



**Swansea University Computer Science**  
**MEng Level M Project Manual**  
***CS-M04 and CS-M14***

**2011-2012**

*Dr Parisa Eslambolchilar*

[csparisa@swansea.ac.uk](mailto:csparisa@swansea.ac.uk)

*Office Telephone: 01792 602658*

Version 2.0

## **Version History**

1.0 October 2009

2.0 March 2011 – updated the text on *Industrial Project – Project Selection* to reflect the fact that this year (2011) Level 3 and 4 project selection will be combined.

## Introduction

This is a manual for students undertaking the modules CS-M04 Group Project and CS-M14 Industrial Project in Level M of the MEng in Computing. It is intended to give guidance on how you should undertake these projects, and how they will be assessed. You should however always consult your supervisor about important decisions and choices. The manual for these projects is combined because both are compulsory for students on the MEng Computing scheme, and because the structure of both projects is similar. It is helpful to have first read the *Swansea University Computer Science Third Year Project Handbook* (M Roggenbach 2009), which covers many of the fundamentals of undertaking, managing and writing about projects. *However there are substantial differences between Level 3 and Level M projects.* This manual concentrates mainly on those differences, and specific issues concerning the actual deliverables.

## Level M Projects vs Level 3 Projects

The Level 3 project is an individual project that is *curiosity-driven*. You are expected to produce a substantial piece of work - but it does not have to be what you initially set out to do. It just has to be a significant body of high-quality work. In contrast, CS-M04 and CS-M14 are intended to more closely model the real software development process. In particular, there is an emphasis on *delivery*. That is, you will be assessed on *how well you have met the initial stated goals*. This has significant consequences for how the projects are assessed, and how you should undertake them.

### Project Milestones - Overview

Both CS-M04 and CS-M14 have three milestones:

1. **Initial Requirements, Specification and Methodology.** These documents describe what you intend to do and how, as well as the risks you expect to face and what you have done about them.
2. **Interim Report.** A description of the progress you have made on the project and any issues that have arisen.
3. **Presentation, Manuals, Testing and Reflection.** A presentation on the project, appropriate user and testing documentation, and a reflective account of the project. The last of these is the only one that you would not be expected to produce in a real project.

## Industrial Project - Project Selection

The Industrial project is chosen in a similar way to Level 3 projects from a list of suggested projects (some industry-related). This year (2011) we will be using a combined list for Level 3 and the Level 4 industrial projects. You will be asked to select a project in the same way you did last year. Students can also propose their own projects in the same way as at Level 3 (they need to get the agreement of a member of staff to supervise it). Unlike your project choice in Level 3, there are important restrictions on the projects you can choose or propose.

1. They must be practical and involve the production of a piece of software.
2. They must have well-defined goals, or be amendable to have well-defined goals.

As an example of a project with well defined goals, consider a project: 'Write a software application to manage undergraduate admissions in computer science'. This project has a clear goal and is practical - the software to be produced is not clearly specified yet, but that is OK (and forms the first part of the project). However, consider a project: 'X has written a new paper on something very interesting - why don't you see what you can do with it?'

This *does not* have well-defined goals. However, it *could* still be a suitable project - *provided by the time the specification must be written the precise goals of the project have been finalised*. This depends on the exact project, the supervisor, and you. However, it's obviously a risk - and as discussed below you need to limit risks in MEng projects - and so such projects must be considered very carefully.

## Group Project - Project Selection

The group project will normally be undertaken in groups of three or four (exceptionally, groups of between two and five). Unlike the Industrial Project, you do not get to choose your project. It will be allocated to you by the CS-M04 coordinator.

### Group Project Assessment

Participants in a group project may not contribute equally. In order to account for this the following mechanism is used.

- Towards the end of the project (typically in Semester 2, Week 10) the project co-ordinator will approach each group member and ask if they are happy for all group members to get the same mark.
- If all agree that the same mark is appropriate, then the mark each person is awarded is the same as the overall group mark.
- If not, then each member is asked to state privately what percentage of the work each group member (including themselves) did.
- This approach prevents individuals 'breaking ranks'. That is, it is not possible for an individual to agree all marks should be divided equally, while actually claiming they did more of the work, without the knowledge of the other group members.
- This information is then used to calculate a mark for a group member as follows:  
$$\text{mark} = gm * aps * n$$
where  $gm$  is the group mark,  $aps$  is the average of the percentage score for the group member, and  $n$  is the number in the group.
- In either case, all group members will be informed which method is being used.

## Marking Forms

Supervisors and Second Markers will use the forms attached in the appendix to assess your work. (They also obviously have access to this manual).

## Project Structure - Milestones and Deliverables

Both the Group and Industrial Projects are divided up into three sections - *Milestones 1 – 3*. Each Milestone has associated *Deliverables*.

### Industrial Project CS-M14

Milestone	Deliverable	Weighting	Due
Milestone 1	Methodology and Requirements Document	15%	Before January exams- Friday 6th of Jan
	Specification Document	10%	
Milestone 2	Interim Report	10%	Semester 2 Week 9 Friday 30th of March
Milestone 3	Poster Presentation	10%	Semester 2 Week 10 - Friday 27th of April
	User Manual	10%	
	Design Document	15%	
	Testing Document	10%	
	Narrative and Reflective Account	20%	

The milestones and deliverables for the Group Project are very similar:

**Group Project CS-M04**

Milestone	Deliverable	Weighting	Due
Milestone 1	Team Structure and Methodology Document	10%	End of Term 1 Friday 16th of Dec
	Requirements Document	10%	
	Specification Document	10%	
Milestone 2	Interim Report	10%	Semester 2 Week 5 Friday 2nd of March
Milestone 3	Poster Presentation	10%	Semester 2 Week 10 - Friday 27th of April
	User Manual	10%	
	Design Document	10%	
	Testing Document	10%	
	Narrative and Reflective Account	20%	

Because the structure is very similar, we will combine guidance for documents that appear in both projects.

**Be Flexible**

This manual gives advice on the contents of documents in a 'typical' project. Not all projects are 'typical' and most have at least some 'non-typical' aspects. In those cases, you will have to interpret this guidance carefully (and take advice from your supervisor). Do not just slavishly follow the advice here if you think it is not completely appropriate in your case. For example, although most projects are about producing *applications*, some projects are about producing other kinds of software (for example an API).

## **Industrial Project - Methodology and Requirements Document**

This document has two roles. First to explain the software development methodology chosen, and second to state the project requirements.

**Methodology**

You should research available software development methodologies, choose the one they believe is best for your project, and explain why. (To do this, you will almost certainly have to briefly describe the methodologies you have rejected.) A few points to note:

1. You need to factor in both your project and the other things you are doing in your choice. For example, are there times in which you will be extremely busy with something else?

2. If you are working with an external company, you need to consider that as well – for example, you may need to choose a methodology that allows you to create prototypes (perhaps partial) that demonstrate features and enable the company to give you feedback (and possibly request modifications). This is particularly true if the company you are working with is not in the software business, and therefore may have a limited understanding of how software is built.
3. You may need to modify the methodology you choose because you will be working alone – most methodologies assume a team.
4. Don't forget you are allowed to have a preference – if after eliminating non viable options you are left with more than one choice, pick the one you prefer (but make sure you say so).
5. You should include a timetable for weekly activity and progress on the project. It is unlikely that a simple linear list will be appropriate with a project at this level as the tasks are almost certain to overlap – you should use, for example, a Gantt chart.
6. Although not formally part of 'methodology' this document should address risk management – identifying risks, assigning them likelihoods and impacts, prioritising them and explaining how the more likely/high impact risks will be minimised or designed out. You should discuss the issue of risk management with both your supervisor and the co-ordinators for the appropriate project (CS-M04 and CS-M14). These co-ordinators are currently Dr Eslambolchilar.

### **Risk Management**

The precise risks faced by your project can vary, but the *typical* ones that have occurred in Level M MEng projects are as follows.

1. *Failure to adequately judge the time and resources required.* This risk should be reduced by your experience in your Level 3 project. It is also the reason why the documents stating what you intend to do are not due until the Christmas period: giving you time to carefully assess and plan.
2. *Failure to adequately learn a required new technology.* This risk can be reduced by spending time mastering new technologies before Christmas, by making sure (if possible) that there are alternatives available, and by choosing an appropriate balance of technologies that are new to you, and ones that are already familiar.
3. *Illness.* You should plan for the possibility of losing some time to (minor) illness. The impact of serious illness on a CS-M14 (Industrial) project is hard to reduce, though it is fortunately very unlikely. *However*, you can and should plan for the possibility of a team member being partially or completely unavailable in CS-M04 (although this again can be difficult if you are in a small team).
4. *Non-contributing team member.* A group project team member may not contribute for reasons other than illness. In this case, the same contingency plans for a team member being ill should be effective.
5. *Feature creep.* That is, adding new features not in the original requirements. This typically occurs there are multiple CS-M04 groups, and each is undertaking the same project. Inevitably, each group then discovers that the others have included things they have not, and they all proceed to add them. The situation may then become 'competitive' and further new features are added. The risk is that the workload becomes unsustainable, and the project is not achievable. The CS-M04 co-ordinator(s) have a role in preventing this. But groups should be confident that what they propose to do is good: even if other groups have other, different features, there will be things *you* have that they do not!

## Requirements

This manual does not intend to try to teach any Software Engineering. However, the requirements part of this documents sets out what the system you are developing is intended to *achieve* – not what is supposed to *do* (that’s the specification) or *how* (the design). The initial project statement you received from your ‘client’ is probably quite vague – you will need to make it more precise. To do this, you are almost certainly going to have to discuss it with your ‘client’ – which may be your supervisor, or may be an external company.

You should not use a ‘narrative’ style to state your requirements. Instead, separate the requirements into small, well defined parts, and use a list style to write them down. Each separate requirement should have a code – some appropriate number and letter combination for example – that allows each requirement to be identified. (For example, suppose you are writing a zoo administration system – you might use codes beginning with A for requirements to do with animals, E for enclosures, F for food, S for staff and so on). You should also separate *functional* and *non-functional* requirements (for example, performance constraints).

Supporting narrative text should be kept to the minimum needed to make the document readable. So for example, you should not explain what a requirement is, or the history and literature of requirements and so on. In this, the Level M projects differ substantially from the Level 3 project: *they are not scientific documents*. A useful approach when writing requirements and specifications is to consider them to be *legally-binding* documents (which they probably would be in a real project). Therefore, you are aiming for *precision* in your descriptions of what you propose to do. Fig. 1 shows a possible format.

Code	Requirement
RPTREQ1	The application is required to provide a user interface within which the user can navigate around the reporting folder structure.
RPTREQ2	The application must allow the user to open a report (.rdl file) or a report project (.rptproj) with Visual Studio.
RPTREQ3	The application must ensure that a project and/or report which has already been checked out from the server by another user cannot be checked out again until it has been checked back in.
RPTREQ4	The application must distinctively show users which reports or projects are currently checked out by another user.
RPTREQ5	The application must allow reports to be executed inside the application.

Fig. 1. A possible format for requirements.

## Group Project - Team Structure and Methodology Document

This document is very similar to the Methodology part of the Methodology and Requirements document for the Industrial Project. You should refer to the relevant parts of the the text above. The main difference is that it should not be necessary to modify your selected methodology because you are working alone (you obviously are not). Also, planning is more complex because there are more people involved. You should also pay attention to *quality control*. Even with the most conscientious people, it is a dangerous

mistake to put part of the project in the hands of one person. There should always be at least two people involved: either directly, or with one responsible for the work and another for reviewing and monitoring it.

## Group Project - Requirements Document

This document is very similar to the Requirements portion of the Methodology and Requirements document for the individual Industrial Project, and so similar advice applies (see above).

## Specification Document

The specification document should describe what the system should do. The precise definition of ‘what a system does’ depends on the type of the project. For example, it might be a description of the functionality accessible via the user interface (for an application), or it might be the public methods (names, arguments, results) for an API. As with the requirements, you are encouraged to write specifications as small, well-defined and labelled parts. You should also state which requirement each specification component addresses. To make it completely clear that your complete set of functional requirements is addressed, you are also encouraged to include a table of the functional requirements, together with each associated specification – this makes it easy to immediately see that all functional requirements are addressed by the specification. As with the requirements, supporting narrative text should be kept to a minimum. Figs. 2 and 3 show a possible format for specification and non-functional requirements.

ID	Description
RPTSPEC1	The application will allow the user to select folders at any level within the ‘Development’ folder hierarchy.
RPTSPEC2	The application will display the contents of the folder selected by the user.
RPTSPEC3	The application will allow the user to select a single or an arbitrary group of files within a selected folder.
RPTSPEC4	The application will be aware of the full file path of file displayed within a folder.
RPTSPEC5	The application will be capable of opening the Visual Studio IDE.
RPTSPEC6	The application will provide an interface to allow the location of the Visual Studio executable file (devenv.exe), to be entered or amended.
RPTSPEC7	The application will pass the selected .rdl or .rptproj file to devenv.exe as an argument.

**Fig. 2. Itemised specification for one part of a project.**

ID	Description
NFSPEC1	The application will use the font Tahoma and the colour represented by the hexadecimal RGB value #004da3, where appropriate
NFSPEC2	The application will contain an online help system easily accessible by the user from within the application.
NFSPEC3	The application will be supplied to the client with a physical user manual describing how to effectively operate all features of the system.
NFSPEC4	The application will be engineered in a such a way that any crash of the application has no effect on the remainder of the users computer.
NFSPEC5	The application will require the lowest possible level of user permissions necessary for its correct operation so as to minimise the risk to the user of operating it.
NFSPEC6	The application will be designed in such a way that any future enhancements are easy to make.

**Fig. 3. Non-functional requirements.**

Fig. 4 is an example of a table cross referencing requirements (from Fig. 1) with specifications (Fig. 2).

Requirement ID	Specification ID's
RPTREQ1	RPTSPEC1, RPTSPEC2, RPTSPEC3
RPTREQ2	RPTSPEC4, RPTSPEC5, RPTSPEC6, RPTSPEC7, RPTSPEC8, RPTSPEC9
RPTREQ3	RPTSPEC10, RPTSPEC11, RPTSPEC12
RPTREQ4	RPTSPEC13, RPTSPEC14, RPTSPEC15

**Fig. 4. Cross referencing requirements and specifications.**

## Interim Report

For both Industrial and Group projects, the Interim Report is about progress on the project. You should report progress made in comparison with your plan in the Methodology and Requirements Document or Team Structure Document as appropriate. If the project is behind schedule – which is quite common – you must explain what you are going to do to correct the situation. *Only under exceptional circumstances should you modify your requirements/specification to do this.* Remember at all times you are going to be assessed on how well you deliver on your initial targets. If you *have* to modify your requirements/specification, you should think – and explain – carefully how you are going to do this. You should choose aspects/features to omit that have the least impact on the project overall. (There are some other cases where changes *may* be appropriate – for example when working with an industrial collaborator. Special circumstances such as this should be discussed carefully with your supervisor.)

## Poster Presentation

The purpose of, and facilities provided for, the Poster Presentation are identical to those of your Level 3 Individual Project. That is, you will have access to a small table, a stand approximately 2m high and 1m wide, and a power socket.

The purpose of the presentation is to explain and ‘sell’ your project. You should design posters that clearly explain the aims and achievements of your project. However, they should also be eye-catching and attractive. Your project title and your name should be prominent and clear, and you should also include the event title.

You should also be prepared to explain your work. Generally, you should prepare a short (2-4min) and long (5-8min) presentation of your project, and these should include a demonstration of your software.

Posters work best when not covered in dense text. Remember that you will normally be present to explain your project, so it isn’t necessary for your posters to describe every detail. However, your posters should make sense on their own and give readers a general overview of your project.

## User Manual

The user manual should clearly describe how to use the software you have developed. Some kinds of application, particularly when they are well-designed and have a good user interface, do not necessarily need long user manuals. So do not be afraid to make this short *if that is appropriate*. This will not always be the case (for example API projects, which will demand a completely different structure).

Good user manuals for software applications are hard to write. It is common, unfortunately, for them to be turgid and unhelpful lists of commands in no particular order. Consider carefully what actions are common and deal with them first. Also, consider presenting information graphically (for example screenshots). An initial ‘quick start’ guide can make a user manual easier to read and more accessible.

## Design Document

The design document should describe the project design in a (semi) formal notation, such as UML. It is *completely acceptable, and encouraged, to base* the design document contents on tool-generated descriptions. For example, Doxygen, Javadoc, etc., or some combination of these. However, for this to be effective, the basis (usually the application source code) must itself be well-documented. In other words, a substantial part of the work of writing the design document is normally in carefully laying out and documenting the application source code. Also, it is *not acceptable* to simply ‘dump’ the output of documentation-generation tools into the design document - it is almost always necessary to ‘post process’ the output.

In addition to a semi-formal description of the design, the Design Document should contain rationales for the key design decisions made. Decisions taken should be compared against alternatives. It is obviously inappropriate to do this for *every* design decision made. So concentrate on critical decisions, and those that are unconventional or unusual. Figs. 5 and 6 show examples using UML – both manually and automatically from the code.

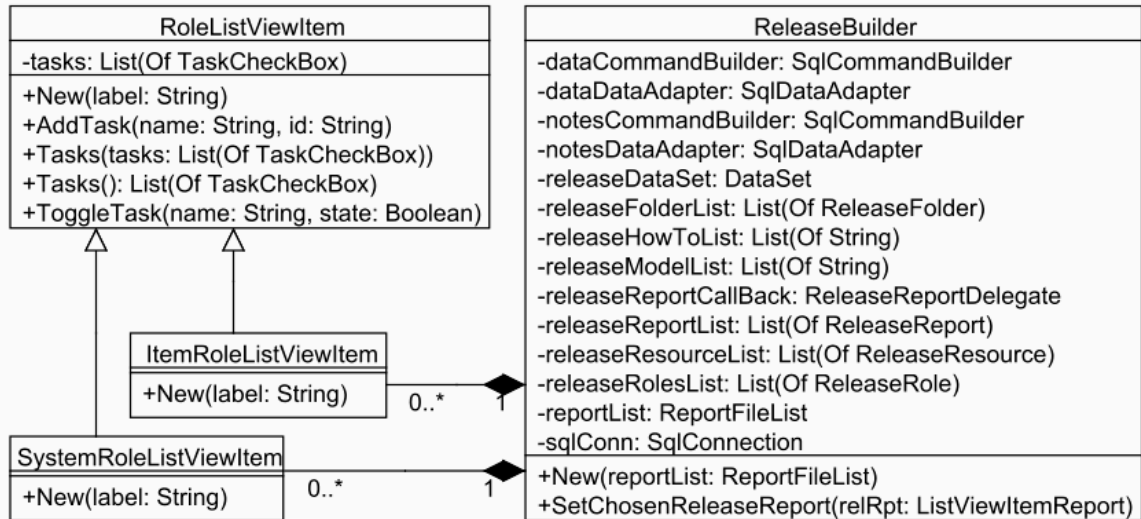


Fig. 5. A manually constructed design example

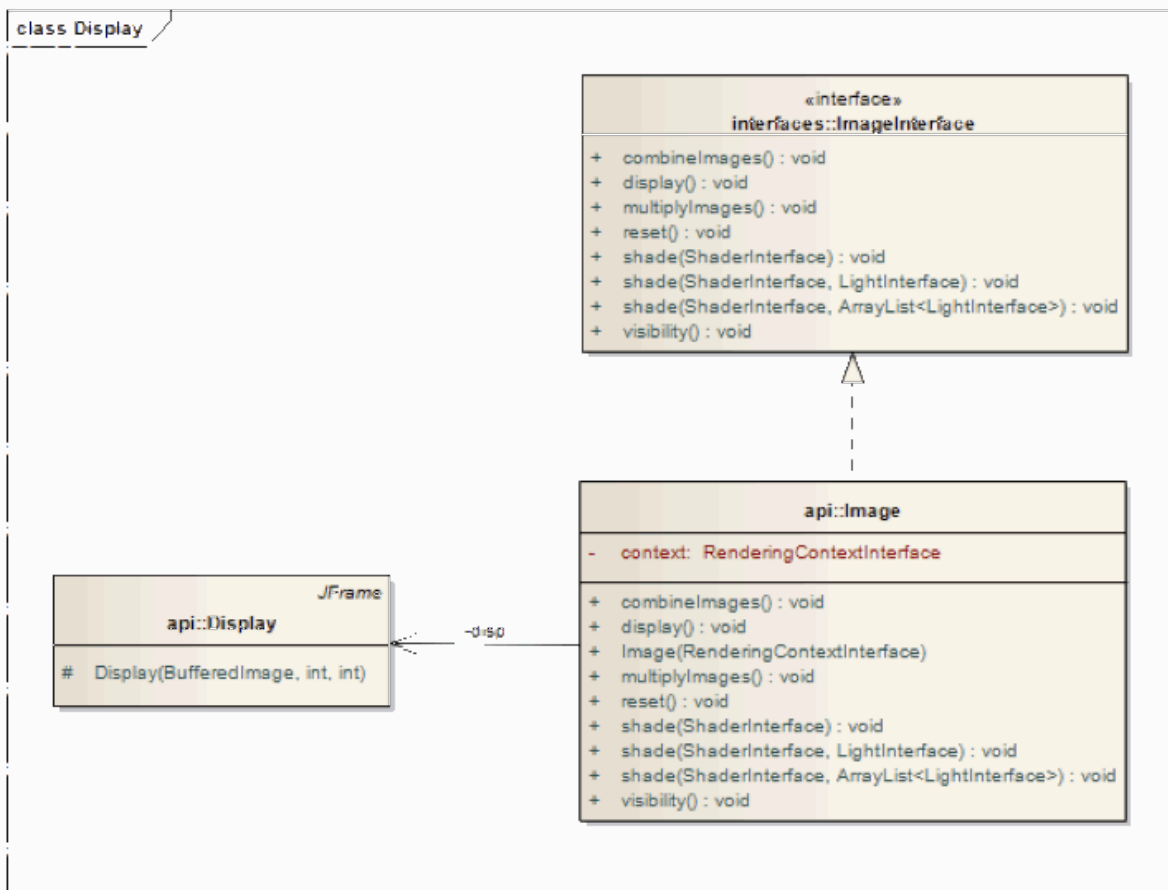


Fig. 6. Design example generated using Enterprise Architect.

## Testing Document

Because this is intended to be a software engineering project, testing should be thorough. It should be considered routine in a project of this type and level that standard industry practices, such as unit testing of components, and regression testing after changes. The Testing Document should contain a brief account of the testing methods applied during development. However, there is no need to detail exactly *what* tests have been carried out in this phase (unless substantial problems occurred; in which case it is also appropriate to discuss it in the Narrative and Reflective Account).

The process that should be fully documented in the Testing Document is the final *acceptance testing*. A suite of tests should be designed to ensure as far as possible that the software meets its specification. You are encouraged to use a tabular layout which (a) gives each test an identifier, (b) explains what the test is, (c) explains what aspect of the specification it refers to, and (d) describes the outcome. In the case of tests that fail, it is usually appropriate to add some commentary: did the test fail because the software was faulty, or was the test itself not completely appropriate? (That is, is the behaviour of the software correct even though the test ‘fails’?) Fig. 7. shows an example testing format.

<b>Asserts Specification(s):</b> SSK1, SSK2, SSK3, SSK4, SSK5
<b>Description:</b> Tests whether the four stack classes are instantiated correctly and that a base array containing (0,0,0) values, or (t=999999999, index=-1) for the intersection stack, are pushed onto the java.util.Stack each maintain.
<b>Acceptance Criterion:</b> <ul style="list-style-type: none"> <li>• DvStack, PvStack, InterStack, ImageStack classes are instantiated by the RenderingContext class.</li> <li>• Upon instantiation they are pushed to contain an initial base array for writing to.</li> <li>• The array is of the size defined by image resolution specified by the user to the RenderingContext class.</li> <li>• DvStack, PvStack and ImageStack contain a base array filled with 0's.</li> <li>• InterStack contains a base array filled with (t=999999999, index=-1).</li> </ul>
<b>Result:</b> PASS.

**Fig. 7. An example testing format.**

## **Narrative and Reflective Account**

This is the document in which you write in the narrative style that will be familiar to you from your Level 3 Project. The Narrative and Reflective Account is not intended to semi-formally document some aspect of the project, unlike the majority of the other documents you will have written. Instead, it is where you describe the course the project took, the problems encountered and solved, and what you have gained from the process.

There is no need to include a description of the development of every part of the project. Instead, concentrate on those areas that caused you problems (and how you solved them, or perhaps worked around them).

Reflection on the project is critical to evaluation. You should explain what you have learned from the project and its progress. In most cases, more is learned from mistakes than correct decisions. So it is usually best to concentrate here on what went wrong: what decisions did you take that later turned out to be the wrong ones? How did you retrieve the situation? Did early incorrect decisions have consequences late in the project? Did you learn from early mistakes and avoid later ones? Looking back, what would you have done differently? It is also appropriate to include a brief evaluation of the success of the project: how well did it meet its initial goals?

# Appendix - Marking Forms



# MEng Computing CSM14 Industrial Project: Milestone 1

**Deliverables:** *Methodology & Requirements Document* (15%, typically 10-20 pages)  
*Specification Document* (10%, typically 10-20 pages)

**Student :** \_\_\_\_\_ **Marker:** \_\_\_\_\_ **Supervisor/2nd Marker**

**Methodology and Requirements Document Mark:** \_\_\_\_\_

## Comments

The Methodology and Requirements Document should address the following issues:

1. Software Engineering and Testing methodologies to be applied.
2. Risk Analysis, and steps taken to mitigate serious risks with a high likelihood.
3. Project Requirements. The project requirements should preferably be expressed in structured form and as precisely as possible - not as narrative text.
4. Project Plan.

**Specification Document Mark:** \_\_\_\_\_

## Comments

The Specification Document should address the following issues:

1. Project Specification. This should address the behaviour of the proposed system and any other important aspects (e.g. performance), and be expressed in a structured form.
2. Meeting the Requirements. The Specification should be cross-referenced with the Requirements to show which aspects of the Specification address each aspect of the Requirements, and to ensure that all Requirements are met.

Note that unlike CS-334/CS-344 a significant aspect of the final assessment is based on *delivery*. Therefore students are making a *commitment* in the Requirements and Specification.



# MEng Computing CSM14 Industrial Project: Milestone 2

**Deliverables:** *Interim Report* (10%, typically 10-20 pages)

**Student :** \_\_\_\_\_

**Marker:** \_\_\_\_\_

**Supervisor/2nd Marker**

**Interim Report Document Mark:** \_\_\_\_\_

## Comments

The Interim Report is a narrative progress statement, describing the work done on, and the status of, the project (for example, how closely the plan is being followed, any problems that have arisen. Only under exceptional circumstances should the Requirements and/or Specification be modified (for example, if there have been issues with an outside collaborator).



# MEng Computing CSM14 Industrial Project: Milestone 3

**Deliverables:** *Poster Presentation* (10%)  
*User Manual* (10%)  
*Design Document* (15%)  
*Testing Document* (10%)  
*Narrative & Reflective Account* (20%, typically 10-20 pages)

**Student :** \_\_\_\_\_ **Marker:** \_\_\_\_\_ **Supervisor/2nd Marker**

**Poster Presentation Mark:** \_\_\_\_\_

### Comments

The Poster Presentation should explain and demonstrate the project. The assessment should be based on the quality of the posters, how well the student explains the project, and how well the application works.

**User Manual Mark:** \_\_\_\_\_

### Comments

The user manual should describe the operation of the application. Credit should be given for clear explanations and ease of access to specific required information (that is, it should not be necessary to read the entire document to locate a particular piece of information; sections should be clearly labelled and/or there should be some form of index). In addition, a good user manual will focus first on the main, common operations; more obscure features should be dealt with later. A 'quick start guide' may be useful.

**PTO**

**Design Document Mark:** \_\_\_\_\_

**Comments**

The Design Document should address the following issues.

1. The design of the system, preferably expressed in a (semi) formal notation such as UML.
2. A rationale for the design decisions made (alternative, rejected designs - and the reasons for their rejection, should be dealt with in the Narrative and Reflective Account).

Note that it is acceptable to use software tools to generate designs, but they should be accompanied by supporting commentary.

**Testing Document Mark:** \_\_\_\_\_

**Comments**

The Testing Document is about *acceptance testing* - not the testing strategy (which should be in the Methodology and Requirements Document) or a description of the process of testing during application development. Tests should be described formally: what is being tested with reference to the specification and design, how the test is carried out (e.g. what operations are performed, and on what data), was the test passed?

**Narrative and Reflective Account Mark:** \_\_\_\_\_

**Comments**

The Narrative and Reflective Account is an account of the progress of the project, with particular attention to problems and their solutions and concluding, reflective remarks. That is, what has the student learned during the process about developing software systems? What would the student do differently if the project was repeated?



# MEng Computing CSM04 Group Project: Milestone 1

**Deliverables:** *Team Structure & Methodology Document* (10%, typically 10 pages)  
*Requirements Document* (10%, typically 10-20 pages)  
*Specification Document* (10%, typically 10-20 pages)

**Students :** \_\_\_\_\_

**Marker:** \_\_\_\_\_ **Supervisor/2nd Marker**

*Note that at this stage, marks are allocated to the **group** and not to individuals.*

**Team Structure and Methodology Document *Group* Mark:** \_\_\_\_\_

## Comments

The Team Structure and Methodology Document addresses:

1. The roles of the team members (it is not necessary for each team member to contribute to each activity).
2. Software Engineering, Project Management and Testing methodologies to be applied. (Because there is no clear chain of authority in a student group project, the project management options are quite limited.)
3. Risk Analysis, and steps taken to mitigate serious risks with a high likelihood.

**Requirements Document *Group* Mark:** \_\_\_\_\_

## Comments

The Requirements Documents should address the following issues:

1. Project Requirements. The project requirements should preferably be expressed in structured form and as precisely as possible - not as narrative text.
2. Project Plan.

**PTO**

**Specification Document *Group* Mark:** \_\_\_\_\_

**Comments**

The Specification Document should address the following issues:

1. Project Specification. This should address the behaviour of the proposed system and any other important aspects (e.g. performance), and be expressed in a structured form.
2. Meeting the Requirements. The Specification should be cross-referenced with the Requirements to show which aspects of the Specification address each aspect of the Requirements, and to ensure that all Requirements are met.

**In all cases, attention should be given to how well the group is working. For example, are the documents uniform in style and content?**



# MEng Computing CSM04 Group Project: Milestone 2

**Deliverables:** *Interim Report* (10%, typically 10-20 pages)

**Students :** \_\_\_\_\_

**Marker:** \_\_\_\_\_ **Supervisor/2nd Marker**

**Interim Report Document Mark:** \_\_\_\_\_

## Comments

The Interim Report is a narrative progress statement, describing the work done on, and the status of, the project (for example, how closely the plan is being followed, any problems that have arisen. Only under exceptional circumstances should the Requirements and/or Specification be modified (for example, if there have been issues with an outside collaborator).

**Attention should be given to how well the group is working. For example, are the documents uniform in style and content?**



# MEng Computing CSM04

## Group Project: Milestone 3

**Deliverables:** *Poster Presentation (10%)*  
*User Manual (10%)*  
*Design Document (10%)*  
*Testing Document (10%)*  
*Narrative & Reflective Account (20%, typically 10-20 pages)*

**Students :** \_\_\_\_\_

**Marker:** \_\_\_\_\_ **Supervisor/2nd Marker**

**Poster Presentation Mark:** \_\_\_\_\_

### Comments

The Poster Presentation should explain and demonstrate the project. The assessment should be based on the quality of the posters, how well the group explains the project, and how well the application works.

**User Manual Mark:** \_\_\_\_\_

### Comments

The user manual should describe the operation of the application. Credit should be given for clear explanations and ease of access to specific required information (that is, it should not be necessary to read the entire document to locate a particular piece of information; sections should be clearly labelled and/or there should be some form of index). In addition, a good user manual will focus first on the main, common operations; more obscure features should be dealt with later. A 'quick start guide' may be useful.

**PTO**

**Design Document Mark:** \_\_\_\_\_

### Comments

The Design Document should address the following issues.

1. The design of the system, preferably expressed in a (semi) formal notation such as UML.
2. A rationale for the design decisions made (alternative, rejected designs - and the reasons for their rejection, should be dealt with in the Narrative and Reflective Account).

Note that it is acceptable to use software tools to generate designs, but they should be accompanied by supporting commentary.

**Testing Document Mark:** \_\_\_\_\_

### Comments

The Testing Document is about *acceptance testing* - not the testing strategy (which should be in the Methodology and Requirements Document) or a description of the process of testing during application development. Tests should be described formally: what is being tested with reference to the specification and design, how the test is carried out (e.g. what operations are performed), was the test passed?

**Narrative and Reflective Account Mark:** \_\_\_\_\_

### Comments

The Narrative and Reflective Account is an account of the progress of the project, with particular attention to problems and their solutions and concluding, reflective remarks. That is, what has the group learned during the process about developing software systems? What would they do differently if the project was repeated?

**In all cases, attention should be given to how well the group is working. For example, are the documents uniform in style and content?**

