Coursework commentary 2017–18

CO3320 Preliminary Project Report

Introduction

The primary purpose of the Preliminary Project Report (PPR) is to encourage candidates to begin thinking about, and working on, their projects at an early stage of the year.

The general standard of this year's PPRs was good, with an average mark of approximately 56%. The pass rate was very high, at approximately 96%.

Individual feedback is provided by staff at Goldsmiths to each student within a few weeks of the submission deadline. The more information a PPR contains about work done to date, problems encountered, and future plans, the easier it is for examiners to provide helpful feedback.

Please note that the PPR is expected to conform to a standard structure, as specified in the current edition of the CO3320 Project Subject Guide. The Subject Guide specifies that the report should be around 2,000–4,000 words. The examiners are looking for evidence of a candidate's ability to write clearly and concisely, and their ability to judge what information should be included and what is irrelevant. Around 15–20 pages is usually about right for the PPR. Note that the PPR must be submitted as a PDF document (not Word or any other format) – this is clearly stated on the online submission page on the VLE, and yet every year we get a handful of candidates who ignore this!

Some of the common weaknesses seen in this year's PPRs were:

- failure to identify an appropriate question to address or aim to achieve
- inadequate literature review (including poor referencing and citation)
- poor software design process
- poor project plan (including lack of thought about testing and evaluation)

Each of these topics is discussed below. Note that these three issues come up every year in the PPRs. If you are reading this commentary at an early stage in thinking about your project, and take time to ensure that each of these issues is addressed, then you should be in good shape for producing your PPR.

Failure to identify an appropriate question to address or aim to achieve

Ideally, the examiners are looking for a project to address a specific problem by following the structure of an academic research project: identifying a specific question to be addressed, proposing a means of answering that question (which may entail proposing a solution to an identified problem), performing some sort of experimental data collection relating to the proposed means of answering the question, analysing the collected data, and drawing conclusions from the analysis which relate back to the original research question.

Projects which merely involve the implementation of a piece of software or website, with no research question driving the development, will struggle to achieve the highest grades, although such projects can receive good marks if approached in the right way. In order for such projects to be acceptable,

they must demonstrate the application of solid software development practice (including requirements gathering, design, implementation, testing and evaluation). Even a project which is, on the face of it, a straightforward software development task, can be cast as an academic research project if appropriate questions can be addressed (*e.g.* Can novel feature *X* improve some aspect of a business process? Can novel user interface feature *Y* improve customer satisfaction of the system?). The more specific a question that can be framed, and the more specific the means of analysis, the easier it will be to provide a definitive answer to it in the project.

A small number of PPRs seemed to offer no original contribution from the candidate. The nature of these projects generally involved the candidate "finding out about" a subject, and reporting what had been found. Projects that involve no software implementation are particularly prone to this weakness. The examiners are looking for a project that shows you putting the techniques and knowledge you have learned during your studies into practice; and for a good project, the examiners are really looking for some original idea and/or for something beyond what has been read in books or other sources. This "extra" contribution might be very small, but there should be something beyond just reading and reporting. For projects with no software development, it is particularly important that what is written is derived from reliable sources of academic knowledge (e.g., journal articles and conference proceedings) and, where appropriate, from relevant texts on industry best practice. Furthermore, it is essential that projects of this nature have a strong CIS/CC flavour and have some relation to the content of at least one of the courses you have studied.

Having identified a suitable project area, some students still failed to clearly set out their aims and objectives. The **aims** of the project describe the broad overall purpose and desired outcomes of the work; the **objectives** describe the concrete steps you intend to take to achieve your aims. The more specific you are about your aims and objectives at the start of the project, the easier it will be to formulate an appropriate plan of work for conducting the project. On the other hand, some candidates set out *too much low-level detail* in their aims and objectives: projects involving software development are particularly prone to this. It is usually the case that the low-level details of what you intend to do should be decided upon only after a literature review and/or requirements gathering process. Try to keep an open mind at the start of the project about what might be the most appropriate way to achieve your aims. Coming to a decision on the most appropriate design is a part of the project itself, and usually not something that should be decided in advance.

Inadequate literature review (including poor referencing and citation)

The literature review is an important aspect of your project, and the PPR should include a summary of the literature you have reviewed to date. The literature review serves to put your project in the context of what other people are doing in the same area. By having a good knowledge of what other people have done, you are less likely to "reinvent the wheel", you might avoid approaches that other people have tried and failed to make work, and you might find inspiration for how to do things better. A weakness of some PPRs was the use of references to websites rather than academic sources such as journals or conference papers. The problem with websites is that they are not peer-reviewed, and the information they contain is not necessarily reliable. If you are using information obtained from websites, consider how reliable it is, and consider including some discussion about the reliability of your sources.

As stated in the previous section, an important role of the literature review is to inform the design of your project. If your project involves the development of a software tool or app, be sure to do the literature review before you get too far into the software design. Think of it as a process of looking for answers to questions you may have about how to design the software and experiments. The literature review is only useful insofar as it helps you in conducting your project. If you are developing software, your review should also include some discussion of what (if any) similar/related programs/apps are already available on the market, what are their strengths and weaknesses, and how your proposed product will differ from these.

Some candidates submit literature reviews that come across as general tutorials in the subject area of their project. This is not what the examiners are looking for. The review should cover material that specifically helps you in achieving the aims of your project; to give an example, if your project is in the area of neural networks for stock market prediction, the review should cover previous work on this topic (e.g., what has been done, what techniques were used, what data was used, what results were obtained, what were the strengths and weaknesses of these previous studies, etc.) – it should **not** be a general tutorial on back propagation or other general neural network topics.

A common issue concerning references and citation is that some candidates included a Reference List at the end of their PPR, but did not indicate in the main text of their PPR which references were relevant where. This is done by using a **citation**, which is a short marker in the main text (e.g. "(Taylor, 2012)") which denotes an item in the Reference List. Even more importantly, some candidates copied sentences from other authors' work without the proper use of quotation marks and citations. It is perfectly acceptable to copy text from another source (within reason), but only if you clearly indicate, through the use of **quotation marks** and a **citation**, where you have obtained the text from. Failure to do this raises the suspicion of plagiarism — trying to present someone else's work as your own — whether intentional or otherwise. There are severe consequences for plagiarism, so be very sure you know how to use quotations, citations and references appropriately.

On the other hand, try to avoid over-using long quotations from other papers where possible. In terms of writing style and the narrative flow of your PPR, it is usually better to describe other work in your own words – but a citation to the original paper is still required, to indicate what work you are describing! Even with proper citation, care must be taken not to over-use quotations from other sources. The literature reviews of some PPRs consisted of little more than a list of quotes from other sources, with little or no original text from the candidate. Such reviews are tedious to read, have little narrative structure or flow, and are generally of very little specific value. A good literature review involves explaining the relevance to the project of what has been done before, and how it will influence how the current project will be undertaken. The literature review can be useful to help justify the choices you make in your project, including choice of research question, experimental design and analysis techniques. It therefore requires significant input, and insight, from the student, not just a list of quotes from other authors.

Poor software design process

To reiterate what was discussed in the previous section, for projects involving software design, the important design features of the software should be explained with justification for the choices made. Ideally,

this justification for making a decision about the design will be by making reference to existing theory or research as identified in the literature review, although in some cases you may have reasons for doing things differently. The important thing is that you should explain why you made the decisions you did. A fairly common weakness in the PPRs is to set out a proposed design without any justification or indication that the candidate has considered other approaches.

Poor project plan (including lack to thought about testing and evaluation)

It is important that the project plan is realistic and achievable within the time available. Some candidates presented project plans that were far too ambitious. It is better to submit a smaller, but complete, project than to submit a more ambitious, but incomplete, one. Plans for further extending the work can always be discussed at the end of the final project report if desired. Drawing up a realistic project plan is *really important*: this year, several candidates submitted final project reports that started off very well, but which included only very scant (or even completely missing) later chapters (e.g., Results, Discussion and Conclusions). In order to pass the project, the final report must describe how *all* stages of the project were conducted at a satisfactory level.

In contrast, a few candidates submitted very brief PPRs with very light project plans. Remember that the final project report is supposed to represent the culmination of at least 300 hours of intense, focussed study. Furthermore, the less information there is in the PPR about exactly how you intend to do things, the less detailed feedback the examiners are able to give you about your planned approach.

A common weakness in project plans is that they are too high-level. A plan that consists only of tasks such as "literature review", "software development" and "testing" is not very helpful for keeping you on track. Try to break down each task into smaller sub-tasks (each one ideally being no longer than a couple of weeks), and add specific start and end dates to each item. It is useful to present the plan as a diagram, so it is easy to check your progress as you go.

For projects where software development is a major part of the work, the examiners will be looking for evidence that you have followed a structured software development methodology, including requirements gathering, design (use cases, wireframes, etc.), implementation, testing/bug fixing, and evaluation. Candidates pursuing such projects should think carefully about exactly what needs to be done for each of these, how you will do it, and how long it will take. In your PPR you should discuss the current state of your development plans – a surprising number of PPRs for projects involving software development do not even mention what language(s)/tools/libraries the candidate intends to use to develop the system. Candidates will often be faced with the question of whether to use an off-the-shelf tool or library to achieve a certain goal, or whether to implement the system from scratch. Either approach is acceptable in principle, and you should think about what would be better for you – you'll need to balance the cost of learning to use an existing tool against the potential time saved once you have learned it (on the other hand, you'll almost certainly learn more about the details of a technique if you implement the code yourself). If you are faced with such a decision, be sure to properly document the alternatives, and justify your final decision, in your PPR/final report.

In addition to identifying a specific question to be addressed, it is also very important to be clear right at the beginning of your project exactly how you are going to **evaluate** the success of your work. A common failing in the PPRs was a lack of a clear plan for evaluation.

Think about what question(s) you want to answer, then think carefully about the criteria for success by addressing the following more specific issues:

- How will you test the system?
- What results data will you collect?
- How will you analyse the results?
- How will you judge the significance of the results (*e.g.* what will you compare them against)?

For projects that involve neural networks or other machine learning techniques, think carefully about what data the system will learn, whether you have access to a suitable source of data, whether the data is reliable (perhaps you need to check it for consistency first, or pre-process it in some way), and whether you will need to partition your data in some way (e.g. into training and test sets, and perhaps validation sets too).

For projects which involve developing software for a group of intended users, be sure to include in your project plan a process of stakeholder consultation at the start of the project to establish their requirements and their views on your proposed solutions. There are very few cases where such stakeholder consultation will not be appropriate. Even if such a consultation is not conducted, all software development projects should include some kind of requirements gathering process, and this should be reported in your final report (and in the PPR if possible). That is to say, the design should be justified (e.g., from a theoretical perspective or from the literature review of previous work, if not from stakeholder consultation), and should not just come out of the blue with no explanation of how, or why, it is designed that way.

For software development projects, in addition to stakeholder consultation at the design stage, it is also important to include some element of stakeholder evaluation after the system has been developed. For such projects, care should be taken at an early stage to decide who will evaluate the end product, and how such evaluation will be carried out. It may be that different sorts of evaluation are appropriate for different groups of stakeholders. Without seeking stakeholder evaluation and analysing the results, it can be hard to evaluate whether the project has succeeded or failed in its goals.

For projects that involve questionnaires and user feedback, many of the PPRs showed a lack of thought about exactly what would be required. Be sure to think about questions such as:

- Who will you ask? (Is there a single group of stakeholders, or multiple groups? How can you select the most representative sample possible from each group?)
- How many people do you need to include in order to generate reliable results? (Think about what is required for statistical significance, although sometimes practical matters may prevent you from including as many people as you would like.)
- What will you ask?
- How will you analyse the data?
- How long will all of this take?

A common failing with questionnaire design in projects is that it is unclear how the answers to the questions will help the student in achieving their aims. Careful attention must be given to the design of questionnaires such that they genuinely contribute to the successful completion of the project. It is all too common to see questionnaires which end up being a waste of time in terms of their contribution to achieving the project's aims.

In data collection and analysis, as in all other aspects of the project, the more detailed and specific you can be at an early stage of the project about exactly what you are going to do, how you are going to do it, and how long it will take, the higher the chance of you completing a successful project on time.

Even if you have done a good job of drawing up a detailed project plan at the start of the project, and attempted to foresee potential problems that might arise and thought about how you might deal with them, it is not unusual for unexpected delays to occur. It is therefore essential to review your plan regularly, and be prepared to adjust it if necessary. It is often useful to rate each feature you are thinking about including as *must have / nice to have / could have*. You should concentrate on the most important features first, and get those fully implemented and tested, and only implement other features if and when you have the time. Some candidates submit project plans in their PPRs which do not match the reality of the progress they have described in the PPR (e.g. the plan might show that system implementation should be well under-way in mid-January, and yet it appears they have not started any implementation). Don't just ignore slippages from the project plan – if you are in this situation, you should talk about them in your PPR and present a realistic plan for how you intend to proceed.

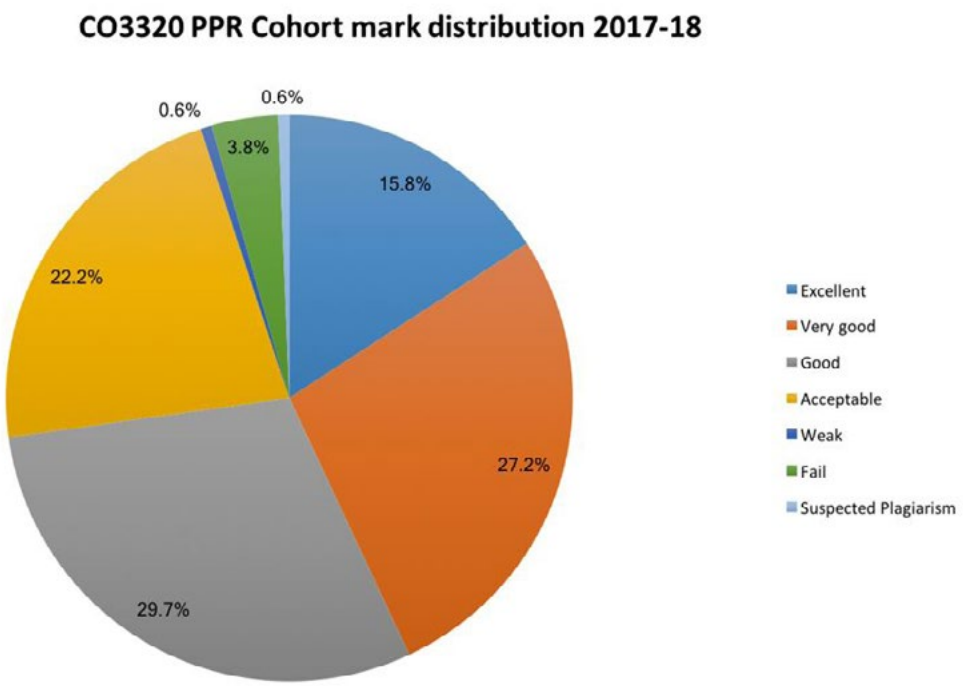
For projects involving the development of mobile or web apps, you should think carefully about what you will submit with your final report in May. The examiners need to see the code you have written, and, ideally, be able to run it. But they can only give you credit for work submitted before the deadline. Giving a URL where the examiners can see your live site may seem useful, but the examiners cannot verify when marking whether what is on the site was working before the submission deadline. On the other hand, the examiners will not have time to set up a web server and other supporting systems to run your submitted code on their own machines. A sensible approach is to include the following in your submission: (1) the source code you developed, (2) a full description in your project report, with screen-shots, of the final system, (3) a video run-through demonstrating all important aspects of the system, and (4) [optionally] a URL to a live web-server where the examiners can interact with your site (but only to experience what you have already fully documented in parts (1), (2) and (3)).

Finally, some candidates did not allocate time in their project plan for writing up their final project report! This is obviously an essential part of the project, and will likely take longer than you expect. It is advisable to write sections of the final report as you go, rather than leaving the writing of the whole report until a few weeks before the submission deadline.

In general, the 2017–18 PPRs spanned a very wide range of standards, from the weak to the truly outstanding. The preceding comments have highlighted some of the common problems. Further advice on how to produce a good PPR can be obtained in the following ways:

- Read the CO3320 Project Subject Guide.
- Look at examples of good projects from previous years in the Project Library section of the VLE.
- Discuss problems and questions with fellow students on the Discussion forum of the CO3320 page on the VLE.

Below is a pie chart showing the distribution of interim grades (see Appendix E in the Regulations for Assessment criteria) for the PPR in 2017–18.


CO3320 PPR Cohort mark distribution 2017-18
0.6%
0.6%
3.8%
15.8%
22.2%
27.2%
29.7%
Excellent
Very good
Good
Acceptable
Weak
Fail
Suspected Plagiarism