CS5950 MSc Project Manual[i]

This manual provides additional information about aspects of the course.

Contents

1) Supervisors & Personal Logs

This section summarises two key aspects of the project: the role of your project supervisor, and maintaining a personal log.

The Role of the Supervisor

Each of you has been assigned one or more members of academic staff to act as your project supervisor(s). Your supervisor will arrange to meet with you at regular intervals, as part of his or her project topic group. The purpose of these meetings is to allow your supervisor to discuss with you project-related activities, and for you to raise any problems/concerns.

If you encounter any difficulties during the project, you should communicate in the first instance with your supervisor, who will then report any serious problems to the MSc/PgDp Programme Tutor. Any issues that cannot be resolved in this way should be directed to either the MSc/PgDp Programme Tutor or the MSc/PgDp Programme Director.

Maintaining a Personal Log

It is essential that you keep your supervisor informed of progress and meet regularly (normally once a week). It is recommended that after each meeting you provide a summary of the discussion, the achievements of the past week and goals for the following week. You should take this seriously, as it acts as a means by which academic supervisors can monitor your progress, achievements, and any problems.

We also strongly recommend that you keep a personal project log (in a notebook or some other form). You should use your log (or a blog) to keep accurate, timely records of project activities. Remember that in September you may not recall why you made certain decisions back in June!

2) Deliverables

During the project you will be asked to produce deliverables - presentations, demos, and documents which illustrate progress. This section summarises those deliverables (in the table below) and provides some background detail.

Project Plan

When embarking on a project of this scale, it is advisable to draw up a plan, setting out what you intend to do, and when you intend to do it. You can then use this plan to check your progress throughout the year. The plan will also be used by the department to ensure that you have started your project in a sensible manner, and that your are proposing to tackle a project that has sufficient content for an MSc degree, but is not too ambitious.

Your plan should be at most 2 pages of A4, typed, and should contain:

- a brief introduction to the project, giving the background and explaining why it is a worthwhile task;
- the main goals, making it clear which goals are central, and which are optional extras;
- risk assessment, describing likely circumstances in which the project would become unfeasible, and the steps that would be required to recover the project;
- a list of the hardware and software resources required;
- an outline timetable.

Ideally, a plan will include a basic project, which should be fairly straightforward to implement, and which should be completed reasonably early. It should also include more ambitious extensions, or possible developments to be tackled depending on the results of your initial work. A planned project should not be one which only comes together at the end of the period.

You must discuss your plan with your supervisor before you submit it.

Submit a paper copy to your supervisor, and email your supervisor and Nigel Beacham each a copy too.

Poster and Demo

The project poster session and demonstrations will be held the week before your report is due as part of the Project Fair.

Your poster should be no larger than A0 in size. It can take the form of a single A0 (portrait) page or be made up of smaller individual pages such as A4/A3. Poster boards will be provided at the session to display your poster.

The poster should contain just the necessary details, including, your name, email, project title, project aims and objectives, features of system, and future developments. Any images/pictures used in the posters must be copyright free or have authorization from the owner.

You are expected to be here then.

The exact dates will be confirmed nearer the time. See *Project Fair* section for further information.

Code/Archive

The deadline for submission of your program listings and on-line archive of project work is specified on the assessment page. Instructions on how to prepare a project archive appear in the *Submission Guidelines* section.

Demonstration

In the last week before your project submission you will be asked to give a short demonstration of your system. The major purpose of this is to allow you to show how your system works, and to demonstrate its capabilities. This demonstration should not be regarded as an alternative to full description in your dissertation, but as an aid to the proper appreciation by the examiners of your work. The demonstration is not assessed.

Dissertation

The deadline for submission of your dissertation is specified on the *Course Timetable* page of MyAberdeen. You will be required to submit **two** printed copies of your report, and an electronic copy. Details of how to submit the electronic copy will be given nearer the final deadline. Further information about the dissertation can be found in a fuller *dissertation* section. You can hand in your dissertation early, but if you hand it in earlier than 2 weeks before the deadline then you will be expected to justify this, and explain how you have completed all the necessary work early (i.e. your project is expected to take you all your time up to the final submission date).

Late Penalties

The deadline for submission of the project code is 28th August. We strongly encourage you to meet this first deadline, but do not penalise you if you do not, as we feel that the time spent continuing to work on the code (at the inevitable expense of the dissertation) is penalty enough! Late submission of the dissertation does attract a penalty: *Work submitted up to one day late attracts a penalty of 10%, up to one week late, 25%; work handed in more than one week late is marked and returned but is counted as a `No Paper'*.

3) Submission Guidelines

The presentation, project and dissertation overall counts as one-third of the marks for the MSc degree. If you are invited to continue work as an MSc student after the May examinations, your presentation, project work, and dissertation serve to discriminate between the award of an MSc, or the award of a Diploma. You may know what you have done in your project, but the examiners can only judge the quality of work by the description that you give in your dissertation. Make sure that it is worthy of the work you have done.

Re-submission is not normally allowed, so you must get it right first time. You should discuss your dissertation with your supervisor, who will normally require to be shown a plan of the structure of the dissertation and one or two draft chapters some weeks before the submission deadline. Remember that your dissertation has to be of an acceptable standard in terms of content and presentation. However good your basic work has been, however brilliant your ideas have been, the dissertation cannot be accepted if it is not well written and properly presented.

4) Project Fair

In the project fair and demo week (see the course timetable page for the exact date) you will present a demo and running presentation about your project at a 'trade fair'. The session will be for members of staff, any guests, and your colleagues.

Your presentation should highlight the goal of your project and its important features or issues that it addresses. You should be prepared to give an introduction to your project, say why it is interesting or worth doing, discuss your project's issues and difficulties. You will be marked on how well you present your project, and not on the content of the project itself (although we will take into account subjects which are hard to explain). The project fair will contribute 5% of your total project mark.

As you only have limited time to answer people's questions, and only limited space to detail your project, you will need to think clearly about what you can get over to your audience in such a short time. Do not try to put in too much detail. Even though one diagram may be worth a thousand words, don't try to get too much into the presentation. Someone looking at your presentation should go away with a clear idea of what it was about, what issues you were tackling, how you set about it and what degree of success you had. Use the preparation of your presentation as a chance to stand back from the actual day to day work you are doing, and think about the issues you are facing.

When we mark the trade show, we will be asking the following five questions about the poster and how you handle discussion and questions about the project:

- Did they introduce and motivate the project, so that a non-specialist could understand the goals?
- Did they speak clearly, making eye contact with the audience, and hold our attention while the student talked?
- Was the presentation and demo well-prepared and helpful?
- Did they manage their time well, and stay focused on the topic?
- Did they handle the questions well?

5) Dissertation

The dissertation is your chance to tell the examiners what you have done. If you are successful and are awarded the MSc then it will be kept as a Department document and will be available for all to read. You should write it in such a way that it is interesting, readable and understandable by a wide range of people, not just you and your supervisor. Your final submission must contain two copies of the dissertation.

The report should be no longer than about 15,000 with a 50 page limit at font 12 Times.

What.

Say what your project was about, describing both the issues involved and the domain you were working in. The dissertation should be a free standing document; the reader should not need to refer to any other documents, or to the appendices, in order to obtain a good understanding of your work. You should, however, consider putting detailed domain information (for example) in an appendix.

Why.

Say why you did this project, i.e. explain why the project was interesting and relevant to the course.

How.

Say how you carried out the work. This should not be a blow by blow account of every event in the course of your project work, nor a line by line description of the program which implements it, but it should be a description of the methods that you used, the decisions that you took, the results that you achieved and the conclusions that you have drawn. Readers should be given in the dissertation an outline of your system and how it does what it does. This should allow them to understand at a high level how to build a similar system and how to interpret your results. Detailed technical information should be provided in your maintenance manual (see below).

Critical Discussion

Remember to discuss your work and achievements in the light of what other people have done and what you might like to have done yourself. This critical analysis is very important, and it is looked for by the examiners. If you think that you took wrong design decisions, then say so. Developing the ability to appraise work critically, including one's own, is a vital attribute for any scientist. Make sure that you appraise your work in the light of the published literature/related work. Given that you already did this in the technical and research report you wrote last term, you only need to highlight your main findings and remind the reader of that earlier report. In addition, at the end of your project, discuss whether you still consider the techniques used appropriate. If this is not the case, then say so.

Much of the material for your critical analysis will come from your work in evaluating your project. See the *verification, validation and evaluation* sections below for advice on doing this.

Structure of Dissertation

Discuss this with your supervisor. Show him/her a draft structure early on. The dissertation should be divided into chapters with an introduction and a conclusion and as many intermediate chapters as you think appropriate. You should ensure that you outline the background to your project, discuss aims and objectives and so on. The report can also have appendices. Remember that many of your readers will want to be able to understand the broad issues without necessarily going into the fine details. Material which is essential to the deep understanding of your work (for example, detailed accounts of experimental runs), but not essential to the general reader should be put into appendices. Your dissertation *does not* need to include review of the appropriate literature and a summary of relevant earlier work, as you've already done this in the report you submitted last term.

You should include an abstract, acknowledgements, references, and a signed declaration that the work is your own. A typical declaration is given below:

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

User Manual

Where appropriate, you should supply a user manual, so that someone other than yourself, or your supervisor, can sit down at the terminal and run your program and test its performance. It is a good idea to find a friend and get them to test the user manual by trying to run the system from its instructions. The user manual is usually bound in as an appendix of the dissertation.

Maintenance Manual

This should be used to describe the details of your implementation. It should be usable by people wanting:

1. To install the program;
2. To modify your program (e.g. for a different domain);
3. To extend the program;
4. To trace bugs in its execution.

This is an important part of your project documentation, and you should ensure that you include details such as:

- Instructions on how to install the system (From the compressed archive you submit);
- Instructions on how to compile/build the system;
- Hardware/software dependencies (Including libraries, other software packages used);
- Organisation of system files (Including directory structures, location of files within various directories, details of any temporary files created);
- List of source code files (With brief summary of role in system);
- Crucial named constants (Document whereabouts of constants to change when modifying the system, e.g. array size.)
- File pathnames (Explain your philosophy for accessing files of data or parameter values. Are the file pathnames given on command line or in environment variables or embedded in program text or as named constants?)
- Modifications (Include, if relevant, suggestions as to what needs to be altered if the system were to be extended - as described in the future work section of your report).

The maintenance manual is usually bound in as an appendix of the dissertation.

6) Project Code/Archive

You should prepare the following for submission on the date specified on the Course Timetable page in MyAberdeen. Only one copy of each is required.

In the tarball that you submit, you need to include the following;

- Your project directory (i.e., the whole directory containing your build.xml file if using Ant, or the whole project directory if using Eclipse.)
- Your database files from MySQL, either as the database files themselves, or using mysqldump to generate a file of the data.
- Configuration files used in JBoss, or other applications for the project, which point to datasources and anythng else that is needed.

Do **NOT** include any of the following or similar:

- JBoss or Tomcat- instead provide details how to adapt or configure it for your project
- Publically available database sources such as freedb.org. If you downloaded it, then put instructions in the manual about where to obtain it and to install it for your application.
- Third party libraries. This is a judgement call. If the libraries are numerous and small, then include them, if they are small in number, but large in size, then provide installation instructions, and don't submit them.

Program Listings

You should prepare a PDF of your code as detailed on the 'information' page containing a listing of your programs together with a listing of test programs, data and results. Remember that the information must be readable. You do not need to print out your code. Remember to label the listings clearly so the reader knows what is program, what test program, etc. Include a contents list.

Archiving your Project

The Department retains archives of all projects for possible demonstration and future use. As part of your project submission you are asked to present the software you have developed in a format that can be used by others. To this end, you should provide a compressed archive file containing the following:

- A one page description of the program;
- Installation instructions - what does a user have to do to get your program running on their machine? How do you compile the program?
- Dependencies - what other elements does the program require to run? Examples include: hardware requirements, operating systems, software packages, libraries. Include version numbers.
- Source code - make sure that you use relative pathnames rather than absolute ones, so that the system can run without depending on your login name.
- Executable.

Details on how to submit your project archive will be available nearer the deadline.

Note that ensuring that your program can be installed and used is a vital part of software development. The instructions and code you provide will be tested during the project demonstration.

7) Abstracts

The Department maintains a Web page describing completed projects. When you submit your project, you must also submit a one page abstract at

http://www.csd.abdn.ac.uk/teaching/projectabstracts/csd.only/. Fill in the form per your programme.

8) Method of Assessment

Who Marks the Project?

The project is marked by two members of staff, neither of whom will be your supervisor. Both mark the project independently and give their report to a secretary. After both have marked the project, then they meet to sign off their marks. If the two of them came to different marks, then they need to explain how and why they came to an agreed mark. If there is a large disagreement, and the two markers cannot agree on a mark, then the matter will be adjudicated by a third party. The external examiner looks at all the dissertations.

Using two markers, neither of whom was the project supervisor, provides a more robust method of assessment, and is a process used in many universities.

Given that neither project marker is going to be your supervisor, then it is important that you document the choices that you made in the project. Why did you decide to develop it one way, and not another way? What problems did you encounter, and how did you overcome them? Your supervisor would know these issues, but the people marking your project will not. It is important that you alert the marker to these issues by discussing them in your report.

Marking Scheme

The project fair, project and dissertation overall count as one-third of the marks for the MSc degree. The whole project must be large enough; for example, if the programming part is tiny, then it might be possible to produce a well-nigh perfect report; this seems unfair. Therefore, the mark may be scaled by the project size. In general, the marking scheme used is as follows:

- Project Fair (poster, demo): 5%
- Program + maintenance documentation: 47.5%
- Report + user manual: 47.5%

The supervisor report and the markers report as used by staff for marking the project.

What is Being Looked For?

The following points summarise the assessment rubric normally used to evaluate projects. When a project does not comfortably fit this scheme, alternative arrangements are made - see below.

A Program

1. Amount done - the number and difficulty of subgoals achieved. Time required to study preliminary theory or previous work in a continuation project will be taken into account.
2. Style. Does the program conform to the principles of good design -- readable, understandable, structured, modular, well-laid-out, self-explanatory, annotated, etc.?
3. Quality. Use and invention of good algorithms and data structures, making good use of language facilities and operating system.
4. Testing/Evaluation. Evidence of a testing and evaluation strategy, together with results in the form of screen dumps and performance data.

B Description

1. Completeness and exactness. Is the whole enterprise described, are all decisions explained?
2. Presentation and clarity. Is it well organised into chapters; is it interesting and readable; are its aims clearly stated; are its conclusions clearly stated; is it well laid out; are the diagrams and tables well drawn and presented?
3. Critical ability. How well does the writer evaluate the work, for example, in justifying design decisions, explaining wrong decisions, saying what could have been done better, where poor understanding created problems, what were the difficult parts, the main obstacles to be overcome?

C Manuals. Accurate, systematically and clearly presented and easy to use and understand.

D Project Management. The supervisor provides an overview of the project. This is taken into account when marking the project and covers the following:

1. Initiative: How much did the student drive the project? Did they need a lot of hand-holding?
2. Quality: What is the standard of the work compared to that of previous students? Is the project significantly harder or easier than it appears from the report?
3. Obstacles overcome: How good was the student in overcoming unexpected problems, such as failures in hardware, utilities and communications links? How much harder did the project become in consequence? Is this fairly reflected in the report?

9) Intellectual Property Rights

In law at present, software produced by students who are not employees of the University but are often in receipt of a government grant, is treated in the same way as if they had written a novel at home in the evenings. In consequence, where you are actually producing software as part of your regular daytime activity under the direction of a University supervisor, and using University computers and software tools/compilers, we need to establish clear title to the IPR. Hence, since 1994, the University Court takes the act of registration to include assignment of IPR to the University.

The reason for this is not so much that the University can get lots of extra income, which is unlikely, but to defend its ability to do research. Research groups must be able to control the copyright of software they are developing, in order that future staff and students can continue to use and adapt it without threat of legal action or predatory pricing. One should also remember that although your piece of software seems very valuable to you, it usually only works as part of a larger system built by several people either as a team or in succession.

Note that you have not signed away your birthright. The agreement only concerns IPR ``arising in connection with or incidental to your course and studies here''. You have not signed away rights to software developed at home for your own purposes on your PC. Instead you have entrusted the University with your claim to the IPR developed here in exactly the same way that University paid research and teaching staff have to do, so that any IPR can be safeguarded, and any financial reward shared out on a fair and equal basis.

In the case of students undertaking industrial projects as part of their course, we ask them to sign a form which also covers arrangements for examining reports and publication rights. The form assigns IPR to the University, and not the collaborating body. If the University makes money from the software, this will be shared equitably with you. For further information, contact the Department Industrial Liaison Officer (Dr Jeff Pan).

10) Testing and Evaluation

The goal of software evaluation is to assess the worth of a software system. Evaluation is a crucial stage in the software development process, both for commercial software, and for software developed as part of an academic exercise. For commercial software, the customer will want to evaluate the software to determine how well it does its advertised job; for academic software, evaluation allows students and staff to judge the success (or otherwise) of the software project. Evaluation is related to software verification, validating and testing.

Verification

Software verification involves answering the question: ``Does the software work properly?'' Verification can be performed by a process of formal analysis, resulting in a mathematical proof that the software is correct. However, this tends to be feasible only for exceptionally well-defined software projects; most commonly, verification is performed by testing the software on a range of test cases. A test case consists of two parts: the input to the system, and the desired correct output. When the test cases are run on the software, if the actual output matches the desired output in every case, the software passes the verification test. In some application domains, there is no such thing as a correct output; instead, there may be a number of outputs, with different degrees of acceptability. In this kind of domain, there is some ``fuzziness'' in deciding if the software passes the verification test. Many application domains have this feature; for example, intelligent systems are often built in domains where human judgement and rules-of-thumb are factors in reaching a conclusion.

In order for the results of the verification to be meaningful, the set of test cases must be representative of the entire range of cases that the software needs to process. In general, there is no formal procedure to ensure this, so software testing is something of an art! However, it is generally recommended that two kinds of test case are chosen: functional test cases, based on the tasks that the system is intended to perform; and structural test cases, based on the internal structure of the software. In other words, functional test cases are created by considering the software as a ``black box'', and there will be at least one test case for each different task the software is intended to perform. Structural testing considers the system as a ``glass box'', and there will be at least one test case for each different path through the software code.

Validation

Software validation involves answering the question: ``Does the software do what the customer wants?'' Note that it is perfectly possible for a piece of software to pass a series of verification tests, and still not do what the customer wants; this usually happens because the customer did not properly communicate all their requirements to the software developers. (Conversely, it is unlikely that software that fails verification testing will meet the customer's requirements, except by accident!) Verification can therefore be considered a part of validation. In addition to performing thorough testing, validation will involve regular trials of the system with the customers and representative users, to ensure that any missing or misunderstood requirements are revealed and incorporated.

Evaluation

Software evaluation involves answering the question: ``Does the software do something of sufficient value?'' Validation and verification are a part of evaluation; software that doesn't work properly and do what the customer wants is unlikely to be of value (again, except by accident). However, evaluation is broader in scope that validation and verification. Typically, evaluation entails demonstrating that the system has a positive benefit to its users, allowing them to do something they couldn't do before, or allowing them to do something better or faster than they could before. The system may work correctly, and may do what the customer wanted, but may still not be of value to users.

Evaluation typically involves running experiments with the software, in real or simulated operational conditions. Sometimes, formal experiments are run, where one test group of users carries out tasks using the software system, and a control group carries out the same tasks by the pre-existing method (perhaps manually, or using a rival system). The software is seen to be of value where the test group out-performs the control group. Often, formal experiments are not conducted, but users are given the system to ``try out'' for a period of time, and report back. For software systems where a user-friendly interface is an important feature, users are often put under observation while they perform tasks using the software, and any difficulties they have are recorded in order to improve the software.

Guidelines for Evaluating Projects

- Develop a clear statement of the customer's requirements, and use this as a basis to create a representative set of functional test cases.
- Meet with the customer regularly throughout the development process, and ideally demonstrate prototype systems, in order to ensure that the requirements are complete and well-understood.
- Examine the code of the software, and ensure that structural test cases are created to test all the major paths through the code.
- Get representatives of the user population to test the software, and systematically record their feedback; avoid interfering with their natural use of the software.
- If the software is designed to improve on some pre-existing system (including a manual process), gather as much data as possible to compare objectively the two systems.
- Above all, be objective: don't be afraid to admit that the software is flawed; the ability to critique one's own work is a necessary skill for high-quality software engineering.

[i] Date created: 11th June 2013, updated: 11th June 2013