should propose an interesting problem, with a suggested strategy for solving that problem during the progress of the project.

Having said this, I have seen and indeed supervised a number of excellent projects, for which the student implemented some sort of website and some sort of database. So, it's not that websites or databases are inherently bad choices as solutions to the problems posed by a project proposal, but a proposal MUST overcome the four problems outlined above.

The heading for this section said (rather confusingly) that "a difficult project is an easy project". What I mean by this is that the "difficulty" of a project is something that will uppermost in the mind of the staff marking your thesis. A "difficult" project is likely to be looked upon favourably because it will be a bigger step away from what you have already been taught, you will need to be reading more academic literature, you will be showing more independent learning, and so on. These are some of the most important factors in getting a good grade, and far outweigh factors such as finishing every part of your practical work. The up-shot of this is that if you choose a "difficult" project and complete it quite poorly, you are likely to get better marks than a student who chooses an "easy" project and completes all of their practical work. If you did choose to work on a website-with-a-database-for-an-SME then the proposal will be so easy that you will really have to complete every part of the project perfectly just to get a pass. ***So, choose a small but difficult project***.

> *[Peter's thoughts: I agree with this completely. The way I grade projects is similar to the way high-diving is scored at the Olympics. In high-diving, the diver nominates the difficulty of the dive they are about to do and the judges grade how well they achieved the difficult dive. So a 'perfect' dive of difficulty 2.0 still scores lower than a 'three quarters perfect' dive of difficulty 4.0]*

Have a research question

In the last section I said that a project proposal should pose a problem, not a solution to a problem. Ideally, it is best to phrase this as a research question, such as the following:

- Is algorithm X more efficient than algorithm Y?
- Is it possible to implement product Z on the cloud?
- Can feature L be added to programming language P?
- Can theorem T be proven?
- Can algorithm Z be adapted to be used in conditions D?

... and so on. There are several advantages to this. One is that this is a standard form of writing in academia, and your project will be marked against academic criteria. Secondly, if the aim of your project is to answer a question then you leave the issue of *how* to answer that question reasonably open ended. It may be that you have a very clear idea, at the start of the project, what you are going to do. That's fine, but as you progress through the project you may well find literature that enlightens your views on how your question can be answered. Thirdly, your answer to the question may not be what you expect. That's fine, it's OK to find out that actually, your algorithm isn't as efficient as you thought, or the theorem cannot be proved, so long as you give solid, convincing evidence for your answer.

Do something practical

If you are working in the sciences, it really is important that you do *something* practical as part of your work. For these purposes "practical" can mean experimental work or mathematical work - it's OK to prove a theorem, for example, as the main part of the "practical" content of your work. What you should avoid though, is vague, nebulous, ‘thought-pieces’, which have no clear results and cannot be evaluated. Avoid anything with a title like "an investigation into X" or "a dissertation on Y". These sorts of writing are well accepted in the humanities (BA courses), but for a scientific (BSc Courses) piece of work you need to propose a question and find some answer to it. Equally, a literature review is not really a project in itself; it needs some research question and evaluation with it to form a complete project.

Focus on evaluation from the start

Evaluating your work will likely be the last practical work you complete before finishing your project writing. However, you should know from the start of your project how you plan to do this. As with unit-testing, or usability testing, it is best to have designed you evaluation in as much detail as possible before you start you practical work. That way, you know that what you are aiming for is something that can be evaluated in the manner in which you have planned. Remember, the purpose here is to determine whether your project has answered your original research question.

In general, your evaluation will fall into one of the following categories:

- **Performance evaluation:** either testing the speed, memory footprint, scalability, load-balancing, or other aspect of the performance of a program or system. This is often the easiest form of evaluation -- it can be performed by a program and so automated, the results can be analysed and presented using a statistics and you will not be reliant on users. Work in programming languages, networking, operating systems, databases, and hardware tend to suit this sort of evaluation well.
- **User-acceptance testing and usability:** if your project involves creating a product for end-users to test, especially if you have an industrial client, then it is essential that you perform some sort of user acceptance testing. Good options for this are the talk-aloud protocol or semi-structured interviews. NEVER, EVER, EVER think that a "heuristic" evaluation is sufficient. Heuristic methods only catch basic errors; they tell you nothing about how your users will actually experience your product.
- **Formal or semi-formal methods:** such as proving a theorem, using a model checker (such as SPIN), using a formal method such as B or Z to show that your work is free of particular types of errors.

Take (academic) advantage of your supervisor

Every student will have at least one supervisor, who will usually be actively involved in research, consultancy or something similar. This sort of work can provide a wealth of good ideas for projects and has several advantages. Firstly, your supervisor will propose projects that have the right scope and difficulty for your degree course. Secondly, if your supervisor has an interest in what you are doing, they will have a vested interest in seeing you succeed and of course will have a lot of relevant expertise with which they can advise you. Lastly, it is likely that your work will be used by other members of a research group which will give you access to feedback on what you have done.

Be flexible (within reason)

Remember that a project is a marathon, not a sprint. It may well be that you get part way along the journey and find out that what you had first set out to do is actually impossible, or impossible within the scope of the project. Or it may be that you find some other way of answering your research question, or you uncover some literature which shows that the question can actually be answered very simply. In this case, you should speak with your supervisor and find a way to reword or even completely change your original research question. This is quite a reasonable thing to do and happens often in "real" research projects, so you should not be worried about it. Your final project does not have to match the original proposal exactly, but you should be able to explain why the changes you made were necessary.

Summary

- DO choose a project that will hold your interest for the duration of the project.
- DO NOT choose a project that is the same size or scope as a coursework, or something that is very similar to work you have been set in a module.
- DO propose a "difficult" problem -- it is easier to pass a challenging project than an "easy" one!

- DO propose a research question, and an idea for solving it.
- DO propose a project with some sort of practical or mathematical component, DO NOT set out to write a commentary on a topic.
- DO have a very clear plan for how you will evaluate your project. This should clearly state how you will determine whether or not you have answered your research question.
- DO NOT evaluate an end-user product with only heuristic methods.
- DO test end-user products with real users.
- DO take advantage of the expertise of your project supervisor and their research interests.
- DO be flexible, if you find that your original research question cannot be answered, or if you find that a more "interesting" research question emerges during your project.

Chapter 1 Workshop: How to choose a good project

(Workshops are exercises to do with your supervisor or alone to help better understand the concept of a research question.

You have already read about what makes a good project and the importance of the research question that you choose. Your research question is absolutely central to everything you do in your project and all of your reading and practical work will be guided by it. So, it's very important that you choose a good one! We can't help you with choosing a project that you find interesting and engaging, that's up to you, but we can help you formulate a valid and sensible research question. Below are some examples of poor research questions with feedback. Read through these and then give your own feedback on each of the research questions in the Case Studies that follow.

Example research questions with feedback:

1. An investigation into fingerprint detection algorithms
    a. This isn't an actual question, so it needs to be rephrased.
    b. The question needs to be specific about what the focus of the investigation is. Is it the accuracy of the algorithms? The performance (i.e. how fast they are, etc)?

2. Can MySQL be used with PHP websites?
    a. This question is too basic. There are already thousands of PHP / MySQL websites out there, so there is little for the student to add. The question is already answered.
    b. It's likely that the student has done something similar in a coursework so this isn't challenging enough for a final year project.

3. How will RFID adoption change IT businesses?
    a. This question is so broad it is unlikely that the student will be able to answer it in the time available for their project.
    b. This question will most likely result in a 'Report into …' style dissertation that lists only secondary sources. If the student is on an IT degree this may be a relevant area to work on. For a Computer Science or related degree title this question falls somewhat outside the area of the degree title. This may mean the student loses marks for "difficulty" or "challenge" but it is also likely that the supervisor will not have the requisite experience to supervise the project.

Case studies for your feedback

Pick five of the following Case Studies. For each:

1. Write down your own feedback (in the style of the research question feedback above)

2. Suggest what sort of degree title (e.g. ‘Computer Science’, ‘Multimedia’, ‘Business IT’) the question might be relevant to
3. Provide a new research question that might be more appropriate for a final year project.

- Can the University homepage be better designed for search engine optimisation (SEO)?
- Is ‘Go’ a good programming language to use for websites?
- Is Groovy or Ruby the better programming language for novice programmers?
- Is PHP a suitable language to build websites that scale to over 10,000 hits per minute?
- Are Web 2.0 techniques useful for library websites?
- Can NullPointerExceptions in Java be detected at compile time?
- Which is the best commercial games engine?
- Is the SCRUM better than SSADM?
- Is there a good way to search video data?

Chapter 2

How to pass your final year thesis project

I've been thinking for a while that it would be good to distil some of the advice that colleagues and I give to students doing final year thesis project, so here it is - enjoy!

Think of yourself as an academic.

Whatever you want to do when you leave University, your work will be written for academics and marked by them. A thesis project is like a small research project and to get the best marks available you should run your project with that in mind. So, you need to perform a thorough literature survey, follow a sound methodology, critically analyse your results and so on.

Have a hypothesis (or a thesis statement or a research question).

A lot of students approach their projects by saying "I want to do this", like "I want to invent a Widget, written in Python", “I want to do a website”, “I want to make a game”. This is not the best way to get good marks. Your project needs to have some sort of clear purpose and in the academic world it's best to state that purpose as a hypothesis, thesis statement or research question. These three are really just different ways of stating the same thing and which you choose is just a matter of taste. So, if your project is a study on the ratio of smokers which develop cancer (e.g. "I want to do some statistical analysis on smoking") then you might phrase that in one of these ways:

- Smoking is correlated with cancer -- *hypothesis*
- Smoking is correlated with cancer -- *thesis statement*
- Is smoking correlated with cancer? -- *research question*

Or, perhaps you want to design a new GUI widget to replace list boxes:

- Users will find MyWidget easier to use than standard ListBoxes -- *hypothesis*
- Users will find MyWidget easier to use than standard ListBoxes -- *thesis statement*
- Is MyWidget easier to use than a standard ListBox? -- *research question*

Part of the point of phrasing your work like this is that it should change the way you think about the work. You now have a very clear goal to reach -- to prove your hypothesis / thesis statement or to answer your research question. Everything you do in your project should now be directed towards this one goal.

Also, you have reduced your risk of failure! What if MyWidget turns out to be rubbish? Well, then you have still answered the research question or disproved the hypothesis - **you've got a result!** In your write-up you can probably say a lot about why MyWidget wasn't as good as you thought; how you achieved your results and how other researchers can use your work to invent even better widgets.

Produce something useful to others.

So, you've got a hypothesis to answer. Great! But your teachers will still want to be convinced that the hypothesis you've chosen is worth six months of your time and lots of hours of their time. How do you know if your work is useful? Firstly, make sure you've read the relevant literature. Use Google, go to the library, and talk to potential users. There should be a group of people in the world who have a clear reason to be interested in what you're doing. That might be a section of the research community, a user group or a company, whoever - but there must be *someone* who wants your hypothesis validated or disproved. You should convince whoever's marking your thesis that this group of people exists by explaining in your thesis what the current literature says about the area you are working in and how your work contributes to this area.

Do something (a bit) novel.

You're not expected to win a Nobel Prize on the back of your undergraduate work. However, there's no point in doing something that everyone has done a million times before. The worst example of this in Computer Science is a project which is basically "I'm going to write a website with a database". Usually the website is for a friend or relative and the database keeps track of users. There's nothing wrong with that if there's some real novelty in it (e.g. you've just invented a new sort of database and this is a demonstrator for it). But often this is something that most first years could complete for a coursework (so, it isn't stretching you) there are simple, free tools that can do the job for you (it's not novel) and the user is bogus (no one is interested in it). Choose wisely!

Have clear success / failure criteria.

This should be taken care of when you write your hypothesis (or thesis statement or research question). However, it's important to know what will constitute a "successful" project for you. What results do you want to end up with? Once you know that, you can choose the appropriate methods to use in your work (e.g. will you be running a focus group? Writing a questionnaire? Writing some programs -- and testing them somehow?). You can also think about reducing the risks in your project. What if you don't finish part of your work on time? Is there some catch-up time in your schedule? What if early tests go badly? Is there time to repeat them?

Don't change the world.

Don't choose a project which is just far too big. If you're going to invent a new computer, writing an OS from scratch and a windowing environment for it is not a one-person, six-month, part-time job. Keep it small and give yourself some scope for extending the project if it goes better than you thought.

Even if you've chosen something small with a clear hypothesis, once you've written out a project schedule it will still feel like far too much to do. You *will* feel overwhelmed. The trick to reducing your fear early on (and making sure you work to schedule) is to break down your work into small tasks. Each task should have its own goal and deliverables with its own

success criteria and a deadline. So, if your hypothesis is "MyWidget is easier to use than ListBoxes", your sub-goals might be:

- Produce a literature survey on list widgets.
- Write comprehensive unit tests for MyWidget in the Gtk widget set.
- Implement a MyWidget in the Gtk widget set.
- Debug (goal is to pass all unit tests).
- Devise usability experiments (deliverable: methodology document).
- Write application software for testing in the experiments.
- Perform experiments.
- Analyse data.
- Write-up thesis.

Note that these tasks are dependent on one another. MyWidget cannot be written before you know how to test it. The experiments cannot be run until you have designed them and decided how to analyse the data they will produce.

Your new list will make life a lot easier, but you will probably still feel that it's too much to handle. To feel better about your work and motivate yourself, it's a good idea to take each of your sub-goals and write out the next *physical* action you need to perform to carry out that task. So, for the literature survey the next action might be "Google for 'list widget'". For the unit testing your next action might be "Find documentation about the Gtk testing framework". And so on. Most people find action lists much more motivating than task lists.

Choose a project that will maintain your interest.

Six months is a long time. If you choose a boring project (maybe you think it'll be "easy") then you'll quickly lose interest, get bored, stop working and possibly fail. In some ways, it's good to choose a project with a lot of scope (so you can change direction a bit and still address your hypothesis) in an active area of research (so there's lots of work to build on). On the other hand, some people would find that sort of project unfocused and confusing.

When you plan your project, think hard about what motivates you. Is it reaching towards a really interesting goal, or the fear of failing really badly? Either way (or both) you need to keep that motivation in mind whenever you're working. So, choose something interesting to do, give yourself at least a little scope for changing the details of the project over the year and keep in mind why you want to succeed (or not fail!).

Don't be too dependent on clients.

If you have a client to work for, especially one from industry, be careful about how you plan your project. If you are expecting resources from your client what will you do if they don't turn up on time? If you want the client to test your work, what do you do if they get a major order in when you're ready to test and no one has the time to help you out? What if the client goes out of business? What if you have to sign a non-disclosure agreement -- can the University still mark your work?

Having a "real" industrial client can be a big bonus. For one thing it's very easy to say that someone is really interested in what you're doing for the project. However, you need to make sure that if the client pulls out or doesn't cooperate you'll still have a viable project. Plan well and you won't have any problems.

Understand that research is not reading and a thesis is not a report.

Most undergraduates (at least in Computer Science) seem to be pretty confused about what research really is. It certainly *isn't* about using Google or reading in the library. Research means adding something new to the body of knowledge on a particular subject. This is why it's so important to know what work has already been done (so you know your work is novel), to have a clear hypothesis (so you know what new understanding you're adding) and to write up your work well (so other researchers can use it).

Also, understand the place of your thesis. You are *not* writing a report which tells people what you did. You are writing a *thesis* which tells people about the research you have done. This can be structured in whatever sensible way you prefer, but it needs to have the following parts:

- **An introduction.** What's your hypothesis? Why is your work interesting? What are your trying to achieve?
- **A literature survey.** What have other people done? What new knowledge will your work add? What is the current state of the art missing and how are you going to address that?
- **Your methodology.** How did you go about validating / disproving your hypothesis? Why is your method sound? Why should anyone trust your results?
- **Your results.** What did you do? How?
- **Your analysis of your results.** What do your results mean? Why are they interesting? Did you validate your hypothesis or disprove it?
- **A reflection on the management of your project and the social, legal or ethical issues that you needed to consider**. Your first supervisor may have a very good idea of how well you tackled your project - however second supervisors may not have any idea. For this reason you need to include an account of the conduct of the project. What problems you encountered, how you overcame them, how diligently you worked, how you sought advice.
- **Conclusions.** What did your work contribute and how could it be continued by others?

Eat well, sleep well, get some exercise and take a day off every week.

Basically, look after yourself. To work productively you need to be in good physical and mental condition. If you feel ill or you are not coping well with life, slow down a bit and take a break. Don't eat junk food all the time or you'll feel sleepy and miserable. Cut down on caffeine and alcohol or you'll get stressed and sleep badly. Sleep a sensible number of hours every night -- consistency is important. Get some exercise because you need endorphins to keep you happy and some oxygen getting to your brain. Omega-3 and the sorts of minerals

that aren't found in hot dogs will keep your brain working sensibly.

Most of all, take a whole day off University work every single week. It doesn't matter what you do with that day (have fun, earn money, write a novel, whatever) but working every day will limit your creativity massively. Most very creative (and productive) people find that their best ideas come after a day off. This gives your mind a chance to consolidate all the material you have learned, synthesise it and solve some of the problems you've been considering. In fact, to revise for an exam or solve an interesting problem, it's a good idea to spend a few days working really hard at reading everything you need to know and taking notes, then take a day off directly before the exam or the day before you're going to write the solution to your problem. This will give you the best chance to properly understand everything you're working and be creative about it.

Be productive but don't spend (too much) time on productivity.

You need to organise yourself well, which is a difficult problem in itself. However, if you spend even five per cent of your time on productivity management (e.g. using Microsoft Project!) then that is far, far too much and a massive waste of your most important resource -- your time.

One thing you can do to really give yourself a head-start is to estimate how much time it'll take you to do every single task in your project. You'll start out finding that your estimates are stupidly far out, but as your project progresses you'll get better and better at correctly estimating your tasks.

Chapter 3

Doing academic research - Primer

Gathering information to help you complete your project, otherwise known as research, can be one of the most time consuming aspects of the task. There is a pay-off, however, in that reports that are well researched will always be graded higher than reports that are written 'off the top of your head'.

Doing the research for your report can be categorised into two distinct types of activity; primary research and secondary research.

Primary research

Primary research refers to information gathering that you conduct yourself. Primary research can take the form of conducting an experiment, building, testing and evaluating a new application, designing and deploying a usability test, conducting some structured interviews or reporting on direct observations that you make.

Often primary research produces numerical or statistical data that can be presented in the form of graphs, charts or tables - so it would be useful for you to develop skills in the use of spreadsheets and the graphical capabilities of your word processor. It also helps to 'know the basics' about presenting statistics. Whilst it is beyond the scope of this report to provide guidance on collecting, collating and presenting statistical data (note: the school offers a module on working with statistics) there are some basic tips that you should be aware of:

- Don't use percent figures (%) when you are reporting on very small numbers. Instead of writing "20% of the people that responded to the questionnaire believed that global warming is a myth" when only 5 people took part in your questionnaire. It is more honest to report that "one person stated that they did not believe in global warming"
- When you have collected a lot of data (for example questionnaires) it is not necessary to include all of the questionnaire response sheets with your report; this is simply cruel to trees and, if the respondents are identified, possibly in breach of data protection legislation. It is better to summarise the responses in your report (use tables and graphs) and include only one example copy of the questionnaire in your appendix. It is VITAL; however, that you keep your data until the project is marked.
- If you report on an experiment it is essential that you describe the experiment exactly– the 'scientific' rule with experiments is that another researcher should be able to reproduce your results by following your description. This means that your report needs to include details such as the location of the experiment, the time of day, the equipment used and the order that events took place.
- It is important to think clearly about, and report on, any factors that might mean your information might not be fully accurate. Two examples:
    1. A student collects data on the number of cars passing a traffic light every hour during the day in order to produce a graph that will demonstrate the effect of the 'rush hour' on traffic congestion ' in this case some days are more 'typical' than others ' what if the student collected the data during the holiday season?

2. A student uses a stopwatch to time five friends buying a ring-tone from an internet website to judge whether the site is efficient to use. Will the fact that these students are computer literate have any effect on the data? What if the student had timed their parents?

In general, primary research is highly valued by academic tutors. Well-conducted research demonstrates that a student has developed skills in designing research and has put in the effort in order to carry it out.

Secondary research

Secondary research refers to information gathering from sources such as books, journals and web pages. It involves gathering facts and information as well as reporting on the research findings of other people.

Including information from other sources is good academic practice and enhances the believability and trustworthiness of your report. Consider the following two sentences:

1. "almost everybody has access to the internet these days"
2. "UNESCO (www.unesco.org), using statistics gathered from governmental and telecommunication sources, report that whilst internet access in developed countries is as high as 80% - access to the internet in developing countries is much lower. In fact, access to a telephone may be as low as one person in a hundred in parts of Africa and Asia"

Question: Which of the two sentences above do you consider to be more accurate?

The first sentence is an opinion – whereas the second sentence provides evidence. Tutors will expect that any reports you write will provide evidence, in the form of secondary research sources, to support the points that you make.

There are very strict rules about how you report on secondary sources. You must not copy information from other sources without clearly stating where the information came from. Failing to identify your sources (in the citations and references for your report) may result in accusation that you have copied (plagiarised) your work.

Another important issue in the use of secondary sources is triangulation. Triangulation simply refers to the practice of using more than one source of information to provide evidence for your report. Only using one source of information to back up any point you make is tantamount to "believing the first thing you read" and suggests gullibility. Take the following case:

In one assignment students are asked to evaluate whether global warming is a reality. One report reads:

> *"Global warming is a reality. In a report found on the Greenpeace website (www.greenpeace.org), figures for annual carbon dioxide emissions show an 11% increase over the past eight years. Carbon Dioxide is the primary cause of global warming".*

Let's think about this. Greenpeace is a well-respected organisation but it has a particular perspective on environmental matters – other sources should also be consulted. A well-triangulated answer might read:

> *"Greenpeace report an 11% increase in carbon dioxide emissions in the past eight years. Other sources (Kimble 2002, US office of statistics 2000) suggest a figure closer to 6%. The UK department of the environment report that the link between carbon dioxide and 'the greenhouse effect' is a highly debated scientific issue. It is clear that the facts surrounding global warming are controversial…."*

This second answer is better because it acknowledges that the reality of any situation is always complex.

One final point about secondary research sources, whilst the internet is an invaluable source of information there is a major problem with it – we can never be really sure about the trustworthiness of information found there. It is often difficult to tell whether information that seems "accurate" may not have been provided by a commercial organisation with an interest in portraying facts favourable to them. Similarly, because information about authors is often hard to find on a website, we cannot always be sure whether the convincing information we are reading has been written by a 'respected professor' or 13 year old from Arkansas.

This uncertainty and unreliability extends to collaboratively written 'community' websites, such as Wikipedia. They provide a useful starting point for your research but should never be relied upon as your only source of information. For this reason academic tutors highly value research sources that come from published books and journals. Why? Because information included in books and journals has usually been edited and reviewed for accuracy (a system called peer review). The tip to take from this is that you should always refer to a variety of sources of information, not just the internet, in order to convince your marker that your work is accurate.

Academics have a particular perspective on the status of TRUTH

Academics tend to only accept, as ‘fact’, information that has been investigated by more than one person or group of people. They have self-imposed rules about which information to value highly and which to ignore. They try to ignore opinion; they try to ignore information from sources that have a vested interest in ‘painting the facts’ in a certain way, for example, by providing partial information that ‘covers up’ the truth.

One of the VERY WORST types of ‘proof’, and one which academics avoid at all costs, is called anecdotal evidence – that is evidence that ‘plays on’ the human tendency to generalise truth from isolated incidents. For example, evaluate the following statement: “I knew a guy who always played beat ‘em ups on his Xbox and he ended up assaulting some kids at his school, this proves that computer games cause violence”

Another type of poor proof is equating the number of people who believe something with its status as truth. ***The popularity of an idea does not necessarily make it true.***

The three main tools that help an academic to evaluate the status of information are:

1. Research triangulation. This is where information is gathered from more than one source to determine whether those multiple sources are 'agreeing' on the information. BE CAREFUL – a Google search can suggest that there are multiple sources of information, but if you really 'dig deep' you find that all of those sources originally got their information from a single source. (Try this experiment: Google search for 'smallest petrol engine'. Now look at the results from: Esato Archive, Gleez and The Sun. Did you notice that they are identical? Does this make the information more true?)
2. Peer review. This is the system academics use to verify that their community accepts the validity and reliability of their information. Basically, before publishing their findings (in a book or at a conference) academics submit them to a group of other, trusted, academics whose job it is to investigate and criticise the information and findings. If they confirm the results and find nothing wrong with the process used to carry out their investigation the results can be published.
3. Trusted sources. Finally academics value information from a 'trusted source' more highly than other sources of information. The most trusted sources are:

    - Academic journals (e.g. NATURE www.nature.com is the most trusted source in natural science, The Lancet www.thelancet.com is the most trusted source in medicine, the ACM www.acm.org is one of the most trusted sources in Computer Science)
    - Academic conference papers from peer-reviewed conferences.
    - Specialist textbooks (which are themselves peer reviewed)

Sources that academics treat with suspicion include: Wikipedia (a good starting place, but completely unverifiable regarding authorship), some journalism, blogs, information provided by political or pressure groups, opinion pieces, etc.

LESSON: Cutting and pasting unverified information, from untrusted sources on the back of a poorly conducted Google or Wikipedia search is probably the worst thing you can do if you hope to get high marks in a project report.

The difference between fact and opinion.

One of the key problems for academics is determining true information from opinion. Opinions are subjective but very often can be presented 'as if' they are 'fact' through a variety of techniques such as using persuasive language and argument, or visual evidence such as photographs and graphs, that only support the opinion, whilst suppressing or ignoring other evidence. Another technique involves asking high status individuals to support the opinion in order to gain popular support. Sometimes academics support or promote a position which the academic community (their peers) has not yet fully agreed. (If you wish to explore this further examine 'the global warming debate' or 'MMR vaccine linked with autism')

In order to determine whether an academic might be acting outside the academic community's remit you very often have to look for clues, such as; who is funding their research? Are they working for a political or commercial organisation that may have an interest in portraying facts in a certain light?

For examples of these strategies see, for example, Ben Goldacre's excellent (and funny) book Bad Science [fourth estate, 2008].

LESSON: This means that sometimes your secondary research task has to go beyond just finding the information, sometimes you are called upon to determine the value of that information; who is providing it and why? Are they providing it within the context of triangulated, peer reviewed and trusted sources?

What does this all mean for your grades?

Your project will be marked by academics, not business people, journalists or family members. Academics look for certain things in a report that allows them to properly grade your academic capability, central amongst these will be your ability to do high quality academic research and to take a considered view on the status of the evidence. Conducting poor secondary research, and failing to evaluate the validity of your sources, WILL EFFECT YOUR MARKS.

IMPORTANT: In order to get a first class mark (i.e. above 70%) it is a necessary pre-condition that your research be triangulated (comes from more than one source), uses a selection of peer reviewed content (journal articles, textbooks, conference papers) from trusted sources.

This does not mean that you shouldn't use Google. It DOES MEAN that you should use it wisely, and that you should ALSO verify your search's content by other means.

References

- http://home.ched.coventry.ac.uk/caw/harvard/index.htm
- Goldacre, Ben (2008) Bad Science fourth edition. Fourth Estate.
- Graham, Leah and Takis Metaxas, Panagiotis (2003) "Of course it's true; I saw it on the internet!"
- Critical Thinking in the Internet Era in Communications of the ACM Vol. 46, No. 5

Chapter 3 workshop: Doing academic research

Read the document "Of course it's true! I read it on the internet":
http://www.scribd.com/doc/55962191/Of-Course-Its-True-Read-It-on-the-Internet

It will take you about 25 minutes. Think about what the author says about information sources on the Internet - they can be commercial, academic, governmental, personal opinion or the output of pressure groups. Discuss with a friend how you value the academic validity of the information from each of these sources - which do you value most highly? Which the least? Why?

Consider the following statement:

"Playing violent video games makes children more likely to be violent in everyday life"

Now undertake the following exercises:

- First decide what you 'gut reaction' is. Do you agree, disagree, or have no opinion on the statement? For the purposes of this activity you will need to 'set aside' you opinion and adopt the position 'I don't know'.
- Using at least three sources that you believe to be academic in nature (remember peer review, trusted source) and in no more than 500 words, briefly report on the evidence provided from your search. Include correct citations.
- Finally, on the balance of the evidence you have found, make a decision on whether the statement is completely true, possibly true, completely untrue or undecidable. If you cannot decide, state why it is difficult to make a decision.
- At the end of your report you should cite three references in the Harvard format.

For help on citations & references see: http://home.ched.coventry.ac.uk/caw/harvard/

HINTS : This debate started in the 1950's about the effects of television on children. The academic discipline that has probably the most research on the question above is SOCIAL PSYCHOLOGY, however you will also find the question examined in disciplines such as sociology, cultural studies, and computer game studies. Useful search terms (yes Google!):

- Media effects debate / Media violence debate
- Albert Bandura / Bobo the clown experiment
- Folk devils
- Video game behavioural effects
- Game Brain
- Media violence research
- The Cultural Policy Challenges of Video Games

Useful Journals and Conferences: Psychological Science Journal, Journal of Personality and Social Psychology, Digital Games Research Association.

In your argument, be mindful of, and comment on, the source of your information (is it opinion? is it statistical? Who funded the research? Who funded the website where you found the information?). Using only one source of information is tantamount to "believing the first thing you read" – so avoid doing this! If, at the end of this exercise, you are wondering how such a question could ever be fully decidable, then you are thinking in the right direction.

The reason it is so difficult is because there are 'multiple variables' that do not reduce to simple causal chains. Defining the terms of the question gets us a bit closer. For example, ***"Playing violent video games makes children more likely to be violent in everyday life"***

Consider:

- What is play?
- How do you define children?
- How do you define violence?
- How do you define 'everyday life'?
- What video games? Tetris? Farmville?
- What does 'more likely' mean - 'more likely' than what?
- How do you define 'makes' - is it 'forces children'? or is it 'pre-disposes children'?
- Does the question consider other factors such as individual personality or social determinants?
- What about the children's perspective on their actions - can they tell the difference between 'play acting violence' and 'real violence'?
- Should we judge children's actions (social psychology) or changes in their brain (clinical/biological psychology or Neuroscience) as more relevant for answering this question?
- What if we observe 'changes in the brain' but observe 'no outward signs' of a change in behaviour?
- Consider how biologists, neuroscientists, social psychologists, sociologists, historians, educators, journalists approach this question. The evidence that they privilege will almost certainly be different. Is it even possible for sociologists (who look at group behaviour) and neuroscientists (who look at brain scans) to ever agree the definitions of the question?

Hopefully you can see from this why designing primary research experiments on human thoughts and behaviour is so very difficult.

Chapter 4

How to write a literature review for your final year project

Every year I supervise projects I find students tend to ask the same questions about their literature review. Most common are:

- How many papers should I read?
- How long should the literature review be?
- Should I read books, articles, or ...?
- Is it OK to reference websites such as Wikipedia?
- Who will read my literature review and what can I assume about their knowledge of the area?
- When should I start the literature review and when should it be finished?

These questions crop up frequently and will be familiar to any readers who are starting their own project. However, when you fully understand the purpose of the literature and how to go about writing one, you begin to realise that these questions are actually not that important. This chapter is designed to help students make that transition, from not yet understanding what the literature review is for, to having a thorough understanding of its purpose and a clear idea of how to write it up.

The meaning of a final year project

A lot of students starting off their projects talk about writing a "report" at the end of their "project" and doing some "research" as part of their literature review. This is where the subtleties of the English language tend to cause a lot of confusion. "Research" can have many different meanings, and to complete a really successful final year project the first thing to do it to understand fully what is expected of you in an academic context. Even if your final year at University is the last experience you have of academic work, remember that your work will be marked according to academic criteria, and academia has quite different aims to industry.

So, to be clear: **research**, in an academic context, means adding something new to the body of knowledge that humans have gathered in your area of interest. Your project as a whole will be a piece of research because you will be creating something new that has not been created before. What that is, exactly, will depend on the field you are studying. It might be a new perspective on a piece of literature, a new proof of a theorem, a new application of a particular technology, or something else. Since you are still an undergraduate it is likely (although not necessary) that your work will be a small step forward. It is unlikely that you will produce something completely ground breaking, so don't be intimidated by fact that your work has to be novel. That said, it may be that you produce an excellent piece of work and your supervisor may want to turn that into a technical report or conference paper with you, which would be great for your CV (or resume).

A **thesis** is a statement of belief that is central to your research. Your **report** will be a piece of writing that defends your thesis, based on your research. So, for example, if your thesis is: *regular online tests help University students to learn new material* then you will need to implement some sort of online tests for new material, design and run an experiment to test your thesis, and write it up in your dissertation. Equally, if your thesis is *water causes cancer in mice* then you will need to plan and run an experiment to determine whether or not this is true and write it up. Notice that you may disprove your thesis in your work. It may be that online tests do not help students learn, or that water doesn't cause cancer in mice. This is absolutely fine, so long as your experiments give a clear answer to the question and you can show that your experiments were performed fairly it doesn't matter whether your thesis turns out to be incorrect or correct (in so far as you have tested it). It may also be that your evaluation is inconclusive, which is also acceptable, so long as your experimental method is good and you can say exactly what further work is necessary to produce a definite result, you will be fine.

Alternatively, you might phrase your thesis as a **research question**. In which case, instead of having a thesis such as water causes cancer in mice you would ask the question *does water cause cancer in mice* and your dissertation would describe your efforts to answer that question.

The shape of your dissertation and where the literature fits in

Every dissertation is slightly different, but good dissertations will all contain the same elements. I should say that the advice given in this section is likely only relevant to science based projects. If you are working in the arts or some areas in the humanities then the expectations of you may well be very different. Still, a good dissertation in the sciences will contain roughly the elements listed below. I say "roughly" because, depending on the exact nature of your work, it may be sensible to expand some sections into two chapters rather than one, or to coalesce some elements into a single chapter. Your supervisor can give you more specific advice on this.

- **Introduction:** should introduce the reader to the broad context of the research and explain why this is an interesting area to work in. So, if your thesis is something to do with mobile computing, you might say something here about why mobile phones are important, why mobile computing is an interesting and important area, and broadly what other researchers are working on. At the end of the chapter you will want to introduce your specific research question, having said why the area you are working in (and therefore your question) is important.
- **Literature review:** Now you have introduced the reader (who will likely not be an expert in your exact area) to the broad research agenda in the field, and your research question, you can start writing more specifically about your own project. In this chapter you will survey the work that other researchers have done to answer your research question, or related questions. At the end of the chapter you should briefly explain how your own work builds on and differs from the work that has gone before it.
- **Method:** this chapter should describe what you did to answer your research question (or to support your thesis, if you think of it that way), and how you went

about it. You should describe your work in sufficient detail that another researcher could recreate your work to check your results.

- **Evaluation:** here, you should evaluate what you have done, and say what answer (to your research question) you have arrived at. It may be that in your method you describe some experiments, and this section records your results and analysis of those results. This is an important section -- most students gain or lose marks in either their literature review or evaluation. Key to producing a convincing evaluation is to plan very early in the project what information you will need to write this section. More on that in another blog post.
- **Conclusions:** should summarise what you have done and how you answered the research question. It may be that your work produced a very clear answer to the question, or it may be that your work points to a need for further research to clarify or confirm your answer. You should refer back to the literature review and summarise how your research differs from (hopefully improves on) the work described in the literature. Make sure you also say what research you would do if you were to continue working on your project.
- **References:** a list of publications cited in the main text, in Harvard style or similar format.

It is likely that most chapters will be roughly the same size, although the introductory chapter and conclusions are usually slightly shorter than the others. Try to let the lengths of each chapter be guided by the amount of useful and important information you have to convey to the reader, don't impose artificial word limits on yourself.

Summaries and synthesis: what should go in your literature review?

Poor literature reviews often take the same form -- they tend to be a (usually short) list of papers that the student has read, briefly summarised. This is not really what is expected and will not gain high marks. Another common mistake is to review literature that has been used to inform some part of the practical work of the project, rather than to review work that has answered the same or related research questions. To do better, your writing needs to not only *summarise* the prior art in your area, but also *synthesise* what is in the literature. In fact, synthesis is one of the key skills that we expect to see from final year students.

So, what is synthesis? The main idea is that you should have understood the literature you have read and, more importantly, you should show that you understand the relationships between items of literature. That means what came first in your field, how it influenced later work, how each step forward in the research improved upon what came before it, and so on. Ideally, you will present your own view of the work you are describing. This partly means that you should be critical of the literature you read, and say where the shortcomings of the work are, and how the work could be improved upon (in particular how *your* work will improve on the prior art). Also, you might have your own view on where your area of research is likely to go in the future.

Critiquing the work of others is something that is often new to students in their final year. One of the most frequent mistakes I see from students is to criticise the style of the papers they read, rather than the research that those papers describe. Avoid writing things like "this paper is not well written" or "this paper is hard to understand". A literature review should really be a review of the research work that has gone before you, not a literary criticism of the style that other authors adopt.

Practical matters: how to start, how to finish and how to do the bit in the middle

Reading and understanding the work of others is a lifetime's work for professional researchers, it is not something that starts and stops on particular dates, according to a Gantt chart. Your final year project will have a hand in deadline, so you need to be a little more circumscribed about how work.

Ideally, you should be reading some literature in your field very early on in your project, to help you choose a good topic and write an initial research proposal. I would suggest that you consider this to be the starting point for your literature review and keep on reading and adding to your writing all the way through your project until you hand in your final dissertation. To do this, I suggest you do two things. Firstly, keep a careful log of what you read in a format you find easy to work with. This might be a log book on paper, or it might be a file on your computer, whatever works best for you. Each entry should include the title of the work you have read and enough information about the authors and so on that you can find the publication again. You should then summarise the work in the paper, including the research question answered by the work, the nature of the answer and the methodology of the research (i.e. what the authors actually did). This will give you a couple of advantages -- you won't forget anything you read because you have a record of it, your literature review can be written from your notes and if you are asked any awkward questions in a viva you can refer back to your log. Secondly, as soon as you feel you have read enough to understand the broad context of the literature, start writing it up formally as your second dissertation chapter. This should really be within two or three months of your starting date. Then, as the project progresses and you read more, you can integrate your new reading into your already drafted chapter. This might seem like a lot of effort early on, but when you come to write up your work, you will be incredibly grateful that your most time consuming chapter has already been written, when it was fresh in your mind, leaving you free to write up the later sections of your work.

An example of good writing

Now you know what a literature review is for, how it fits into your dissertation and how to go about writing your own, it would probably be useful to see an example of (part of) an example review. The paragraphs below give such an example and the text [in italics] is some commentary to explain how each part of the writing contributes towards the start of a good thesis chapter. When you read this, don't worry too much about the subject matter; just try to concentrate on the style of writing and the structure of the text.

The area of pervasive, or ubiquitous, computing was founded by Wieser (1991) *[referenced]* who predicted that computers would one day be integrated into everyday objects and interact with people seamlessly. Although few such products are available today Weiser's work has led to the creation of a number of research areas, including ambient intelligence (Eli and Epstein 1998), smart dust (Khan et al, 1999) and the Internet of Things (Brickley et al, 2001). *[Sets the historical context of the area and defines related areas.]*

An early application of pervasive computing was the active badge location system, described by Want et al (1992), in which users and objects were tagged with an "active" badge which could locate and identify them. This system was based on ultrasound locationing, whereas later systems might use RFID technology to achieve the same effect. *[Describes how the field has changed over time]* Uses of the active badge system included routing phone calls, email alerts and so on to the physical location of the receiver. *[Contextualises the fundamental research]*

Chapter 5

Why read from primary sources?

Or: why reading blog posts is harder, not easier than reading papers

I've been meaning to write this for a long, long time. Now that I have an enormous pile of marking to get through in double-quick time, I have the perfect excuse for a bit of structured procrastination.

What is a primary source?

A primary source is an original piece of writing describing some research, written by the person or team who performed that research. A secondary source, is a description or discussion of a piece of research by someone who has read about the research, but did not carry it out themselves. If an academic performs an experiment and writes it up as a journal paper, that paper is a primary source. If another researcher then quotes the paper and cites it in one of their papers, then that is a secondary source. Newspaper articles, magazine articles, entries on Wikipedia, and most websites and blog pages are secondary sources. When it comes to scientific research, only writing published in peer-reviewed conferences, journals, books and magazines constitute a primary source.

What is peer-review and why does it matter?

Even if a paper is a primary source describing some research, that doesn't guarantee that the research is rigorous, reliable and high-quality. To ensure that all academic writing meets basic standards of quality assurance, scientists use a system of peer-review. This means that a number of professional scientists (usually two or more) will read through the work carefully, and critique it before it is published. If the work is of very poor quality, or very badly written, it will be rejected and the author(s) will have to re-write their paper and try to publish it elsewhere. If the work is of a high enough standard to publish, the authors will be given a list of improvements they must make before the paper goes to print. This way, the academic community itself tries to ensure that inaccurate, incorrect, or incomprehensible work doesn't get published in high quality conferences and journals.

Why read primary sources?

Students often complain about making the leap from reading textbook-style prose to formal, academic research literature. Part of the problem is that the style of writing is different, and takes some getting used to. More deeply, though, students today have likely grown up with the web and with reading informal, secondary sources, making the change is hard work, and nerve-wrecking for some. Why waste hours wading through pages and pages of long-winded, complicated, weirdly-written prose, when you can read a quick, accessible summary on Wikipedia? Well, of course Wikipedia is a good place to start to get a basic overview of an area and help your understanding of the primary sources you are reading. However, it is absolutely essential to read the primary sources themselves. Why?

Reason 1: secondary sources editorialise

A secondary source will describe some parts of the primary source, but not others. The secondary source will take a particular point of view (i.e. the author will voice their own opinion) and will pick

the parts of the primary source that are useful for that discussion. This doesn't necessarily mean that the secondary source is particularly biased (although it might be); it's more that secondary sources are selective in what they discuss. For example, if a paper on Web2.0 discusses the implementation, performance and usability of Web2.0 sites, a secondary source on the subject of usability is likely to leave out any mention of implementation and performance. So, by reading secondary sources you miss out on a lot of the detail of the original work and much of that detail may be very important to you and your work.

It is probably worth saying that there is an important exception to this: survey papers. A good survey paper should be like an extended literature review that discusses, in some detail, the literature available in a broad area of Computer Science. These survey papers are a good place to start when writing your own literature review. You can usually find survey papers in well established journals, or specialist survey journals such as ACM Computing Surveys.

Reason 2: secondary sources are sometimes wrong

Every academic field has a number of ideas which are passed on from one generation to the next with little reference back to the original research that generated those ideas. Be somewhat sceptical about this, most of the time there are good reasons to feel assured that this knowledge is sound, especially in fields where mathematical proof is the main way of advancing the field. However, in more subjective or experimental fields (such as Software Engineering or Usability) results can sometimes be misunderstood or misinterpreted over the years.

An example of this is Winston Royce's "Waterfall Method" which (as you probably already know) is a method for organising and planning large programming projects. The central idea in Royce (1970) is very simple and easy to understand: you split the work into a number of different "phases" (requirements gathering, analysis, design, coding, testing, maintenance) and your team performs each phase in turn. There's even a nice image to go with the idea, just to make it nice and easy to understand:

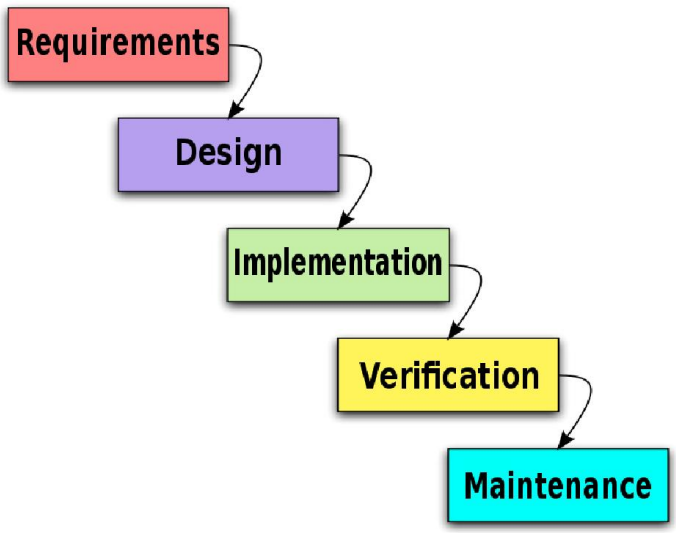


Image source: Wikipedia

For many people, this is where their understanding of the waterfall model stops. But in Royce's original paper there is a long discussion of the drawbacks of organising a project in this manner In fact Royce says that it is "risky and invites failure" (pp. 329). Moving on, the majority of Royce's

paper is a list of changes to the sequential model which make it more workable. Some of these are of particular interest, for example "plan testing" is a step that Royce advocates should go with program design. In modern, more "agile" development methods we would advocate writing unit tests around this time, so Royce is presenting a very modern approach. The last modification Royce makes is to "involve the customer" at several points in the process. Again, a much more modern approach that many authors would say goes with agile or "eXtreme" development methods.

The picture Royce paints is not a simple sequential model at all; it's much more complicated than that. Tarmo Toikkanen has written an interesting blog post on this subject. He speculates that the reason people advocate for the basic waterfall method is that the diagram and analogy make it very easy to understand, so people don't delve any deeper into the details. In fact, Toikkanen points out that NATO even have a military standard (DOD-STD-2167) based on Royce's work. [Aside: If you wanted to test Toikkanen's idea that it's the diagram in Royce's paper that leads to the misunderstanding, what experiment would you devise to test that idea?]

More parochially, we often see University students writing something like "in my project I will use the Waterfall Method" sometimes even with a citation. DON'T DO THIS! Read Royce (1970) in full, understand what he's really arguing for, then use a more modern method, or at least use Royce's iterative method found at the end of the paper.

In a similar way, many examples of original research have become unquestioned ‘folklore’ long after the relevance of the original research has been overtaken by more recent research findings. One example of this is ‘san-serif fonts are easier to read on screen, serif fonts are easier to read in books’ another is the incorrect application of George Millers 1956 work on ‘working memory’ (the famous 7 +/- items) to completely inappropriate domains of use.

Reason 3: different primary sources may disagree

Research is all about creating and discovering new ideas. Very often primary sources disagree on how best to do that, or they have competing ideas and only through years of research and discussion does a consensus evolve. There are examples of this throughout the history of science. Whether it's the flat Earth debate, Big Bang vs. the Steady State theory, structured programming vs. object oriented programming, through debate, reason, mathematical arguments, prototype systems, models, simulation and all sorts of other techniques, the history of science is full of arguments and competing ideas.

When you read a secondary source, very often whatever "debate" has taken place is already in the past and the author of the secondary source will simply describe the consensus that has since been reached. For example, there were many good reasons for cosmologists to believe in the steady state theory before evidence for the Big Bang became overwhelming. Only by going back to this literature can we see how the debate unfolded and why the evidence that supported the Big Bang (to do with background microwave radiation in the cosmos which was discovered in the 1960s) was so convincing.

In Computer Science there are also many of these debates. For example, most programming languages do not have a "goto" statement. In fact, Java has a keyword called "goto", but it is not used. In the late 1960s and 70s there was a heated debate about whether "goto" was a safe and useful construct and you can read through that debate in Dijkstra (1968), Knuth (1974) and plenty of other sources. Without going back to these papers, which were written well before the debate was settled, can you fully understand the arguments that, eventually, banished the "goto" statement from most modern languages.

Conclusions: reading blog posts is harder than reading papers

So, why did I say in the title of this chapter that reading blog posts is "harder" than reading papers? Actually reading blog posts may be easier, but in terms of getting a good grade in your project you are unlikely to produce a high quality literature review based on blog posts. Blogs will tend to be selective and biased in their nature. This isn't a criticism of blogs, far from it; blogs are a great place for lively debates. They aren't such a great place, necessarily, to describe careful, peer-reviewed research in great detail - that's best left for conferences and journals.

References

Edsger W. Dijkstra (1968) Letters to the editor: Goto statement considered harmful. Communications of the ACM 11, 3 (March 1968), 147-148. DOI=10.1145/362929.362947 http://doi.acm.org/10.1145/362929.362947

Donald E. Knuth (1974). Structured Programming with goto Statements. ACM Computing Surveys 6, 4 (December 1974), 261-301. DOI=10.1145/356635.356640 http://doi.acm.org/10.1145/356635.356640

Royce, W. Winston (1970), Managing the development of large software systems: concepts and techniques In proceedings of IEEE WESTCON, Los Angeles , 1--9 .

Toikkanen, T. (2005) Don’t draw diagrams of wrong practices or: why people still believe in the waterfall model. http://tarmo.fi/blog/2005/09/dont-draw-diagrams-of-wrong-practices-or-why-people-still-believe-in-the-waterfall-model/

Workshop Chapter 6: Reading from primary sources

The focus of this workshop is on going back to primary sources to fully understand the "wisdom" that is often passed down to students as "common sense". Some of this "common sense" represents very useful, hard won experience. On the other hand, some comes from a misunderstanding of the primary sources, and some is just plain wrong. It's only by going back to read the original research that has been done that we can figure out which is which.

Case study 1: Royce's waterfall method

Read through Royce's original paper on the Waterfall method. If you don't have access to it on the ACM library, you can find a copy here. Write a short essay (maximum length one page) explaining what Royce actually thought of the Waterfall Method.

Case study 2: Working memory and powerpoint slides

Miller (1956) is often used by people who teach lecturers how to improve their lecturing. The usual context in which Miller's work is presented is to do with the presentation of information to students and novice lecturers are usually told "Never put more than seven bullet points on a powerpoint slide".

Imagine you are working on a final year project where the research question is something like "How can complex academic concepts be best presented to web site users". Clearly, Miller's work is going to be important in your literature review, and it might even point the way to a good answer to your research question.

Find a copy of Miller (1956) and read it through. Write a couple of paragraphs, as if you were writing up your literature review, describing Miller's findings. In particular, you should comment on whether or not Miller's paper really does support the idea that you should not put "more than seven bullet points" on a slide or webpage.

Case study 3: How much more productive are "good" programmers, compared to "average" programmers?

Recently, in the blogosphere, Steve McConnell and Laurent Bossavit have had an argument over the origins and accuracy of the commonly held view that "good" programmers are ten times more productive than "average" programmers. Read both Steve McConnell's side of the story, here, and Laurent's post (translated into English) here. Write a research proposal (with the same structure as your final year project proposal) that proposes a programme of research to answer the question "are good programmers ten times more effective than average programmers?". Think carefully about the literature you will need to review, as well as the original experiments you will need to carry out.

Case study 4: Goto and structured programming

In the early days of programming languages Donald Knuth and Edsgar Dijkstra, two of the "big names" in Computer Science who made many early advances in the field, had a major disagreement about the "goto" statement that can be found in the C programming language and others. "goto", if you haven't come across it already, can be used to "jump" to any other place (sometimes with restrictions) within a program. For example:

```
...
label: foobar
```

```
...
if (SOME_TEST) { goto foobar } // Jumps to label
...
```

You may well have heard about this discussion, or have seen papers with titles like “SOME IDEA considered harmful” which refer back to Dijkstra’s letter from the 1960s. Read through Knuth (1974) and Diskstra (1968) and write a short summary of the arguments for and against using the “goto” statement, based on the primary sources.

References

Dijkstra, E. W. (1968) Letters to the editor: Goto statement considered harmful. Communications of the ACM 11, 3 (March 1968), 147-148. DOI=10.1145/362929.362947 http://doi.acm.org/10.1145/362929.362947

Knuth, D. E. (1974). Structured Programming with goto Statements. ACM Computing Surveys 6, 4 (December 1974), 261-301. DOI=10.1145/356635.356640 http://doi.acm.org/10.1145/356635.356640

Miller, G. A. (1956), 'The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information', *The Psychological Review* 63 (2) , 81-97.

Royce, W. W. (1970), Managing the development of large software systems: concepts and techniques In p*roceedings of IEEE WESTCON, Los Angeles* , 1--9 .

Chapter 6

How to read research papers

Photo by ailatan on Flickr

Whilst chapter 4 is about how to write literature reviews and chapter 5 is about why you should read academic literature in Computer Science. This chapter is about how to read research papers.

We often find that if students haven't done much of this sort of reading before they get to their final year, then getting started can be a bit of a shock. So, this chapter is designed to help you get started with reading academic literature and, just as importantly, to help you get the most out of the papers you do read in the short space of time you have available in your final year.

Remember the structure of a paper is just like the structure of your thesis.

Other chapters in this book have discussed the overall structure of your thesis, but in outline this is the sort of structure you should be expecting to produce:

Introduction should introduce the reader to your research question and the broad context of the research.

Literature review should describe the work that other people have carried out to answer your (or similar) research questions.

Method should describe what you did to answer your research question (or to support your thesis, if you think of it that way), and how you went about it.

Results should evaluate what you have done, and say what answer (to your research question) you have arrived at.

Conclusions should summarise what you have done and how you answered the research question.

Academic writing of all sorts follows something like this structure, including all of the papers that you will be reading for your project. There are a couple of exceptions to this rule. One is theoretical papers which sometimes put their "related work" (or literature review) somewhere towards the end of the paper rather than after the introduction. The second exception is survey papers. Surveys are extended literature reviews and, as such, are a good place to start in your own literature reviews. ACM Computing Surveys is a journal that publishes survey papers or you can sometimes find them in reputable journals.

Briefly review each paper for relevance

You don't have time to read everything, so it's important to make sure that what you do read is really relevant to your thesis. So, to check whether a paper is likely to be relevant to you first read the Abstract. This should give you a brief summary of the whole paper. So, at the very least the abstract should give you a good idea of what research question the authors were trying to answer. Next, read the Conclusions. This is also likely to be a summary and may well give you a better idea of what results the authors obtained and what work they did not finish but left for "future work". If that doesn't give you a good enough idea of the relevance of the paper to your own work, try reading the last part of the Introduction. This is usually where the authors summarise what is written in each of the following sections of the paper, so that should give you a much more detailed view of what the rest of the paper contains.

If, after all of that, you think the paper is irrelevant to you, then discard it and move on to something else. Otherwise, you are ready to move on with your reading...

Focus your reading on specific questions

If you just go ahead and read a paper from start to finish the chances are that you won't get very much out of the exercise. You are likely to ramble around the paper, not taking very detailed notes and, at the end of the paper, you may not have learned much.

A much better way to go about your reading is to keep in mind a number of clear, focussed questions and read the paper with the intention of writing down answers to these questions in your notes. That way you will finish with a clear set of notes that you can be confident will be useful to you when you start writing up.

I would recommend you use this set of questions to guide your reading:

- What research question were the authors asking?
- Why did the authors believe that their research question was important?
- How did the authors go about answering their research question?
- What results did the authors obtain or, what did the authors learn from answering their research question?

Making use of your notes

When you have finished reading you should have a stack of notes on all the papers you have read. This should be a much more concise way to start writing up than having a much bigger stack of papers and (most likely) not much memory of what was in them! So, the next thing to do is decide on the structure of your literature review chapter.

The first paragraph of your chapter should introduce the rest of the chapter. This is a good place to remind the reader of your research question and explain how the current chapter relates to it.

The last paragraph of your chapter should summarise what you have reviewed. This is a good chance to help the reader navigate around your thesis. Briefly review what you have said in the chapter and refer the reader to the next chapter, explaining how the next chapter follows on from the current one.

The middle part of the chapter is more difficult and, since your writing will depend on your particular research question and the literature you have read, there isn't much generic advice to be given here. However, you can start by reading through your notes and looking for common themes. Think about how best to present the ideas to a reader who has not read the same literature. Do you want to take the reader chronologically through the literature, from the earliest point to the present day? Would it be easier to understand if you split the reading into particular topics that are related? When you have what you think is a good structure, write some section headings into your thesis and think about which papers go in which sections (of course, some papers may well go into several sections).

Write the citations into each section using something like EndNote, Mendeley or BibTeX to format them for you. Play around with the structure until you are convinced that it will make sense, then write in the details of each section.

Chapter 6 Workshop: How to read peer-reviewed papers

This workshop is best done in a group using papers that you and your colleagues are really going to use for your literature reviews. However, if you don't have a group to meet with or you haven't got far enough with your work to have a paper to read, then use a paper from the references. These papers have been chosen to be readable by final year students, but don't worry too much if you don't understand all of the technical details.

Workshop

In your group, each of you should bring in a printed copy of one paper that you have found interesting and you expect to include in your literature review. Swap your papers around, so that you aren't reading your own paper and, ideally, you aren't reading a paper in your particular area of interest. Spend about twenty minutes on directed reading and note taking, then another twenty minutes discussing your notes. Of course, you should not just read the paper straight through, you should read the abstract and conclusions first and then read as much of the content of the paper as you need to answer the following questions:

1. What research question were the authors asking?
2. Why did the authors believe that their research question was important?
3. How did the authors go about answering their research question?
4. What results did the authors obtain or, what did the authors learn from answering their research question?

We have developed a template for you to use for this exercise which you can find here.

References

Guinard, D.; Fischer, M.; Trifa, V. (2010) Sharing Using Social Networks in a Composable Web of Things. Proc. of the First IEEE International Workshop on the Web of Things (WOT2010). Mannheim, Germany, March 2010

Langendoen, K., Baggio, A. & Visser, O., 2006. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture.Proceedings 20th IEEE International Parallel Distributed Processing Symposium, p.1-8.

Madhavapeddy, A.; Scott, D. & Sharp, R. (2003), Context-Aware Computing with Sound., in The Fifth International Conference on Ubiquitous Computing. Anind K. Dey; Albrecht Schmidt & Joseph F. McCarthy, ed., 'Ubicomp' , Springer, , pp. 315-332 .

Schulz, R.; Glover, A. J.; Milford, M.; Wyeth, G. & Wiles, J. (2011), Lingodroids: Studies in spatial cognition and language. ICRA 2011, The International Conference on Robotics and Automation, Shanghai, China, May 2011

Chapter 7

How to evaluate your project

Image by mpeterke on Flickr

This chapter is all about how to evaluate your project. This is something you will do ideally about half way through your work, but more realistically towards the end of your work, just before you write up. However, it's important that you plan your evaluation early on; otherwise you run the risk of getting almost to the end of your work and finding that your evaluation process isn't going to give you any good results. So, the main thing you need to understand when you are planning is how evaluation works in the sciences and how to apply that understanding to your own projects.

Proof

Sometimes we see student dissertations which make claims like "this experiment has proved that ..." Almost always the student has used the word "proof" incorrectly, and will lose marks because they have given the impression that they have not understood the contribution that their work has made to the field they are working in. Very often students in this situation simply haven't understood how scientists use the word "proof".

Proof, in a scientific context, is a mathematical argument that is used to convince other mathematicians or scientists that a theorem (or a mathematical idea) is true. Proofs must never involve evidence or experiments, only arguments. There's an example proof of a simple theorem at the end of this post.

Once mathematicians are convinced that a proof is correct (and sometimes that is difficult in itself, if the proof is several hundred pages long) then it is irrefutable. This is very different to the sort of science that is advanced by experiments, where another scientist can find new data or evidence that shows that an old idea was wrong.

So, we generally say that a theorem can be proved correct whereas a hypothesis (or guess!) can only be tested via experiments. A hypothesis might turn out to be wrong if experimental data cannot be found to support the hypothesis, or contradictory evidence is found. If a lot of evidence is found to support a hypothesis we might call it a theory. Even so, a theory cannot be proved correct in all cases. For example, if you came up with a theory that said all atoms have a particular shape, you might invent a special microscope to look at atoms and find out if they have your shape. This would provide some evidence to support your theory. You couldn't, however, test every single atom in the Universe, so your hypothesis might well become a theory, but it can never be "proved" correct.

[Aside: there's a long literature in this sort of philosophy of science. If you are really interested, read Karl Popper on Falsification, AJ Ayer on Verification and Paul Feyerabend on scientific revolutions and Imre Lakatos on Proof.]

Scientific method

Scientific method is the way that scientists decide whether a particular hypothesis (or guess) is likely to be a good model for the way the world works. If most scientists accept that the hypothesis is likely to be true, then we call it a theory. Of course, even theories have limitations, and it may be that as more experiments are carried out we find that a different theory fits the evidence better, or that the theory only works in certain circumstances. This is exactly what happened in physics to Newton's laws of motion. It turns out that Newton's laws describe the world pretty well in most cases; they can certainly tell you when your train is likely to arrive at its destination. For other circumstances, for example when you are travelling very fast, close to the speed of light, or for very small particles like quarks, other theories (like Einstein's theories or quantum mechanics) better fit the data we have gathered. Of course, much of this work in areas like physics is driven by what we can measure and observe. Better telescopes mean better theories of cosmology, and so on.

In computer science we also have hypotheses that we can test. For example "functional programming languages can run just as efficiently as imperative languages", "online learning increases student engagement", "objects and inheritance improve code reuse in software companies", and so on.

To be a true hypothesis, and not just the opinion of the author, a statement must be refutable, that is, it must be possible for experiments to determine that the hypothesis is incorrect. The opposite statement to a hypothesis is called an alternate hypothesis. Examples for the hypotheses listed above would be "functional languages are necessarily slower (or faster!) than imperative ones", "online learning has no effect on student engagement" and "objects and inheritance have no effect on code reuse in software companies".

So, to evaluate your own research questions, you need to do the following:

- Devise a hypothesis.
- Form your alternative hypothesis.
- Plan an experiment that tests whether the hypothesis or the alternative hypothesis is true.
- Conduct your experiment.
- Analyse the results of your experiments.
- If the results are conclusive, STOP. Else, re-run the experiments, or devise a better experiment and repeat.

In a student project, you may not have time to repeat your experiments, especially if they involve people, but you should design your evaluation in such a way that this would be possible, were you to continue the work.

About experiments

A good experiment should test one variable and one variable only. So, if your hypothesis is "neural network algorithms run faster in C than C++" then you will probably want to implement some neural network algorithms in both languages. You should make sure that the programs are as similar as possible, except for the language you are using. If you implement slightly different algorithms, it may be the algorithm and not the language which is causing any change in performance you observe. In this case, the programming language is called the independent variable and the algorithms are called the controlled variable and the speed is the dependent variable which is being measured.

Bad Science? The case of usability tests.

Many students undertaking projects who have developed a web application, web site or content management system (often in response to a client brief) ask about the most suitable evaluation method for their project. In these cases precisely *what* to evaluate is often less clear cut than an experiment with an independent variable and a controlled variable (as in the "neural network algorithms run faster in C than C++" example above).

In order to settle on an evaluation method for such cases it is often necessary to return to the client and question them about their goals for the application you have built for them. The first question you need to ask is “Can my application be evaluated by automatic means?” i.e. for applications that are evaluated for *technical* performance (network performance, speed of execution, resource efficiency, memory performance, robustness against attack, etc) the answer is usually ‘yes it can’ and your evaluation may consist of running a number of automated performance tests and collating the results into comparison tables.

However, in cases where the client might be interested how humans (users) interact with the application and ‘perform better’ because of it, the evaluation solution is usually some form of ‘user based test’. Your client may be interested in the performance of the application you have built in supporting users to achieve their goals. There are a number of parameters that could be tested:

- **Effectiveness:** Can users actually perform and complete the (desired) specified task?
- **Efficiency:** Can users do it quickly, without getting bored or frustrated?
- **Satisfaction:** Is it fun, or at least pleasant to use?
- **Learnability:** Can users 'pick up' the application without reaching for a manual or asking for help? Does it support learning? (see ISO 9241 section 11)

Different applications will have different emphases in terms of what you need to evaluate. Games, for example, need to be satisfying or challenging most of all. Terminal applications for call centres need to be efficient most of all. Most kinds of ‘stand-alone’ technology (Car park ticket machines, vending machines, ATM machines) need to be learnable most of all. All applications need to be effective.

At this point it is important to stress that working with real users introduces a large number of uncontrolled variables that cannot easily be ‘designed out’ of any user test you may undertake. The fundamental question about the ‘scientific validity’ of usability tests (i.e. whether you are ‘really’

evaluating user performance rather than the application's performance in the test) is very difficult (but not impossible) to answer.

There are two specific limitations with usability test data.

The reliability limitation: *Is the data reliable?*

> No, if you are testing users who are not typical of the true intended user group, or if there are significant individual variations within the test group (this is made worse by small sample sizes*)

The validity limitation: *Are the conditions under which the data was recorded reproducible?*

> No, if the test is run differently each time (users briefed differently, different equipment used, not quite the same test questions asked)

As Jeffrey Rubin points out [usability] "testing is always artificial" (Rubin, 1994, p.27). [*Aside: There is a fascinating and long running debate on the scientific status of usability between Jakob Nielsen and Rolf Molich, particularly in relation to the small sample sizes championed by Nielsen. See here or here]

There is some good news about usability testing, however. Often it is entirely unnecessary to worry about the number of uncontrolled variables in a usability test because you are looking for *indicators* rather than proofs. In fact, if you think of a usability test as a 'design tool' rather than an 'experimental tool' you are closer to the way tests are used in the commercial world. This does not absolve you of the responsibility for designing and recording a test in which you have addressed the reliability and validity limitations OR that you have set appropriate benchmark metrics for judging the success (or not) of the application in supporting effectiveness, efficiency, satisfaction or Learnability. It does mean, however, that usability tests can, if designed properly, be a very good evaluation method for your project.

Mind your language: 'user friendly' and 'significant'.

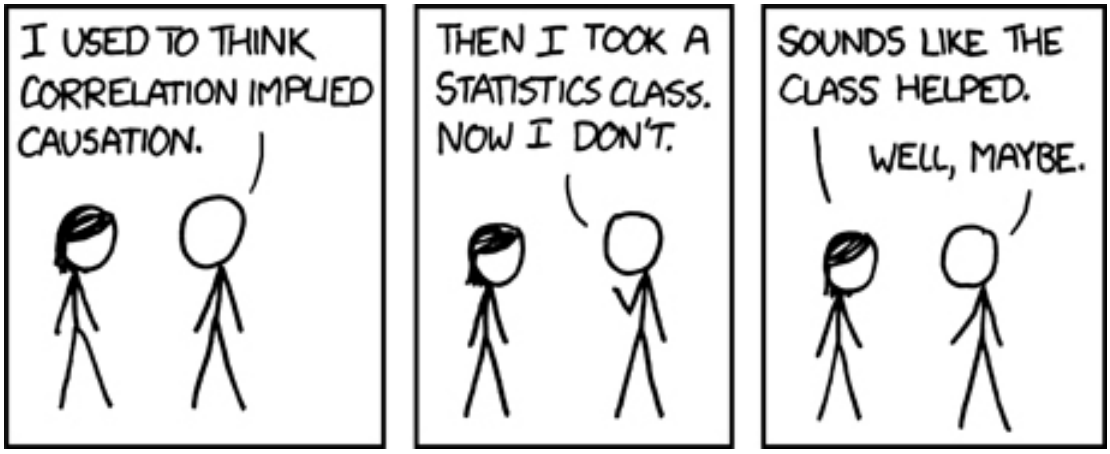
There are two phrases you need to be very careful about using in your test reports.

1. ***User-friendly.*** Never use this phrase! Software cannot be friendly; it is not (yet) sentient. This is called ANTHROPOMORPHISM and should be avoided. To claim that an interface is 'user friendly' is also subjective and not testable. To claim, however, that an interface is USABLE is more sustainable as long as we measure against some predetermined metrics.

2. ***Significant.*** In normal English, "significant" means 'important', while in Statistics "significant" means probably true (i.e. not due to chance). A research finding may be true without being important. When statisticians say a result is "highly significant" they mean it is very probably true. They do not (necessarily) mean it is highly important. Be careful when you claim to have found 'significant' results.

   (http://www.surveysystem.com/signif.htm)

Interpreting your results: *correlation does not imply causation*



Correlation by xkcd

When you perform an experiment, you are hoping that the outcome will lend some evidence to either your hypothesis or your alternate hypothesis. Going back to the example above, they hypothesis "neural network algorithms run faster in C than C++" has an alternate hypothesis "neural network algorithms run no faster in C than in C++". If we run an experiment to test this, and assume it's a fair experiment, and the results are that all our algorithms run faster in C, what has this told us? A naive answer would be that the experiments have confirmed the hypothesis that C is the faster language for this sort of algorithm. A more subtle answer would be that efficient neural networks are correlated with neural networks written in C. That means that when the algorithm is written in C it's likely to run quickly, which is what the experiment reported. This does not necessarily mean that the algorithms implemented in C ran quickly because they were written in C, it may be that there was some other factor involved that the experiment didn't effectively control.

In experimental work it is very important to understand this subtle distinction, otherwise you can easily fool yourself into believing that your experiments have discovered something far more conclusive than is actually possible.

To give you a better idea of how this distinction between correlation and causation works, below are some examples of incorrect conclusions drawn from perfectly reasonable correlations. See if you can work out why the conclusions are unreasonable:

- Children with bigger feet have higher reading ages. Therefore, people with bigger feet are more intelligent.
- Teenagers who text late at night have poor motivation in class (see news reports here). Therefore, using mobile phones leads to poor performance in class (see a more sceptical analysis here).
- In the last 150 years there has been a dramatic increase in the number of people who report being abducted by aliens. There has also been a trend towards global warming. Therefore, alien abductions cause global warming.

In your own work, just be honest and straight forward about your results. If they aren't conclusive then say so and demonstrate your understanding by describing what future work could be done to gather more data.

Some basic dos and don'ts

This is some more specific advice, based on good and bad practice we have seen from students over the years:

DO be clear and honest about what results your evaluation has obtained.

DON'T claim to have "proven" anything if you haven't written a formal, mathematical proof.

DO use an appropriate experiment for your hypothesis. For example, if your work is about evaluating the performance or security of a technique, there is no need to involve real users in your evaluation. If your hypothesis is about usability you really must involve real users.

DON'T use questionnaires unless you can guarantee to get a large sample size of answers (always well above thirty) and you understand the statistics needed to analyse the results. If you are in any doubt at all about this then seek the advice of a qualified statistician before you start your project. If you can't do that, think about using an alternative evaluation method such as semi-structured interviews.

Appendix

Example proof: The square root of 2 cannot be written as a fraction of whole numbers

Theorem

```
The square root of 2 cannot be written as a fraction of two whole numbers.
(This is sometimes called the Theorem of Theaetetus)
```

Proof (by contradiction)

```
Imagine we could write the square root of 2 as a fraction of two whole
numbers, say x/y where x and y are integers.

Let's say that x and y don't have any factors in common, so x/y is already
written in its simplest form and no numbers can be "cancelled out" of the
fraction.

So, we can also say that (x/y)*(x/y)=2

Therefore (x*x)/(y*y)=2

Therefore (x*x)=2*(y*y)

So we now know that x*x is even, since x is 2 times another number.

Since x*x is even, we also know that x is even (by the "Lemma" or little
theorem that squares of odd numbers are never even).

Therefore, there must be a number, which we'll call z such that x=2*z
```

So, (2*z)*(2*z)=2*(y*y)

Or, more simply, 2*z*z=y*y

y must also be even, by the same argument that we used to say that x is even.

If y is also even, there must be some number, which we'll call w such that y=2*w

But if x/y=2z/2w then the fraction x/y was not in its simplest form like we assumed above.

This contradicts our initial assumptions, which must have been wrong.

So, the square root of 2 cannot be written as a fraction of whole numbers.

Chapter 8 workshop: How to evaluate your project

Below are a number of case studies describing student projects. Some are real and some are made up. For each of them, write a paragraph explaining whether or not you think the experimental technique that the student has chosen is appropriate for their research question and why. If you think the technique chosen doesn't fit the project well, suggest a more appropriate one.

Case study 1: using digital wallets to secure e-commerce sites

Ahmed is studying for an IT Security degree and has started his project which is about using digital wallets to secure web-based shopping sites. He plans to build two simple websites which allow users to buy some basic products. One site will be secured with digital wallet and the other will be secured by more conventional means. When Ahmed has built his two websites, he will ask some of his friends to use both of them and then ask them to answer a short questionnaire based on their experiences.

Extra credit

Read through Anderson (1994). How might Ahmed use the ideas in that paper?

Case study 2: a better development environment for novice programmers

Jenny wants to know how software tools can best be used to support novice in learning to program for the first time. She is going to implement a simple programmers editor in Java and, based on her literature review, she will add some novel and interesting features to her editor. The only "novice" programmers Jenny has access to are the first year students at her University. She is going to devise a workshop for them which will teach about twenty students a new feature of the programming language they learn in the first year. About half the students will complete Jenny's workshop using her editor and half will use the editor they usually use in class. Some of the first year students have come direct from school or college, some are "mature" students, some have done used various programming languages before (either in college or professionally) and some have never seen any code before they came to University.

Extra credit

Read through Bornat *et. al.* (2008) and see the materials on the homepage of the second author. How could Jenny use the ideas here to make her study more rigorous?

Case study 3: is MySQL faster than Oracle?

Leroy is studying for a Computer Science degree and the research question for his final year project is "is MySQL faster than Oracle DB?". He plans to compare the two databases by implementing a simple database that he has found in a text book in both MySQL and Oracle. When he has implemented both versions of his database on his home laptop he will run some simple SQL queries once on each databases and time them using his watch. Based on these timings he will determine which he thinks is the faster database.

Extra credit

Read through Georges *et. al.* (2007). How could the ideas in that paper be applied to Leroy's research question?

Case study 4: are online tests more effective for primary school students than paper and pen tests?

Parveen is studying for a degree in IT and hopes to teach ICT in schools when she graduates. Her research question asks whether online or "paper and pen" tests are more effective in primary schools. She is working with primary school near her University and one of the teachers is willing to let her design an arithmetic test for a class of twenty students. Parveen will devise a simple formative test and will deliver that test to half the students using the method that the teacher currently uses. The other half of the students will take the test using a website that Parveen will create specifically for primary school pupils. After the students have taken the test Parveen will interview the teacher, using a semi-structured interview technique, and determine how effective her website was.

Extra credit

How can Parveen rephrase her research question to make it more likely that her project succeeds?

References

Anderson, R.J. (1994) Why Cryptosystems Fail *Communications of the ACM* 37, 11 (November 1994), 32-40. DOI=10.1145/188280.188291 http://doi.acm.org/10.1145/188280.188291

Bornat, R.; Dehnadi, S.; Simon (2008) Mental Models, Consistency and Programming Aptitude. In Proceedings of the Tenth Australasian Computing Education Conference (ACE2008), Wollongong, Australia, January 2008. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 78, Simon and Margaret Hamilton, Ed. [See also: http://www.eis.mdx.ac.uk/research/PhDArea/saeed/]

Andy Georges, Dries Buytaert, and Lieven Eeckhout. (2007) Statistically rigorous java performance evaluation. In *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications* (OOPSLA '07). ACM, New York, NY, USA, 57-76. DOI=10.1145/1297027.1297033 http://doi.acm.org/10.1145/1297027.1297033

Chapter 8
Pitches, Presentations and Viva Voces

I don't know what I think until I see what I say
E.M.Forster

The act of speaking and the act of writing are quite different. Only rarely in life do we carefully prepare what we are going to say beforehand; maybe for a job interview, or a wedding speech, but the occasions are fairly few. For the rest of the time the gap between thinking and speaking is so small as to be unnoticeable; most of us turn thought into speech with effortless ease. Conversation is a natural part of our everyday life.

Not so for writing! Planning, drafting, revising until the text becomes close enough to what we intended is an arduous task. Writing feels hard. Speaking feels easy.

One thing I have often noticed when supervising projects is how little some students talk about their own ideas. I think this is a mistake. In general, the more often you discuss your ideas and talk about the problems you have to solve, or the interesting research you have just read, the easier you are going to find it to organise your thoughts (and your writing).

Who you talk to doesn't really matter – it's the act of talking that helps you clarify what you think and what you are not sure about (although talking to someone interested in your ideas like your supervisor, or an expert academic, or other students, certainly helps).

Over time you will find that you develop better ways of explaining and discussing complex project ideas and, as you research, your specialist vocabulary increases in ways that become helpful when you need to start writing up.

Talking is good. Talking helps.

A great idea for a movie.

I often ask my project students to develop 'pitches' – short spoken introductions to their project with which they might interest other people.

The word 'pitch' has a long and interesting history, originating in the concept of placing or throwing; as in pitching a tent or pitching a ball. The words 'sales pitch' originate in 1876, probably from 'a stall pitched as a sales booth'. In the twentieth century the word moved into advertising: 'pitching for a contract' and from there into the movie business: 'pitching a film idea'. The link to movie pitching is the most useful for us: imagine you are a young screenwriter with a great idea for a film. You are in London (or Hollywood) and you see a famous producer. You are going to need to interest her in the film in the shortest time possible (she is on her way to lunch) and with the most impressive and exciting concept. You need to be 1) interesting and 2) quick!

So it is with project pitches – very often you will need to explain your project ideas succinctly and clearly to people you wish to interest in your project (maybe a client company, or an expert academic) or people you wish to help you (like test users or members of the public).

To Be Continued!

Chapter 9

The top 9 writing mistakes made by project students

A large part of any final year thesis project is the write-up, but every stage of the project will involve some form of writing. Whether you are writing a proposal, an interim report, a draft report, documentation or the final thesis, a very large part of your time will be spent composing text. These are the top ten mistakes we see from students year on year, avoid them and you can save yourself a lot of time (and earn a lot of marks)...

1. Unsubstantiated claims

As a general rule, any statement of fact or opinion about your work, or your topic of study should be substantiated in some way. You don't have to worry about the 'bleeding obvious', we know that 1+1=2 (unless your work is in the "foundations" of mathematics or philosophy, in which case 1+1 may well be undefined), but anything less obvious should be backed up by solid evidence. That evidence can be a reference to literature, an argument you make in your writing, the results of your own experiments, or any other suitably rigorous evidence. Avoid so-called *sweeping* statements that are effectively impossible to substantiate:

- "All Object Oriented programs couple algorithms and data". Do they? All of them?
- "Testing improves the performance of students". Really?
- "Everybody is on the Internet nowadays". This is provably false!

These sorts of claims are poor academic practice and suggest that you have been a little sloppy in your thinking about your work, and of course that's not the impression you want to give.

2. Dangling pronouns and other references

This seems to be an almost universal problem with student writing. Consider the following paragraph:

In his 1953 paper, Henry Gordon Rice proved that for any non-trivial property of a partial function there is no general decision method to determine an algorithm that computes a partial function with that property. It has far reaching consequences for compilers, static analysis and other fields in practical computing.

What does the "It" in the second sentence refer to? Rice's theorem, his paper, or something else? It isn't clear from the text, although we can guess that the author meant to discuss the theorem. Better though, to be clear about the meaning in the first place. Every pronoun ("I", "he", "she", "it", "that", "who", etc.) should clearly refer to exactly one noun. The first sentence gets this right, it is clear that "his" refers to the noun "Henry Gordon Rice" and not any other noun in that sentence. So, we could improve the paragraph above by re-writing it:

In his 1953 paper, Henry Gordon Rice proved that for any non-trivial property of a partial function there is no general decision method to determine an algorithm that computes a partial function with that property. Rice's theorem has far reaching consequences for compilers, static analysis and other fields in practical computing.

3. Using a secondary source rather than a primary one

It's much easier to read the popular press, blogs and other "informal" media than research papers. Very few marks will be awarded for this, though. In a final year project you need to show that you can perform a small academic study, so marks will be available for reading peer-reviewed academic literature. There are two major pitfalls to avoid here. Firstly, you will occasionally come across some disreputable conference or journal which does not use peer-review. Worse, it is possible to come across "articles" on the Internet which have citations and publishing records and look, to all intents and purposes, like a genuine piece of academic writing, but have actually never been submitted to a journal or conference. This is very poor practice on the part of anyone who puts this kind of thing up on their own blog or website, but it does occasionally happen. A reputable publishing venue will have some sort of statement on their website stating how articles are reviewed (look for "Instructions to Authors"). This should say that every article is reviewed by at least two people and sent back for corrections before being published. Without this sort of peer-review any poor standard of work can be "published" without anyone checking even basic standard of good practice, such as detecting plagiarism.

Secondly, whatever references you cite should be *primary*, rather than *secondary* sources. A primary source is one where the author(s) reports work that s/he (they) have personally carried out. A secondary source is one where an author reports on work that someone else has completed and published elsewhere. Secondary sources include news reports, magazine articles and blog posts about research completed by others.

4. Confusing structure and use before definition

Any academic writing should generally be written for a reader who is an expert in the general field of the study, but not necessarily in the specific area of the study. If, for example, your final year project is on ‘genetic algorithms’, your work might be marked by an expert in artificial intelligence, or in computer science generally, but not necessarily by an expert in genetic algorithms. So, write with that in mind, and make sure that you don't use any specific technical terms without defining them. This is often very difficult to get right at first, especially when you have been working with your own ideas for a very long time. A good plan is to swap drafts of your work with a fellow student who is on the same degree course but working in a completely different field for their final year project. If you can both understand each other’s work, that's fine. If not, make changes.

5. Claims of "proof"

At the beginning and end of your dissertation you will want to set out the aims of your work and describe the conclusions you have reached. Occasionally we see students writing sentences such as "this study proves that ...", "this thesis will prove ...", and so on. "Proof" is

specifically a mathematical method, and if you have genuinely proven a theorem, by all means say so. If not, then don't use the word "prove" and be *very* careful about what you do claim. For example:

- If you have tested a piece of software and it has passed all your test cases, then you have shown that your software is free of the specific errors you have checked for. You have not shown that it is "error-free" or "works".
- If you have performed some sort of user testing, or any other usability / accessibility testing, then you may have demonstrated that the system under study is "usable" or "accessible" as far as you have tested it. However, without a large-scale study that is as much as you can claim. Be **very** circumspect about reading research in the area of usability; there is much good research ***but also much which is over-blown.*** Be especially careful of authors who also run consultancy practices, make sure you cite their academic literature, and not anything that could be considered advertising. Make sure everything you cite is peer-reviewed. *[Peter's comment: So True!]*
- Be very, very careful about using questionnaires. I usually tell all my students to just avoid them altogether. It is very; very hard to produce a questionnaire which holds up under academic scrutiny and you will need an amount of statistical sophistication to produce sensible results. Also, you need a very large sample-size because your questions will be circumscribed (and for other reasons). This makes questionnaires very difficult to use in short, single-person projects. If you are in any doubt, then use a "semi-structured interview" to interview test subjects and the "talk-aloud protocol" or "cognitive dimensions of notation" for usability testing. Before you start, read some papers on evaluation methods, such as Hollingsed and Novick (2007) Usability Inspection Methods after 15 Years of Research and Practice, or Hornbaek and Law (2007) Meta-Analysis of Correlations among Usability Measures.

6. Ad-hominem remarks

We see this very rarely, but just occasionally a student will forget that they should be writing about academic research, and criticise an author directly. Examples include "$X is stupid", "it would be stupid to think that..." "People who use Macs are idiots" ... "nobody in their right mind would ... "and so on. Don't do this, stick with the research and don't criticise the person.

7. Don't be 'meta'

If there's one piece of advice students regularly misunderstand it's that you should be cautious and critical of the literature that you read and cite. Of course, you *should* be critical and cautious, but you should also be sure that you are writing about the content of the research that you are reading about, not the quality of the writing. I should probably say, this piece of advice does not hold up if you are studying literary criticism, or anything similar, but for science-based subjects, stick with the science. Don't say things like:

(Danson, J, 2009) is a poorly written paper. It is confused and hard to follow.

Also, don't say things like:

The problem with this paper is...

If there's a problem with the research that the paper describes, then by all means discuss that, but not the writing.

8. Don't write a giant list

(See also the chapter on literature reviews) one common mistake we often see is to structure a thesis as if it is a list of points, not a single piece of prose. Do not write about the literature in your field one paper at a time, try to tell a story about how the field has developed, culminating in saying that there is clearly a gap in the research where your contribution can fit. So, avoid writing like this:

Foo Bar (2007) On the usability of Flibble widgets

Bar describes the Flibble widget, which is used for ...

instead, work your thoughts on Flibble widgets into a longer piece of writing on widgets. If you do need to break up and structure your literature review, make sure your headings are topics which group together related pieces of research.

9. Don't use bullet lists to replace prose

A very common error we see in project write ups is where the student writes a series of 'bullet lists' instead of connected and well supported prose. The problem with this is that bullet lists do not often demonstrate complex thinking; rather, they simply provide 'shallow' summaries of topics (when we need some depth!) For example:

"People use CSS for:

- Interoperability
- Future proofing
- Accessibility
- Conforming to standards"

In this example – what does 'interoperability' mean? Why does CSS provide 'future proofing'? Is CSS the only way to enable accessibility? There is SO MUCH TO SAY on these subjects! The bullet list destroys your chance to demonstrate your research and ideas.

Chapter 9 Workshop: Top 9 writing mistakes

Below are some (made up) examples of academic writing, which all need some serious improvement. For each one, you should comment on what is good and bad about the writing (assume the content is OK, even if you think it isn't!) and then re-write the text based on your comments. For some of the case studies citations will be missing. You don't have to find each paper that is needed to back up the authors claims, but you should note where a citation is necessary.

Case study 1: Object oriented programming

"Object oriented programming" is a phrase made up by Alan Kay in the 1960s. Object oriented languages started with SIMULA67 (Nygaard, 1962) which was invented to help engineers with simulation problems. Other popular languages followed with SmallTalk (Kay, 1978) , CLOS (Bobrow, 1988), C++ (Stroustrap, 1984) and Java (Gosling et. al, 1995). Object-oriented languages have a number of common features, including

- objects,
- classes
- inheritance
- and message passing.

Case study 2: Web 2.0

Web2.0 is all about XHttpRequest, but some people think it's all about rounded corners and gradient fills! Web2.0 became really popular when Google brought out GMail. GMail refreshes every time you get a new email, so that u can see the new email immediately. It does this using XHttpRequest, which is where the server pushes new data to a client (like a web browser). This is different to the usual thing, where the client requests data from the server. This is also how sites like Facebook and Twitter update your news feeds. The other thing people say about Web2.0 is that "in Web2.0 you provide the content and they make the money", which refers to the fact that a lot of websites like Flickr and Blogspot get the users to provide all their content rather than making their own.

Case study 3: The Internet of Things

Like "Web2.0" the Internet of Things is a silly marketing term. It refers to the idea that in the future a lot of physical objects will be tagged and tracked on websites. That's not all there is to the Internet of Things though, M2M communication is another part of it. When M2M becomes common, a lot of apps will behave more automatically and won't need real users