MSc Computer Science
Summer 2012 Individual Projects

Projects Organizer:
Dr. Graham Roberts
g.roberts@cs.ucl.ac.uk

Contents

1 Introduction

This document describes in detail how the individual projects are structured and covers all the things you need to know about or do[1]. Please read all of this document carefully!

The key activities are summarised here and described in detail in the rest of the document:

- March and April 2012, think carefully about what kind of project you would like to do, put together one or more project proposals based on either your own ideas or those published by staff, find a supervisor and do as much preparation as possible.
- If possible, register for your project by the start of the start of your exams in late April and definitely before your last exam in May. You must have a supervisor to register.
- Early June (after your final exam) to mid-August do the main project work.
- Mid-August to the end of your project focus on writing your Project Report.
- Submit your Project Report, on or before the 4th of September 2012.

2 Finding a Project

2.1 Finding a Topic and Supervisor

Most members of the academic staff, and some members of the research staff, are all potential project supervisors. However, it is not the case that any member of staff is able to supervise any project. A number of staff, but not all, will want to specialise in certain types of project, usually related to their research area. You should aim to find someone whose interests are a good match for your own. This person will be your internal supervisor.

A number of projects also have an external supervisor, where you are working with someone from outside of the department. We strongly encourage projects with external supervisors, so take advantage of this if you can.

There are a number of routes to finding a supervisor. The principle route is through the project suggestions web page.[2] This page lists most staff who are available to supervise projects this year, along with project ideas from potential external supervisors. This is not a restricted menu from which you will be forced to make a choice, and there are a number of other options:

[1] But if you find anything missing please let me know so I can add it!

[2] http://www.cs.ucl.ac.uk/students/student_information/individual_projects/project_ideas/

- You are completely free to devise your own project idea. However, you will still need a supervisor, and it may be best to talk to someone before your ideas become too fixed, as an academic will be able to judge whether the idea is feasible (students tend to propose very ambitious projects!) or could be made so with a bit of tweaking (you need to be flexible and willing to take advice). If you have an idea but don't know who might supervise it, talk to your tutor from the previous year about who might be a good supervisor for your project, or contact the projects organiser.

- Have you had any experience outside the department – for example, previous work experience, holiday work or an internship – that could generate a project idea? This scenario could involve an external as well as internal supervisor, and many such projects have been successful in the past.

- Have an informal chat with staff members you know, either because you have taken one of their modules, or because they tutored you. Many projects get generated by this kind of informal contact. This works best if you have some concrete ideas you can talk about.

- Have a look at staff research interests as described on their web pages. The project suggestions web page has links to all the staff web pages. Research interests are another good way to find out the kinds of projects staff may be interested in supervising, even where there are no project suggestions listed (as for a new member of staff). Email the relevant person and ask them if they could supervise a project in an area of joint interest.

A good project will have depth and challenge, making good use of the Computer Science material you have been learning about. Projects also provide a great opportunity to learn about new ideas and concepts not covered in the other modules you take. You can try out different programming languages, experiment with cutting-edge technologies or explore a new area of Computer Science. Don't limit yourself to what you already know!

All projects are required to include research into the problems being investigated, analysis of the problems and potential solutions, the design and implementation of the solution, and a thorough evaluation of the results.

Most projects aim to design and implement a piece of software, so programming is normally a significant part of any project. Within this constraint, however, there is a lot of flexibility. The software may be a user application developed against a detailed set of requirements or for an external supervisor, but equally it may be a tool needed to carry out experiments (for example, to measure user responses for human-computer interaction experiments). Another variation is that the software can be a proof of concept that a design or algorithm can be implemented and have the desired performance requirements.

Projects that do not involve a significant amount of programming are not impossible. For example, you may want to work in an area of theoretical Computer Science perhaps involving a lot of mathematical analysis. Providing you have a supervisor who agrees to supervise such a project and deems it to be suitable,

then it can be done. If you undertake such a project it should demonstrate relevant skills in areas such as mathematics, analysis, synthesis, critical assessment and design.

2.2 External Supervisors

Every year there are a number of projects that have an 'external supervisor'. These projects generally arise either because of contacts that the student has made themselves (e.g., with a former employer), or because CS staff have academic collaborations outside the department that have generated project ideas, or an external person or company has contacted the department to propose some interesting project ideas.

An external supervisor is anyone not registered as a Computer Science Examiner here at UCL; essentially anyone from outside the department, whether a UCL academic or not.

Every project with an external supervisor *must* also have an 'internal supervisor', who is an examiner within the department. In general you cannot count on your external supervisor being able to provide you with help in areas such as technical support in programming. They will be an expert in the domain of application of the project, but not necessarily expert in CS matters. So check how much technical support your external supervisor can offer. If it will be limited, make sure that your CS internal supervisor is knowledgeable in the necessary areas.

External supervisors will be expected to provide 'domain knowledge', anything you need to know to tackle the problem that is not covered on our modules. In the past this has ranged from the physiology of the human ear to the tourist industry in Northern Cyprus. The external will be the source of the project idea and may in some cases take on the role of a 'client', whose requirements you may need to survey in some detail before even beginning to think about coding up a solution. Your external supervisor doesn't need to be a computer scientist, and probably won't know what a UCL CS project report is supposed to look like in detail. Getting your work into the required shape is the responsibility of your internal supervisor.

You should, in addition to your weekly meetings with your internal supervisor, be in regular contact with your external supervisor too. It is also helpful if your internal and external supervisors can be in occasional contact – phone or email contact is fine.

Where there is any difference of opinion between your internal and external supervisors, you must be guided by that of your internal supervisor. That person is more fully aware of the academic requirements that you must meet, and of the rules and procedures of the department. If your external supervisor does not easily accept this, get your internal supervisor to contact them and resolve the situation.

As an external supervisor is not a registered CS Examiner, they are not able to be either a first or second examiner for a project. However, their input to the marking process is still valuable; what normally happens is that the first examiner (the internal supervisor) will consult with the external supervisor before deciding on a mark. Another member of the CS department, the second examiner, will then provide a further independent mark.

Finally, although we in general welcome the involvement of external supervisors as a source of interesting project ideas, we can't – beyond the collaborative projects already mentioned on staff suggestions pages – arrange them for you. We can't, for example, answer queries like "can you find me a project where I'm working inside a bank?". If you don't know anyone in banking, and there are no banking-related suggestions already on the staff pages, then you're out of luck!

2.2.1 Intellectual Property Rights

For most projects you own the copyright on what you create by default. However, where there is an external supervisor or organisation, a close connection with funded research work or some other substantial input from another your supervisor or another source, an issue that may arise is that of who owns the Intellectual Property Rights (IPR) and how widely information about the project can be made available.

If you think IPR may be an issue, make sure that your supervisor, and other relevant people, agree the IPR details and that external people know you have to write a detailed report on your work in order to get a decent mark for your project.

Not all details of a project have to go into the final report. Those that are most sensitive as far as your external supervisor is concerned may not be very important from our point of view. For example, in the case of a project with a bank on using neural nets for credit rating purposes, all that was required was that the origins of the vector components input to the neural network (relating in some way to an individual's credit-worthiness) not be discussed. Since from the point of view of a neural net (and its trainer) these are just a string of numbers, this didn't really affect the write-up at all, and instantly converted the report from very sensitive in the eyes of the external supervisor to totally innocuous. Also, you have the option of adding a 'restricted access' disclaimer to your final report, stating that it can be seen by supervisors and examiners only.

Most IPR or similar issues can be easily resolved with a bit of goodwill and common sense on both sides. If you have any reason to think that problems of this sort may arise, get your two supervisors to talk to each other straight away; don't leave it until writing-up time.

Occasionally a project with an external client or supervisor requires a more formal agreement, which is legally sound. In such cases it is possible to arrange for UCL Research Services, or another relevant UCL department, to draw up a contract between UCL, the client and the student. The contract will properly

define issues such as who owns the IPR or what the deliverables should be, and make sure that everyone is properly legally protected. Your internal supervisor or the projects organiser will be able to provide more information if such a contract is needed.

2.3 Project Registration

When you have sorted out a project and have an agreed supervisor, you should formally register your project. You do this by email, by filling out a registration form[3].

Please be prompt in registering when you have a project arranged! Don't wait until every detail of the project is finalised. It is more important that we know as soon as possible (i) that you have a project; (ii) who your supervisor is; (iii) a working title and basic description of the content (these latter can both change, with your supervisor's agreement, as the project develops).

2.4 Project Resources

The great majority of projects can make use of readily available resources:

- Lab computers or your own machine
- Standard operating systems and software
- Open Source software
- On-line and UCL Science Library resources

These resources are either provided to you as a UCL student, available for free or, in the case of computer hardware (desktops, notebooks), you probably already own.

The web, of course, has a great deal of material available but don't ignore the UCL Science library as it gives you access to the full digital libraries of societies such as the ACM and IEEE. For research purposes, especially if you are doing a research-oriented project, this will give you access to a extensive collection of research and academic publications.

For software design and development, there are many high quality Open Source tools and other software available, particularly for languages like Java, Groovy, C, C++ and UML. This software is available for free download and many projects make use of this resource.

Do not pirate (i.e., steal) software or use any software that you don't have a proper license to use. Always respect copyright and licences. Being found to have used pirated or unlicensed software can have serious implications for your project.

[3] http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/images/students/Projects/msc_form.txt

A wide variety of licenses are used with Open Source software (such software may be free to download but will still be licensed). Most of the time the license will have no impact on your project but if you are planning to make your software available to other people, make it Open Source, or you are developing software for an external supervisor, you do need to take notice of which license is used and what its consequences are. In particular, a license like the GNU Public Licence (GPL)[4] requires the source code to be made available to users of the software. If you use Open Source code or libraries in your code, then the licensing can have important implications for the code you write.

2.4.1 Additional Resources

If your project needs non-free or specialised resources then it is the responsibility of your supervisor (internal or external) to provide those resources. Make sure at the start of the project that such resources will be available and that you will have proper access to them. Also make sure that documentation, training and safety procedures are properly available if needed.

There is no 'project budget' available for buying software or hardware, if your supervisor cannot provide it. Also we do not expect you to spend your own money to pay for such resources. If the resources you need for your project idea are not available then you will need to modify the project or choose a different one.

2.5 Backing up Your Work

You are reminded that if you are working on your own computer, it is YOUR responsibility to back-up your work and deal with hardware problems. Software or hardware problems (including last-minute problems with printers) cannot be used as excuses for missing the project submission deadline.

You should make emergency plans for any breakdowns, perform regular backups (preferably by copying your files onto your filestore at UCL) and not use pirated or dubious software. If you use Microsoft Windows make sure you use good virus checking software. On-line backup services, such as Dropbox, provide a good way of keeping copies of your work, and many are free for up to two gigabytes of data or so.

2.6 Data Protection and Security Issues

It is possible that your project may involve the use of data that records personal details of real people. Such use is covered by the Data Protection Act and this states that all such information must be kept with an adequate degree of protection and security. For that reason, we would prefer that you use dummy details wherever possible. If this is not possible, then it is essential that you obtain clearance from the chair of the security subcommittee (currently Prof. Stephen Hailes), giving details of the data you wish to hold and the security measures you propose to put in place before you use or capture the data. Further, you

[4] http://www.gnu.org/licenses/gpl.html

must register the data with the Data Protection Officer, again before you use it. The Data Protection Officer is Lynette Hothi.

The UCL firewall imposes strict limits on external access to machines within the department. While you can access departmental machines via secure mechanisms such as ssh and VPN, these are intended for personal use only, not to provide more general access from outside UCL. It is not permitted to set up a server, such as a web or database server, on a departmental machine and access it from outside UCL, especially to run any sort of publicly available service. There may be ways to circumvent the firewall, doing so would be considered a serious security violation, so please don't try!

3 During the Project

The summer projects start in early June and finish in early September, lasting roughly three months. You work full-time on your project and there is no other teaching.

It is probably a good idea to have a short rest after your last exam, which is typically in late May or early June. Up to a week is sensible, longer will take too much time away from your project. Note that you do not need to wait for your exam results before starting! Start as soon as you can but be aware that your supervisor is also likely to be coping with the very busy period associated with marking and examiners' meetings. Regular weekly project tutorials with your supervisor should start as soon as possible.

3.1 Tutorials

It is your responsibility to arrange and attend tutorials, and to be properly prepared for each tutorial. Make sure the day and time of each tutorial is in both your own diary and your supervisors diary.

From early to mid June through to submitting your project report in September you should normally see your supervisor at least weekly. You should expect that your supervisor will take a holiday of one or two weeks at some point during your project, or may be attending meetings or conferences. Ask about this early on, and make sure that you have work to do while they are away that you feel confident you can complete in their absence. If your supervisor will be away for more than two weeks during the summer, you should ask for suitable alternative supervision. Once again, be sure to organise this early, and be sure that you are happy with the arrangements, since they are very difficult to re-arrange at short notice. Do not take an extended holiday yourself during the summer; the time for your project is limited, and you will find that this is a big mistake. Save your holiday for September after submitting your project.

At the start of each tutorial you should report on what you have done in the last week, then move on to discuss any problems or issues, your plans for the next week and so on. Your supervisor will give feedback (criticism and praise!), help you plan your work and check that the project is running properly. It is

also a good idea to show examples of your work, demonstrate your program code running or highlight results produced, so that your supervisor can build a clear picture of what you have produced. Be honest with your supervisor, especially if your progress is slower than you expect or you are having trouble understanding things.

Always remember: this is *your* project. You should be the one in charge and driving it forward. Your supervisor is there to give you help, advice and feedback but not to do your work for you! Dont expect to be given a list of instructions on what to do each week; make your own decisions.

3.2 Establishing the aims of your project

This is the single most important aspect of your initial planning. The aims of your project MUST be clear to you before you begin your work in earnest. Discuss them with your supervisor: what is your project fundamentally about? What are you hoping to design/build/investigate? Within the overall framework of your project, what is the target, what would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution? After you have thought about these issues, it should be possible for you to reduce your aims to one page of A4. If you can't summarise your aims in this way then you need to do some further thinking and planning. Make sure that your supervisor is in agreement with your aims as finally established.

It is strongly advised that you keep notes of your progress in the form of a logbook, detailing the decisions you make as the project advances. You will find this invaluable when it comes time to write your project report.

Always remember: this is *your* project. You should be the one in charge and driving it forward. Your supervisor is there to give you help, advice and feedback but not to do your work for you! Don't expect to be given a list of instructions on what to do each week, make your own decisions.

3.3 A Typical Project Schedule

A typical project schedule is as follows:

- Late May/Early June: Background reading, detailed planning, preliminary analysis and design, identifying and learning how to use tools and code frameworks, coding small test programs and simple prototypes.
- Mid-June through July to mid-August: The bulk of the problem solving, design and implementation (serious coding, not just small tests and prototypes).
- Mid-August: At least three weeks before the submission deadline start writing your project report full-time. At this point you should finish any significant implementation work and concentrate on writing, since you will be marked more on your ability to present and explain your design and implementation decisions rather than just on the code (but good quality code is important). Look at the guidelines for the report structure and

make sure that you keep to the format requirements in relation to page limit, font size, etc.

- Late August: The detailed editing of your project report to get it into the best shape possible. Don't underestimate how long it takes to edit and refine your report.

- Early September: Leave about three days before the submission deadline for printing and submitting your report. At this time there is huge pressure on printing resources both within the department and college IS services, and because of this last-minute problems can arise. Even if you are printing at home or elsewhere outside UCL, don't count on having a trouble-free time – what would you do if your printer at home broke down the night before the deadline? It is not worth taking this kind of chance, aim to be finished several days before the deadline instead.

  Note that printer failure is not an acceptable excuse for missing the deadline – you are expected to plan around such potential problems. Also beware of ink jet printers getting through a lot of ink cartridges!

- The project deadline: This is the date and time by which you must submit your project report (see later for the date and detailed information about the report). It is unusual for the date of the September project deadline to change; if a change is necessary, this will be announced by email. You are required to check your UCL email regularly (at least once every day, more regularly as deadlines and exams approach); not reading your email is not an acceptable excuse for missing a deadline.

- The Project Demonstration: Either a week or so before or after the project deadline you should arrange a time to give a demonstration of your project results to your supervisor and your second examiner (the other person who marks your project, see section 7 for more information). This is an informal event where you have the opportunity to explain what you have done and demonstrate what you have created. The demonstration is of particular benefit to your second examiner as, unlike your supervisor, that person will not have been following the weekly progress of your project and will be seeing it for the first time. The demonstration should last 15-20 minutes and does not require you to give a formal presentation or prepare items like Powerpoint slides.

It is very easy to over-estimate what you can achieve in the time available, so monitor your progress carefully and focus on the elements that will have the most value to your project. By far the most common lament supervisors hear is along the lines of "I've only done half of what I expected by this time". The chances are you will say something like this too but don't worry, it is quite normal!

Many projects are best structured by taking an iterative or incremental approach in the main development phase. Having determined what you are trying to design and build, start by constructing a very basic but working version of it. Then step-by-step add one feature or requirement at a time, so that you go from one working version to the next. Keep going while there is time available,

reviewing your progress after each step and selecting the next most important feature to add. It is unlikely you will have time for every feature you would like to have, so make sure you focus on adding what gives most value to your project. Some features may have to be dropped or implemented in a basic form to demonstrate their feasibility but no more.

4 The Main Report

The Main Report documents the results of your project work and is a very important deliverable as it will be read by all the examiners that mark your project. Examiners give credit for good quality writing, so aim for a clear and concise writing style. You should use relevant notations, terminology and computer science knowledge. The proper evaluation and critical assessment of your work is also important. Above all your report should be both readable and interesting to read. Beware of creating a report that is too long, tedious to write, and tedious to read. Write something that your examiners will want to read!

Before starting to write your report you should plan its structure by creating a contents outline and get your supervisor to review the outline. Then, as you write your report, you should be showing your supervisor draft chapters to get feedback. Note, however, that it is *not* your supervisor's job to be your editor or proof-reader, so don't expect your supervisor to fully read everything and comment in detail on your drafts.

It is strongly recommended that you plan your contents outline during the first half of August and also begin to do some writing in order to get started – getting started is often the hardest part of writing the report so don't keep putting it off! In the second half of August writing your report should become the main project activity. Don't underestimate how long it takes to write *and* edit a good report. The worst mistake to make is to believe you can write your report quickly right at the end of the project; you can't, and won't have time do a good job. Be ruthless when editing to get the text into the best shape possible.

4.1 What to write about

1. Write about the *interesting parts* of your project. No-one wants to read about every single detail of how you implemented your project! Devote appropriate space and time to this aspect, addressing issues such as:

   What design choices did you have along the way, and why did you make the choices you made? What was the most difficult part of the project? Why was it difficult? How did you overcome the difficulties? Did you discover anything novel? What did you learn?

2. Write about the context in which your work fits. You should provide enough background to the reader for them to understand what the project is all about. For example: What problem are you solving? Why are you solving it? How does this relate to other work in this area? What work

does it build on?

Your reader is technically literate, but may not understand in detail the area of your project or any tools or building blocks you've used, so help them understand.

3. Give an overview of your work. If your project involved designing a system, give a good high-level overview of your design. In many projects, the initial design and the final design differ somewhat. If the differences are interesting, write about them, and why the changes were made.

   If your design was not implemented fully, describe which parts you did implement, and which you didn't. If the reason you didn't implement everything is interesting (e.g., it turned out to be difficult for unexpected reasons), write about it.

4. Describe any preparatory work you needed to do. Some projects involve detailed requirements capture, and for others the requirements are essentially handed to you. If you had to do detailed requirements capture (e.g., a survey of potential users, etc.), then this might be described in some detail. If you didn't do detailed requirements capture, then just summarise the requirements.

5. Provide a User Manual. For all software projects, you need to include enough information to show how to run the software (see the information on "Appendices"). However, some projects are heavily user-centered, and so the user interface is a key part of the project; in these cases a user manual would not just be an appendix, but a core part of the project report. However, structuring it as a user manual may not be the most effective way to present your ideas, so use your discretion.

6. Describe how you evaluated your work. Most projects involve the creation of software. Building software that works well is difficult, and you may have spent a lot of time on testing. In most cases your reader does not want to know about all the detailed tests you ran, but they will be interested in your testing strategy and philosophy. Some software is particularly hard to test, and in such cases you might need to go into quite some detail on why it's hard to test, and how you overcame this obstacle.

   Some projects involve designing systems whose final correct behaviour is not known in advance. Usually such projects would be classed as "research" projects. Typically such a project report will devote a lot of space to evaluation of how the final system behaved, and where it worked well and where it didn't. No system behaves well in all circumstances, so your reader will be interested in both how well it works, and in the circumstances in which it works less well.

7. Provide a critical evaluation of your work. Your reader wants to know that you understand the advantages, disadvantages, strengths and limitations of your work. No project is perfect, there's never enough time for that! Provide a critique of your work.

Some examples: Did the design did the job you intended, or were there problems? How does the resulting system compare against the competition? If you didn't finish implementing all the features, how hard do you think it would be to do so? What advice would you have for someone who wished to take your system and use it or extend it? What would you do differently if you could start all over again?

4.2 Report Structure and Content

The structure of a typical report is outlined below. Your report structure can and should vary to suit your specific project. Don't feel compelled to write text to fill out each section described below, especially if there isn't anything interesting to write about. Add sections that are relevant to your project.

A report should be broken down into a series of chapters, with each chapter consisting of a sequence of numbered sections and sub-sections. A typical list of chapters for a design and programming project is described here but, remember, you can adapt or replace this to fit the needs of your project.

- Chapter 1 Introduction
  - Outline the problem you are working on, why it is interesting and what the challenges are.
  - List your aims and goals. An aim is something you intend to achieve (e.g., learn a new programming language and apply it in solving the problem), while a goal is something specific you expect to deliver (e.g., a working application with a particular set of features).
  - Give an overview of how you carried out the project (e.g., an iterative approach).
  - A brief overview of the rest of the chapters in the report (a guide to the reader of the overall structure of the report).

  This chapter is relatively short (2-4 pages) and must leave the reader very clear on what the project is about and what your goals are.

- Chapter 2 Background Information and Related Work
  - Outline and reference the sources of information you are drawing on (papers, books, websites, etc.). State how each relates to your work.
  - If relevant, a survey of similar programs or applications to yours, and how yours is differentiated.
  - Outline the software, tools, library code, frameworks and similar that you are using.

  You should not include well known things (e.g., HTML or Java) or try to give tutorials on how to use a tool or code library (use references to books and websites for that information). Everything you include should be directly relevant to your work and the relationship made clear. This chapter is likely to be fairly substantial, perhaps 8-10 pages.

- Chapter 3 Requirements and Analysis

- Give the detailed problem statement.
- A structured list of requirements.
- Use cases (a use diagram and list of use case titles, with the full use cases appearing in the appendix).
- Results of analysing the requirements to extract information. For example, data modelling to find the data to be stored (ER diagram), views/web pages needed and so on.

The level of detail of the requirements and use cases will depend on the nature of your project. If you are doing a Software Engineering based design and implementation project, then they will need to be done thoroughly. If there is a substantial body of requirements and use cases, then a summary should be provided and the bulk of them should be provided in an appendix section, rather than directly in this chapter.

If your project is not Software Engineering oriented, then you still need to describe the requirements you are working to and relevant analysis information.

The length of this chapter depends on the kind of project, but you are typically looking at 5-6 pages.

- Chapter 4 Design and Implementation
  - Describe the design of what you have created.
  - Start with the application architecture, giving its overall structure and the components that make up that structure.
  - Give a description of the design of each of the the components that make up the architecture.
  - Include the database or storage representation.
  - Provide implementation details as necessary.

As with other chapters, the structure and contents of this chapter will depend on the nature of your project, so the list above is only a suggestion not a fixed requirement.

Find an ordering and form of words so that the design is clear, focusing on the interesting design decisions. For example, what were the alternatives, why select one particular solution? You have a limited number of pages so be selective about details. Also remember that someone (your examiners!) has to read this so don't overwhelm them with intricate descriptions of everything that only you can follow – but do make sure the key details of the solution are in place. Use appropriate terminology and demonstrate that you have a good understanding of the Computer Science principles involved.

You can use diagrams and screen shots to help explain the design but don't overuse them. Diagrams and screen shots should add information, not duplicate what is written in the text, and definitely avoid page after page of diagrams as this will disrupt the flow of your text. Where relevant, UML diagrams can certainly be used but, again, don't flood the chapter with diagrams. Additional diagrams can always be included in an appendix section.

It may be useful to include sections of code to highlight how a particular algorithm is implemented or how an interesting programming problem was solved. However, avoid lengthy sections of code, as they can disrupt the flow of the text. Also make sure that your code fragments are readable, easy to follow and properly laid out. It may be better to use pseudo-code rather than actual code, especially when describing an algorithm. If you need to make use of longer sections of code, you can put the code in the appendix and reference it from the text.

An alternative way to organise the content of both this chapter and the preceding one, suitable for some projects, is to have a sequence of chapters for each major iteration of the project. This allows the progression of the project to be shown, with each iteration building on the last.

This is a core chapter in your report and will usually be quite substantial, 10 pages or more.

- Chapter 5 Testing
  - Describe your testing strategy (unit, functional, acceptance testing and how they are carried out). How were test cases selected.
  - Examples of specific tests and how they were carried out (e.g., using mock objects to break dependencies). Focus on the interesting cases.
  - A summary of the test results and what coverage was achieved. The detailed test report(s) should appear in the appendix.

  If your project requires substantial evaluation of data and results, or other forms of testing that are not code-based, then adapt this chapter to suit.

  This chapter will typically be 2-4 pages in length but could be more depending on the depth of testing done.

- Chapter 6 Conclusions and Evaluation
  - A summary of what the project has achieved. Address each goal set out in the introduction.
  - A critical evaluation of the results of the project (e.g., how well were the goals met, is the application fit for purpose, has good design and implementation practice been followed, was the right implementation technology chosen and so on).

- Future work. How could the project be developed if you had another 6 months.
- Wrap-up and final thoughts on your project.

This chapter is typically 2-4 pages long but could be longer if the project work requires more extensive evaluation.

- List of references. Give publication details for all the items referred to by references you have made in main text of the report.
- Bibliography. This lists all the sources of information that you made use of during the project but are *not* referenced in the text. The items in the list must be relevant to your project, so don't just list everything you may have looked at or read.

  The list of references and bibliography are often combined into one section labelled Bibliography.

You are not limited to the chapters suggested above but do remember that the report documents your *results* and is not meant to be a diary of progress or a novel. For some projects it may make sense to have a separate evaluation chapter before the conclusions and the way that requirements are specified can also vary as techniques like developing use cases may not be appropriate for your project.

4.3 Appendices of the Report

You should add each of the following appendix sections if they are relevant to your project. A source code listing should always be included, or an equivalent listing if your project has other kinds of outputs.

The appendices provide important background information to your examiner, by including relevant information that should be present but not as part of the main chapters.

1. *System Manual* This should include all the technical details (Where is the code?, What do you type to compile it?, etc.) that would enable a student to continue your project next year, to be able to amend your code and extend it.
2. *User manual* This should give enough information for someone to use what you have designed and implemented. It is a good place to include screen shots of the application.
3. *Supporting documentation and diagrams* If you have additional diagrams or things like a set of use case specifications that are relevant to the documentation of your work then they can be included in an appendix section. Don't just include everything, only items that are directly relevant to the documentation or description of your work.
4. *Test results and test reports* If you have test results that add to the value of the report, but which would not fit within the page limit of the main

report, you can include then as an appendix. Don't add them just to pad the report though.

5. *Evaluation data and results* If your project involves activities such as generating, analysing and evaluating data of some sort, then sample data, descriptions of analysis methods and detailed results can be included here. Make sure that what is included is useful and supportive of your main chapters though, don't just include long lists of numbers or similar that look meaningless.

6. *Code Listing* Your code should be properly presented and formatted neatly. Don't let long lines of code arbitrarily wrap round to the next line, as this looks very messy. In order not to use up too many pages you may like to use the 'a2ps' Unix facility, which allows you to put two pages of code onto one side of paper in landscape orientation – see the Unix 'man' pages for details.

   In general, don't add more than about 25 pages of code listing to the report. If your code does not fit within 25 pages, you can provide a listing of the most interesting parts our your code (but include around 20-25 pages) or parts of the code you may need to reference from the main chapters. If you don't include everything, it must be clear that this is not the complete listing. Make it clear which parts you've include, add a brief explanation of why you've included these particular parts, and provide a brief summary of which code you have omitted.

   You must make the all the source code (including project and build files) available on a CD or DVD (or on a *thin* USB Thumb Drive). Each copy of the submitted report should include a copy of the disk (see Section 5). The disks should also contain a copy of the report in pdf format and can include any other files you want to submit as part of the project.

4.4 Report Formatting

This section gives basic format requirements that everyone should use. The format requirements are not overly restrictive, for example there is no requirement for you to use a particular type face. However, do not use too many different typefaces in your report, or in general spend too much time developing an elaborate visual presentation. It is better to keep the look of your project report simple and straightforward. An over-elaborate presentation can in fact create a negative impression, that the author thought that the material was rather thin and felt that an eye-catching style might disguise this!

By all means use plotting/drawing applications to create graphs and figures, but if, for example, it is going to take you most of a week to learn to use a drawing package, you would be better advised to hand-draw your figures neatly and get on with something else.

4.4.1 Report Length

The absolute maximum length allowed for the report is 120 pages, where a page is defined as a side of an A4 sheet of paper. When double-sided printing is used this means a maximum of 60 physical sheets of A4 paper, as a single sheet holds two pages, back and front. To save paper and reduce the weight of complete reports we encourage you to use double-sided printing.

The main chapters, excluding the appendices, should not be more than 45 pages, ideally around 40 pages (where, again, a page is defined as one side of an A4 sheet of paper, so 45 pages is roughly 23 physical sheets of paper when printed double-sided). This goal is to be *concise* and to the point. Examiners do *not* give credit for writing a longer report with every possible detail included. Good writing matters!

4.4.2 Title Page

The first or front page of your report is the title page. As well as the title of the project, the year of submission, and your own name, the title page should also include the name(s) of your supervisor(s) and your degree programme (MSc Computer Science). Make sure your name stands out a bit and is easy to find!

4.4.3 Title Page Disclaimer

On the title page, towards the bottom below the other items, you must also include a disclaimer in the words given below.

"This report is submitted as part requirement for the MSc Computer Science degree at UCL. It is substantially the result of my own work except where explicitly indicated in the text."

Then follow this with the words:

"The report may be freely copied and distributed provided the source is explicitly acknowledged."

or, if your project includes confidential information that prevents it being more widely circulated, by

"The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author."

You should consult with your supervisor or the Projects Organiser (Graham Roberts) if you intend to use this statement.

You are reminded that the project is an individual project and that the work submitted should be substantially your own as stated in the disclaimer. Within your report you should identify clearly:

- Which work you have completed by yourself and represents your own individual contribution.

- Which work you have completed in conjunction with other people with whom you have collaborated (such as fellow students from your degree programme).
- Which work you have incorporated from other sources, such as from previous years' students, from external sources (e.g. third party algorithms, methods, publications, source code or code libraries), or from Research Projects with which you have been allied with during your project work (e.g. those of your Supervisor or of external companies).

4.4.4 Abstract

On the page immediately following the title page you must have a short abstract giving a descriptive summary of your project. The abstract should be no more than half a page and typically consists of three short paragraphs:

- What the project is about, the principle aims and goals, and specific challenges.
- How you carried out the project and what work it involved.
- The results and achievements of the project.

The abstract needs to be read quickly by various people to get an overview of what your project is about, so make sure it doesn't get too long.

You will also need to submit one separate copy of the abstract when you submit your project report. This copy should include your name, your supervisor's name and the project title.

4.4.5 Contents List Page

A contents list should follow the abstract. The contents list at the start of this document illustrates how it might be formatted.

4.4.6 References and the Reference List

It is very important that all sources of external information and ideas referred to in your report are properly referenced. Such information includes:

- Published research papers, in publications such as journals and conference proceedings.
- Books, book chapters and specific sections or pages in books.
- Web sites, pages and other online material.
- Recorded seminars, talks and presentations.
- Source code, libraries and tools used by your project.

A complete reference consists of a flag or tag in the your text associated with the referenced material and an entry in the Reference List. Most importantly the Reference List entry must contain sufficient information for the reader to

locate the referenced material if they want to read it.

There are a number of different styles for making references, several of which will be outlined below. You should check with your supervisor for their advice on which style to use for your project.

- Parenthesised or Harvard style. Parentheses are used to denote the reference, for example Bloggs (2002). If the reference forms a natural part of the text and directly refers to someone's work, the name or names are not put inside the parentheses. For example, "The algorithm developed by Smith & Jones (1997) is faster than ...", "The paper by Dent (2010) argues that ..." or "Simpson (1999, 2002, 2006) identifies ...".
  If the text refers to an idea or concept and just uses a reference to point to an example in passing, then the name and date are put inside the parentheses. For example, "It has been claimed that the algorithm is slow (Patel 2003) but ..." or "Earlier work (Smith 1993, Shah 1997) supports the claim that ...".

- Alpha style. A label is placed in square brackets, for example [Simp01] or [KNUTH87]. The label is formed from the surname of the first author, often shortened to three to five characters, followed by the last two digits of the year of publication.

- IEEE style numbered references in square brackets. The reference is a number in square brackets, for example [1], [3].

Whatever style you use, you should use it consistently and not use any other style.

The reference list gives the full details of each reference. Each reference starts with the tag or label used in the main text and is followed by the reference details. For example:
Bloggs, J (2002), "The title of the paper", Journal of Computer Science, vol. 45, no. 2, pp. 207-226
or
[BLOG02] Bloggs, J (2002), "The title of the paper", Journal of Computer Science, vol. 45, no. 2, pp. 207-226
or
[3] Bloggs, J (2002), "The title of the paper", Journal of Computer Science, vol. 45, no. 2, pp. 207-226

The information included in the full reference should include volume and issue numbers, page numbers and any other information needed to locate the referenced text. References to information on the web should include the URL:
Shah, A (2011), "The title of the article", http://www.cs.ucl.ac.uk/info/a123.html
If you specify a URL you should have confidence that it will remain valid for some time (at least until after the examiners have read your report!). If not, then you may need to reference the site rather than the specific page, providing additional information, such as a search term, to help locate the information. However, if possible avoid providing references to unstable URLs.

Your program source code should also include references, for example to point to information about an algorithm described in a paper that you have implemented or where a library being used comes from. If you have copied and pasted someone else's section of source code into yours, then it too must be referenced. For source code you should include the full reference, as it appears in the Reference List, in a comment in the code.

For further information about references, citations and avoid plagiarism problems see the UCL web pages on 'What is a Citation?'. [5]

4.4.7 Other format requirements

As noted earlier, the main body of your report (excluding appendices and code listing) should normally not be more than about 40-45 pages, and should include information about the most interesting aspects of your project. If you write a lot more than this, getting above 50 pages, your report is becoming too long. The extra material risks being regarded as padding, and will not be viewed favourably by your examiners. However, each project is unique and has its own natural length, and you will probably know when you have said everything that you think needs to be said. If in doubt, of course, ask your supervisor if they think you've included everything that's relevant.

You must use 1.5 line spacing and are strongly recommended to use 12 point type. On no account should you use a typeface less than 10 points – it is unreadable! Make sure the left or right hand page margins are big enough so that when the report is bound all the text on a page can be read without stretching open the binding. If possible use double-sided printing.

Pages should be numbered in case they become accidentally separated. Numbering should start from one on the first page after the title page. Chapters should also be numbered and sections and sub-sections should use hierarchal numbering as used in this document.

It must be possible for the whole work to be bound in a single volume, so please use standard A4 paper throughout.

4.4.8 Word Processing Tools

You are free to use any word processor or text processing tools you like. The main advice is to learn how to use your chosen tool effectively, well before you start writing your report. Getting page, chapter, section and sub-section numbering working automatically will save a lot of time, as will getting the contents list generated automatically.

We would encourage you to learn and use LaTeX to format your report. LaTeX is a document markup language and processing system widely used in academia, and has many advantages for writing structured reports and scientific documents. This document is formatted using LaTeX.

[5] http://www.ucl.ac.uk/current-students/guidelines/plagiarism_citation

Keep frequent backups of what you write!

4.5 Style and Grammar

Some punctuation rules:
Full stops (.)
A full stop never has a space before it and is always followed by a space, except when followed by a closing bracket (see below). A full stop used as a decimal point should not have spaces on either side of it.

Commas (,)
A comma never has a space before it and always has a space after it. The only exception is when it is used as a separator in large numbers, such as 5,789,567. Commas – like brackets and hyphens (see below) – separate out subordinate clauses or phrases which merely add information. Within a sentence, you can tell if commas are being used correctly if you can lift out the words involved and have a sentence that still makes sense.

Colons and semicolons (: and ;)
These are 'almost end of sentence' markers that follow the same rules as a full stop. Semicolons are in particular useful when a full stop feels too abrupt but a comma would seem to link two succeeding sentences too strongly. However, many people never use them; if you are unsure about their use it is probably best to stick to full stops and commas.

Slash (/)
A forward slash (used as in 'his/hers') should not have a space on either side of it.

Hyphens (-)
When these are used as a pair within a sentence, in a similar way to a pair of brackets, then both hyphens have a space immediately before and after them (you should really use an en-dash (–) rather than hyphens). However, when a single hyphen is part of a word (as in 'criss-cross') then there are no spaces to either side of it.

Use of brackets
An opening bracket always has a space before it and never has a space after it. Conversely a closing bracket never has a space before it and always has one after it, unless followed by a punctuation mark such as a full stop or comma. Just as in programming languages, in English text brackets have to come in pairs (.....). If the items between a pair of brackets form a sentence, then there must be a full stop immediately preceding the closing bracket. Quotation marks, showing what some speaker actually said, behave just like brackets with regard to full stops.

Apostrophes (')
These are used in two ways, to indicate possession, as in 'John's book' and and in contracted forms, such as 'it'll' as a shortened form of 'it will'. For possessives the rule in relation to singular and plural nouns is:

If the noun is singular, there is an apostrophe followed by an 's', as in the 'The College's buildings...'. This rule is followed whether or not the singular noun ends in an 's', for example in 'The princess's clothes...'.

If the noun is plural, there is an apostrophe after the 's'; for example in writing about Oxford or Cambridge 'the colleges' buildings' is appropriate in referring to the buildings of all the colleges. NOTE: there are some words that denote possession that do not have an apostrophe: his, hers, its, ours, theirs, yours.

No-one ever writes 'her's' but the corresponding misuse of the apostrophe in 'it's' is probably one of the commonest of all grammatical errors. 'it's' always means 'it is', and is an example of a contraction. Contractions such at 'they'll' for 'they will' and 'it's' for 'it is' reflect the rhythms of natural speech. Many people hear a kind of inner voice as they write and it is natural for most of us to write as we speak. Nevertheless, contractions are not recommended in a formal report. Try not to use them.

5 Project Submission

The submission deadline is:

Noon Tuesday 4th September 2012

This is an absolute deadline. Extensions are granted only for unavoidable circumstances that have had a substantial impact on you project, such as serious illness or bereavement. If you have a case for an extension due to such circumstances, talk to your supervisor as soon as possible, and report the extenuating circumstances to the MScCS Programme Director Kevin Bryson. An extenuating circumstances form, with supporting documentation, must be submitted as soon as is practical. An extension must be applied for and there is no guarantee that one will be granted.

If the circumstances warrant it, a short extension (up to a week or so) can be agreed, providing there will be no problems with completing marking on time. A longer extension will not allow enough time for marking to be completed before the exam board, delaying the award of your degree until the following year.

It is unusual for the submission deadline to change; if a change is necessary, this will be announced by email. The opening times of the Reception desk for project submission will also be advertised by email. You are required to check your UCL email regularly (at least once every day, more regularly as deadlines and exams approach); not reading your email is not an acceptable excuse for missing a deadline.

5.1 Binding Your Project Reports

A report must be submitted bound and not loose leaf. The binding should allow the report to lie flat, so that reports can be stacked in a pile safely.

The department strongly recommends using the spiral binding service from the ULU Student Print Centre. See here for more information or visit in person: http://www.ulu.co.uk/content/621903/student_services/print_centre/[6].

The binding used must be strong and not allow pages to easily fall out. Do not use folders or ring binders, or any sort of cover that will not lie flat.

The front and back covers of the report should be robust enough to protect the pages inside. In addition, the front cover must be a transparent plastic sheet so that the full title page of the report is clearly visible. In particular, it must be possible to see your name and the project title without having to open the report. Transparent overhead slide projector sheets make suitable front covers.

5.2 What to Submit

The following should be submitted at the CS Department Reception Desk on the 5th floor of the Malet Place Engineering Building:

- Two complete printed and bound copies of your project report. Make sure that all the pages are in the right order, that everything is included, and that all the format requirements have been met.
- If you are submitting CD/DVDs containing your program code and other relevant files, a copy should be included with each report, securely held inside an envelope or CD cover attached to the inside of the back cover of the report. Label each disc with your name and project title.
- One separate copy of your abstract (descriptive summary) of your project. These should be headed by the project title, your own name and that of your supervisor(s).

In addition, a complete copy of your project report, and a copy of the abstract, should be submitted online using Moodle. Instructions will be emailed in the week before the deadline.

6 Plagiarism

Detailed information about plagiarism can be found on the relevant web pages on the main UCL Web Site [7]. Make sure you read through all the information on these pages and understand the full implications.

In addition, please take note of this statement:
"You are reminded that all work submitted as part of the requirements for any examination or assessment at UCL must be expressed in your own words and incorporate your own ideas and judgements. Plagiarism – that is, the presentation of another person's thoughts or words as though they were your own – must be avoided, with particular care in course-work or essays written in your own time. Direct quotations from the published or unpublished work of others must always

[6] Do *not* use the Thesis binding service to bind your report.
[7] http://www.ucl.ac.uk/current-students/guidelines/plagiarism

be clearly identified as such by being placed inside quotation marks, and a full reference to their source must be provided in the proper form. Remember that a series of short quotations from several different sources, if not clearly identified as such, constitutes plagiarism just as much as does a single unacknowledged long quotation from a single source. Equally, if you summarise another person's ideas or judgements, you must refer to that person in your text, and include the work referred to in your reference list. Failure to observe these rules may result in an allegation of cheating. You should therefore consult your tutor or programme director if you are in any doubt about what is permissible"

As the project is such an important component of your degree, plagiarism in project work is taken very seriously, and when discovered has very severe consequences for the culprit's degree and possibly for their entire future career. It is NOT worth it!

6.1 Get Referencing Correct

The most common form of plagiarism occurs when material created by others is not properly referenced in your project report. The underlying rule is:

The examiner reading your report must be left in no doubt whatsoever which are your words and ideas, and which are the words and ideas of others.

Any content in your report not tagged by a reference is assumed to have been written or created by yourself. Otherwise when you include a reference tag in your report (see 4.4.6) it must be clear what part of your text the tag applies to. In particular, if you are quoting someone else's words then those words need to be clearly identified as written by that person. Usually this is done by putting the words or sentence in double quotes, for example:
Bloggs (2003) states that "A sentence quoted from a paper that someone has written".
or
"A short paragraph defining a concept relevant to your work, quoted from a research paper", Bloggs (2004).

Sometimes it is useful to put the quoted text in italics as well. The same need to reference also applies to diagrams and images included in your report, with the reference tag appearing in the figure or diagram label, for example:
Figure 2 A diagram showing something, source Bloggs (2005)

What is absolutely not acceptable is to copy and paste some text, maybe up to several paragraphs, and just scatter in one or two reference tags with no quotes or other formatting. This fails to identify clearly which are your words and which are being referenced. Even worse is to intermingle some text of your own or to make edits to the copied text so it is not identical to the original but not your own words either.

Don't forget that the need to reference also applies to source code as well, or any other additional material you include with your report.

6.2 Turnitin

You have the option of submitting your project report to Turnitin and it is strongly recommended that you do so. Turnitin will do a detailed analysis of your work to find text that matches entries in its extensive database, giving you an indication of whether you might have an issue with plagiarised content.

Turnitin will always find some matches in your work due to its very rigorous processing algorithms. Many of these matches are likely to be false positives and it is important to recognise this. However, if Turnitin starts finding sentences and paragraphs that match, and you see larger areas of text being highlighted, then you have a problem.

6.3 Why It Happens...

Plagiarism in projects, on those occasions when it has occurred, has more often been a desperate response to a looming deadline rather than a cynical strategy implemented from the very start of the work. This is largely the result of bad planning and time management. Things shouldn't have got so bad that plagiarism seems like the only viable option. Please look at the sections on project time planning. The most important aspect of this is that you meet with your supervisor regularly, ideally each week at a regular time.

If you feel that you are slipping behind, that your project has had to be neglected in favour of coursework, that you have had family or other difficulties that have impacted on your project work, that you don't understand some of what your project entails, then whatever the problem is talk to your supervisor about it. Remember also that for more serious and/or wide-ranging problems, or if you feel you cannot talk to your supervisor, the MSc Computer Science Programme Director is always available to see you.

7 Project Marking

The pass mark for MScCS projects is 50%. A mark of 70% or more is a distinction.

Your project report may be seen by up to three groups of people as part of the assessment process.

1. First and Second Examiners
   All projects are initially given independent marks by two members of UCL CS academic staff (often referred to as the First Examiner and the Second Examiner). The First Examiner is normally your supervisor. If you have had an external supervisor too, your internal (CS department) supervisor may consult with them in deciding on a mark. We try to choose as Second Examiner someone who shares an interest in the topic of the project.

   The following criteria are taken into account:

   - Background Reading (general understanding of the subject area).

- Report Organisation and Structure.
- Clarity of Expression.
- Reasonable and Well-Justified Conclusions (critical appraisal of the work).
- Key Problems Identified and Solved.
- Documentation (user/system manuals, design docs as appropriate).
- Completeness (objectives achieved fully).
- Overall System Design (for analytical projects: Requirements and Objectives Well Understood and Presented).
- Appropriate Use of Data Structures and Algorithms (for analytical projects: Appropriate Use of Design Methodology etc.).
- Appropriate Use of Tools, Libraries, Existing Code etc. (for analytical projects: Overall Quality of Final Design).
- Well Structured and Readable Implementation (for analytical projects: Evaluation and/or Verification of the Design, e.g. by prototyping).
- System Testing and/or Verification (for analytical projects: Practicality of the Design).
- Evaluation and critical analysis of the work done.

You should have a look at the assessment form that the First and Second Examiners must complete (available on the MScCS projects web page). This will help you fulfil the criteria successfully.

After assigning individual marks to your project, the First and Second Examiners discuss your project and allocate an 'agreed mark'. It is usually the case that the two examiners can agree on a mark. In those rare cases where this cannot be achieved the situation is resolved by appointing a Third Examiner or referring the project to the External Examiners (see below).

2. Project Reading Party: Projects are additionally read by two further examiners at a 'Reading Party', where the aim is to ensure that a consistent standard of marking is achieved (don't worry, it is not really a party where the examiners have fun!). If there is disagreement between the Reading Party mark and that which was agreed by First and Second Markers this is also resolved by appointing a third examiner and possibly the External Examiners; serious disagreements which need the involvement of the External Examiners are rare. The Reading Party also serves to moderate the marks of all projects to ensure projects of the similar quality get similar marks.

3. External Examiners: The External Examiners are experienced senior academics from computer science departments outside UCL. They are part of the examining process for all subjects, but their role in relation to projects is to confirm that the assessment process is being run properly, that projects are of appropriate quality and to resolve any disputes between those involved in earlier stages of the project marking cycle. They do not read all the projects, only those where there is a conflict over marking.

The decision of the External Examiners with respect to a project mark is final.