Final Year Project Handbook

Computer Engineering Students

1989
كلية إتصالات الجامعية
Etisalat University College

Professor D. Vernon

Revision 2.0
January 2007

Table of Contents

1. The Importance of Final Year Projects 3

2. Choosing Your Project 4

3. Planning, Executing, and Managing Your Project 5

3.1 Problem Identification 5
3.2 Requirements Elicitation 6
3.3 Problem Modelling 7
3.4 Systems Analysis and Specification 8
3.5 System Design 9
3.6 Module Implementation and System Integration 10
3.7 Testing and Evaluation 10
3.8 Documentation 11
3.9 Good Engineering Practice and Safety Regulations 11
3.10 Back to the Beginning – Managing Your Project 12

4. Documenting Your Project 15

4.1 Project Specification 15
4.2 Interim Progress Report 17
4.3 Final Report 19
4.4 Presentations 22
4.5 A Very Short Guide to Good Writing 23

5. Project Assessment and Marking 25

1. The Importance of Final Year Projects

Your final year project is one of the most important aspects of your engineering degree. To see why, let's look at a definition of engineering, taken from the IEEE.

> "Engineering is that profession in which knowledge of the mathematical, computational, and natural sciences gained by study, experience, and practice is applied with judgement to develop economically effective use of matter, energy, and information to the benefit of humankind."

Engineering is first and foremost the application of knowledge. However, the application must be carried out with *judgement*, to ensure that the resultant system is *effective* and *efficient*, and that it is of *benefit* (which raises the issue of the ethical responsibilities of engineers – a topic for another day). The final year project is one of the primary the mechanisms used by the College to provide you with an opportunity to gain experience in the practical, effective, efficient, and beneficial application of what you have been studying for the past several years. Naturally, you will continue to gain engineering experience after you graduate but the final year project will be your first exposure to the full rigour of engineering practice. It is essential that you learn from this exposure and practise all of the engineering methodologies involved. It is particularly important that you learn not just to apply what you know, but to apply it with judgement, with the ability to assess what you are doing and to be critical of it.

There is another reason why your final year project is so important: it will inevitably be used as a discriminator to decide how good an engineering student you are. If you end up with a result in your degree examinations which is on the borderline between one grade and another, the examiners will look at how you performed in your project and then they will make a decision as to which grade you should be assigned.

Finally, your final year project counts for 25% of your 5th Year marks and 17.5% of your overall degree mark.

So, for the next 8 months, you should devote yourself totally to your final year project. Think of it as your passport to the engineering profession – your formal studies are your ticket but without your passport, you can't travel. Note, however, that you shouldn't neglect your other studies in the pursuit of your project: a passport is useless without a ticket!

Now that we have established the importance of your final year project, let's look at the important issues in pursuing it. There are four principal concerns:

1. Choosing a project
2. Planning, executing, and managing your project
3. Documenting your project
4. Assessment of your project

We will look at each of these in the following sections.

2. Choosing Your Project

Given that you are going to spend a lot of time working on your project, it is essential that you pick a project which you like and which you are capable of doing. Note that these are not necessarily the same things: just because you like a particular project doesn't mean you are qualified to do it. You may not have taken all of the requisite courses or it may be a more theoretically-aligned project whereas you might be a more practically-oriented engineering student (or vice versa). Think long and hard before making your final choice. At the very least, you should take the following steps in assessing and choosing an appropriate topic.

1. Find out what are your options.

   A list of projects proposed by academic staff will be distributed to you in week 1. You should:

   - Read all the descriptions
   - Identify the ones that interest you
   - Read them again

2. Make a short-list of three projects.

3. Think about proposing your own project. Using the descriptions you have read as a guideline, write your own proposal. Note, however, that the feasibility and suitability of your proposal will have to be assessed before it can be added to your list. Submit your proposal to the Project Coordinator who will have it reviewed by an appropriate member of staff.

4. Go and talk to the supervisors (*i.e.* the member of staff who proposed the project or the person nominated by the project coordinator in the case of your own proposal).

5. Go away and *write down* what you think the project is about.

6. Submit a ranked project selection form to the project coordinator by the end of Week 2.

7. Your selections will now be reviewed by the project coordination panel.

8. A list of allocated projects will be published in Week 3.

9. Now you can begin your project in earnest … you should begin by making a preliminary plan (see next section).

3. Planning, Executing, and Managing Your Project

Most students have no idea how to begin their project. This is understandable: it is the first time they will have had to tackle a large amount of work that is probably poorly defined (the project descriptions provided by lecturers are rarely complete!) To get started, it helps to know the key activities that result in a successful project. They are:

1. Problem identification
2. Requirements elicitation
3. Problem modelling
4. System analysis and specification
5. System design
6. Module implementation and system integration
7. System test and evaluation
8. Documentation
9. Project management

3.1 Problem Identification

Problem Identification involves a lot of background work in the general area of the problem. Normally it calls for the use of prior experience, typically experience you may not yet have. It requires an ability to look at a domain (e.g. telecommunications or engine control) and to identify the issue that needs to be addressed and the problem to be solved (*e.g.* elimination of noise or cross-talk on a communication channel, or engine control for temperature-dependent fuel efficiency). It also required an understanding of the theoretical issues by which we can model the problem. So, the first thing you need to do in your project is become an expert in the problem at hand: a *problem-domain* expert.

At the same time, you also need to know how to handle the tools that will enable you to solve the problem. These might include the operating system, the programming language, the application programming interface (API) definitions, class libraries, toolkits, or any application-specific analysis utilities. That is, you also need to become a *solution-domain* expert.

The only way to become an expert in both the problem domain and the solution domain is to learn as much as possible about the area and to learn it as quickly and efficiently as possible. Many people come unstuck at this first step and they launch themselves into a frenzy of unstructured research, reading much but learning little. Here are some tips to avoid this happening.

- Collect any papers, articles, book chapters you can on the area and make a copy for your own personal archive.

- Make sure you keep a full citation index, *i.e.,* you must record exactly where every article you copy comes from. Typically, you need to record the title of the article, the authors, the name of the magazine/journal/book, the volume and number of the journal or magazine, and the page numbers. If it's a chapter in a book and the author of the chapter is different from the editor of the book, you need to record both sets of names.

- Not all the articles you collect will be equally relevant or important. Consequently, it's not efficient to give each of them the same attention. But it's not easy to know how relevant it is until you read it. So how do you proceed? To solve this dilemma, you should know that there are three levels of reading:

1. *Shallow Reading*: you just read the article quickly to get an impression of the general idea. Read it twice. This should take a half-an-hour to an hour.

2. *Moderate Reading*: Read the article in detail and understand all of the main concepts; this will probably require you to read it several times and take a couple of hours to assimilate.

3. *Deep Reading*: Here you make an in-depth study of the article. This may take you several hours or even a couple of days. After many careful readings, you should know as much about the topic as the author.

The way to proceed with your 'reading in' is to

- ♦ Shallow read everything and write a 5-line summary of the article

- ♦ If you think the article is directly relevant to your project, label it, and put it on a list of articles to be read at the next level, i.e. Moderate Reading.

- ♦ Read all the labelled articles and write a 1-page summary.

- ♦ If the article is going to be used directly in the project, e.g. as a basis for a hardware design or a software algorithm, then you need to add it to your list of articles to be read at the next level, i.e. Deep Reading.

- ♦ Read all the Deep-Read articles and write extensive notes on them.

Note that the 'reading in' phase of the project can last quite a long time (there's a lot of reading *and writing* to be done) and it can overlap partially with some of the other early tasks, such as requirement elicitation, the topic of the next section.

Finally, it is very important that you realize that, in order to fully understand anything that you read, you must write it up in your own words. If you can't express or speak about a given idea, then you haven't truly understood it in any useful way. This is so important that it's worth saying a third time:

Writing is an Essential Part of Understanding

This is why, in all of the above guidelines, you see recommendations to write things down.

3.2 Requirements Elicitation

Having chosen your project, you will have in your possession a short description of what is involved in the project. You will realize by now that this is completely insufficient for you as a basis for doing the project. Consequently, your next task must be to find out exactly – and completely – what the project entails. This isn't as easy as it sounds. You might think that you should just ask your supervisor and he or she should tell you. It doesn't work like that. Quite often, a supervisor won't have an exact (and complete) model of what is required – supervisors are just like engineering clients and customers in the business and industrial world. It is your job to help your supervisor identify exactly what he wants. That's what good engineers do: they help people understand what they want and then they build it for them. Here's how you do it.

1. Talk to your supervisor.

2. Write down everything he or she says (by 'write down' I mean take notes of his or her words).

3. Write up everything he or she says (by 'write up' I mean express what your supervisor said *in your own words*).

4. Build a document describing what you think is required.

5. Go back to your supervisor and ask for her or his comments.

6. Return to step 1, and keep returning until you are both happy with *your* requirements document.

This all translates into one simple rule: find out what you want the final system to do and how it should behave, write it down, and get everyone involved to agree to it … in writing. And don't spare the detail: every single aspect of what's wanted should be teased out and agreed: what it does, what it doesn't do, how the user is to use it or how it communicates with the user, what messages it displays, how it behaves when the user asks it to do something it expects, and especially how it behaves when the user asks it to do something it doesn't expect.

This process is called requirements generation. It's also called *requirements elicitation* because it reflects better the fact that you have to work actively with the client to find out what they really want (as opposed to what they initially say they want, which is a completely different thing). Perhaps it should be called *requirements extraction* because it's sometimes like pulling teeth (but, of course, not in the case of any of the Etisalat University College supervisors!) Once you have been though the requirements elicitation process several times and you are happy that you really know where you want to go, you must write it down and get everyone concerned to agree to it. This then becomes part of the *system specification*: it says what **you** are going to do. But that's the subject of the section after next.

3.3 Problem Modelling

Once you know the requirements, and are an expert in the problem domain, you can abstract the problem from the problem space and model it computationally: this means we can identify the *theoretical* tools we need to solve the problem. Examples include statistical analysis for the elimination of noise on the communication channel, characterization of the relationship between fuel consumption and engineer cylinder temperature for the engine control; the extraction of facial features from images, and the statistical classification techniques used to match these feature with faces in a database.

This is the foundation of all engineering and science: the creation of a rigorous – usually mathematical – description of the real physical problem to be addressed, be it control, communications, electronics, or some computational model. For example, if your problem concerned with packet routing, you might represent it using a graph and deploy formal graph theoretic tools for its analysis; if your problem is concerned with signal analysis, you might choose a Fourier representation or an eigen-vector representation and deploy the appropriate theorems in Fourier analysis or linear system theory. If your problem is to do with building a database, you will probably model the system with an entity-relationship diagram and validate the model by normalization.

The key to all successful engineering is the use of an explicit model: if you don't have a model, you are probably not doing engineering. Connecting components (or lines of code) together is not engineering, irrespective of whether it works or not. Without the model you won't be able to analyze the system and, thereby, make firm statements about its robustness, operating parameters, and limitations.

You may also wish to consider the use of a symbolic mathematics package such as Maple in the development of your mathematical model.

3.4 Systems Analysis and Specification

With the requirements document, problem definition, and computational model identified, we can now say exactly what our system will do and under what circumstances it will do it. This is the system specification. In writing the specification, you should begin with the requirements document and then you should identify the following.

- The system functionality
- The operational parameters (conditions under which your system will operate, including required software and hardware systems)
- Failure modes and actions on failure
- Limitations & restrictions
- User interface or system interface

It should also include[1]

1. A **functional model**. This will usually take the form of a functional decomposition: a hierarchical breakdown of the major functional blocks involved in the processing/analysis/transformation. Typically, this will be a modular decomposition of the computational model. Each leaf node in the functional decomposition tree should have a short description of the functionality provided, the information (data) input, and the information (data) output.

2. **A data model**. The identification of the major data-structures to be used to represent input, output, and temporary information. This is sometimes known as a data dictionary. Note that we are not interested here in the implementation of the data-structures (e.g. linked list, trees, arrays) but with the identification of the data itself. Very often, it is useful to use entity-relationship diagrams to capture the data model.

3. **A process-flow model.** This model specifies what data flows into and out of each functional block (i.e. into and out of the leaf nodes in the functional

[1] These terms refer to the procedural or imperative programming model; different but related terms apply to the object-oriented model.

decomposition tree). Normally, data-flow diagrams are used to convey this information, and are organized in several levels (i.e. DFD level 0, DFD level 1, etc.) The level zero DFD is equivalent to the system architecture diagram and shows the sources and sinks of information outside your system.

4. **A behavioural model**. This will typically use a state-transition diagram to show the behaviour of the system over time, i.e. the different states it can be in, the event and triggers that cause a change in state, and the functional blocks associated with each state. It is also often useful to create a control-flow diagram: a version of the data-flow diagram with events and triggers superimposed on each process.

5. A clear and detailed definition of all the user and system interfaces; one of the best ways of encapsulating this information is to create a **user-manual**.

All this information is collectively known as the '**system specification**' and is the result of an activity know as systems analysis.

Once you have this specification, before proceeding you must return and see if it actually matches what the user needs: *i.e.* you need to validate that the system specification satisfies the requirements (you would be surprised how often it doesn't). If it does, you can proceed to the next activity: software design. If it doesn't, then either the requirements were wrong and need to be changed, or the specification was wrong, and needs to be changed, or, more likely, both were wrong and need to be changed. You should get the explicit agreement of your supervisor that all is in order. If it isn't, then you must go back to requirements if necessary and revise them and the specifications (with your supervisor's agreement on everything). After this, you validate again, and you keep doing this until everyone agrees. Then, and only then, should you proceed to the next phase of the execution of your project: Design.

3.5 System Design

You are now in a position to design your system using whatever design methodology is appropriate for the area (and these will inevitably be specific to the particular area, be it filter design, amplifier design, software design, and so on). That said, there are a few general guidelines that apply to all areas:

- ❑ Identify several design options – algorithm, data-structures, files, interface protocols – and compare them.

- ❑ Analyze your design to ensure it is technically feasible (*i.e.* validate its realizability). Remember, you can't always build everything you design, either for theoretical reasons (ideal filters, for example) or for pragmatic reasons (a 1-Farad capacitor would make for some interesting implementation problems).

- ❑ Analyze your design to ensure it meets the specifications (i.e. validate its operational viability)

- ❑ Cost your system (*i.e.* validate its economic viability)

- ❑ Choose the best design. *You* will have to define what 'best' means for your particular project. It might mean the cheapest to manufacture, it might mean the fastest, and it might mean the smallest – it all depends. It's up to you to identify the test for optimality. As John Canny in MIT once put it when comparing different filtering techniques: *you choose your optimality criterion, and you take your choice.*

Note well that this is the hallmark of good engineering: the practice of qualitative and quantitative assessment of different options. Note too that our original definition of engineering is reflected in this design process: the creation of effective, efficient, and beneficial systems.

3.6 Module Implementation & System Integration

Finally, we are at the point where we can build the hardware and/or write the software. There is not much to say here since the construction methodologies are so domain specific, even more than in the case of design. However, there is one small piece of advice which is applicable to all areas: *use a modular construction approach.* Don't attempt to build the entire system in one go in the hope that, when you switch it on or run it, it will work. This is the so-called Big Bang approach (everything comes into existence at one instant) and its name is very appropriate for it almost always results in initial chaos. It is much better to build (and test) each component or modular sub-system individually and then link them or connect them together, again one component at a time.

3.7 Testing and Evaluation

Most undergraduate engineers (and some graduate ones) misunderstand the meaning of the word *testing.* They think it means showing that something works: their project, for example. But it doesn't. Testing means much more than this. Certainly, you need to show that it works (*i.e.* that it meets the requirements and operates according to the specification), but a good testing strategy also attempts to break the system: to show not where it works but where it fails. This is sometimes referred to as stress testing. A well-engineered system will always have been stress-tested: that is, taken beyond the point at which it was expected to operate to see how it behaves under unexpected circumstances. This is particularly important for safety-critical systems (*e.g.* a heart pacemaker, an airline navigation system, a stock-exchange transaction processing system …). In engineering, we normally formalize the testing process by referring to three distinct goals:

1. *Validation*
   Simply stated, this test answers the questions: *Have I built the right system? Does it satisfy the requirements?* It may seem obvious, but you'd be surprised the number of times that the system which is built isn't what is wanted at all. You should compare the system's behaviour with the original requirements and system specification. Validation is extremely important and it should be carried out with great attention to detail.

2. *Verification*
   In this case, the questions are: *Have I built the system right? Is it computing the right answer?* This is what most people understand by testing.

3. *Evaluation*
   Finally, we ask: *How good is the system?* Again, the hallmark of good engineering: we seek to assess the systems performance and compare it to that of other similar systems. Ideally, you should identify some quantitative metric by which to

compare the systems, since numbers are the best and perhaps the only way to objectively describe performance. For example, the mean time between failures (MTBF) or the number of incorrect rejections in a pattern recognition system. Quite often, we use statistical measures as our comparative metric, *e.g.* the mean and standard deviation of some performance measure when the system is subjected to a large variety of input parameters and conditions.

3.8 Documentation

We noted earlier that writing is an essential part of understanding. We note it again here but in a different sense. In this case, writing is essential in order for others to understand what you have done. There are two reasons why you want others to understand your work:

1. So that you can be given credit for it (your final mark depends on it);
2. So that others can carry on your work and develop or maintain your system.

It is extremely important that you document your work at every stage of your project. We saw already that documentation is essential in the initial reading-in, requirements, and specification phases but it is equally important in the design, implementation, test, and maintenance phases.

The best way to organize your writing is to *keep a log book* of all work in progress. You should go out and buy a nice hard-cover notebook and write everything you do on the project into this log book *every day*. Every thought and observation you have on your project should go into this book, along with notes of meetings with your supervisor, results, theoretical developments, calculations, everything. This log book will become an invaluable source of material when you come to write up your project in the final report.

However, don't wait until the end of the project to begin the process of formal documentation. At the end of each phase of the project (or at the end of each task) you should write up a formal report on that phase. These reports will, in turn, become an excellent basis for your final report.

Finally, there is one other form of documentation which you will have to create during your project. This is the project presentation. Since the final report and the project presentations are so important, we will devote all of the section 4 to these topics.

3.9 Good Engineering Practice and Safety Regulations

Projects requiring the construction of physical models will be allocated dedicated bench-space in the laboratory. If any other special facilities are required, you should inform the Superintendent Technician. In choosing electronics components for experiments you should try to make use of standard components. Orders for specialized components may be submitted to the Project Technicians if there is no equivalent stock item. However, no responsibility is accepted for the effects of supply lead time.

A record of all components and parts used and their cost will be entered on your project record card. Expenditure above a quoted maximum limit must be referred to the Project Coordinator for signature.

As we noted above, you must keep a log book during your project in which to keep a full dated record of design work, theoretical work, experimental results, and conclusions. Note instrument numbers in case the experiment has to be repeated in identical circumstances. Project supervisors, assessors, and examiners may ask to inspect your log book at any time.

Projects involving the design and construction of equipment should be performed in a competent and well-engineered fashion with due regard to all relevant safety regulations. If mechanical construction is required, you should apply to the technician who will advise on the most suitable materials to use and the methods of construction. Thereafter, a clear sketch, drawn approximately to scale, is required, with dimensions included in metric units.

In view of the requirements of the Health & Safety Regulations you are not allowed to work in any room alone if your work involves exposed live conductors at dangerous voltages or large moving mechanical parts.

You should familiarize yourself with the contents of the *Electrical Safety Handbook,* a copy of which was issued to you in Year 1, and you should conduct all your work in accordance with its recommendations.

3.10 Back to the Beginning: Managing Your Project

One of them most important things you will learn when doing your project is the need to manage your time. Final Year Projects require a considerable amount of time. You should expect to spend at least 150 hours working on it, and probably 200 or more. Any attempt to try to complete a project in the last couple of months or so of the second semester is doomed to failure. They are complex and require careful thought and analysis to identify manageable component parts.

Consequently, it is essential that you begin your project early, work consistently at it throughout the year, and track your progress closely. Naturally, the best way to do this is to plan your project in considerable detail. We will identify here one of the fundamentals of good project management: scheduling.

A project schedule is an indispensable tool: building it forces you into thinking about all the things you need to do, their inter-relationships, the time each will take, and what each one will be used for. So, draw up a schedule:

- Identify all the major tasks; break these down into sub-tasks. Note well that the best input for this task is your system specification: there will be a task for each functional block and each data-structure, as well as sub-tasks for analysis, design, implementation, test, integration, and documentation. There will also be tasks for system test and evaluation, as well as documentation and report writing.

- For each task and subtask
    - estimate how much effort you expect it to take (hours) and over what period you will spread that effort (days): this is the task effort & duration
    - identify the required inputs – information, software, hardware, and, most

    important of all, the results of other tasks in your project.

    - Identify the expected outputs

    - Identify a course of action to take if the task fails for some reason (*e.g.* the software or hardware doesn't arrive in time)

- Now try to identify the sequence in which you should do each task. In this, you will have to consider the relationships between each task and the use of the output of one task as the input to another.

In drawing up your project schedule, you may find it useful to use a standard project management tool (such as Microsoft Project). These tools make it easy to draw the schedule and to track your progress. However, they won't do the planning for you, *i.e.* they can't identify tasks, subtasks, effort, duration, etc. That's something you have to do yourself. You should be able to make a good attempt at this by the time you've finished reading this handbook.

Project management tools usually represent a finished schedule in one of two ways, a PERT chart and a GANTT chart

PERT stands for Program Evaluation Review Technique. A PERT chart represents each task/subtask as a box containing its identity, duration, effort, start date, and end date (among other things). It displays not only project timings but also the relationships among tasks (by arrows joining the tasks). It identifies the tasks to be completed before others can begin and it identifies the *critical path*, *i.e.* the sequence of tasks with the longest completion time. The critical path is important as it defines the absolute minimum time needed to complete the project. Note that the critical path can change as task start-dates and durations are changed. Figure 1 shows an example of a PERT chart for simple project. On the other hand, a GANTT chart represents tasks by horizontal bars and lines are used to show the dependencies; see figure 2.

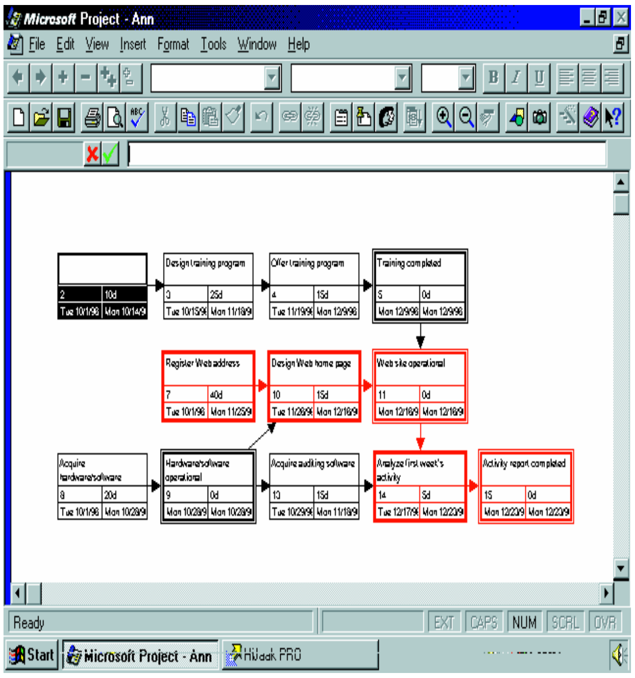
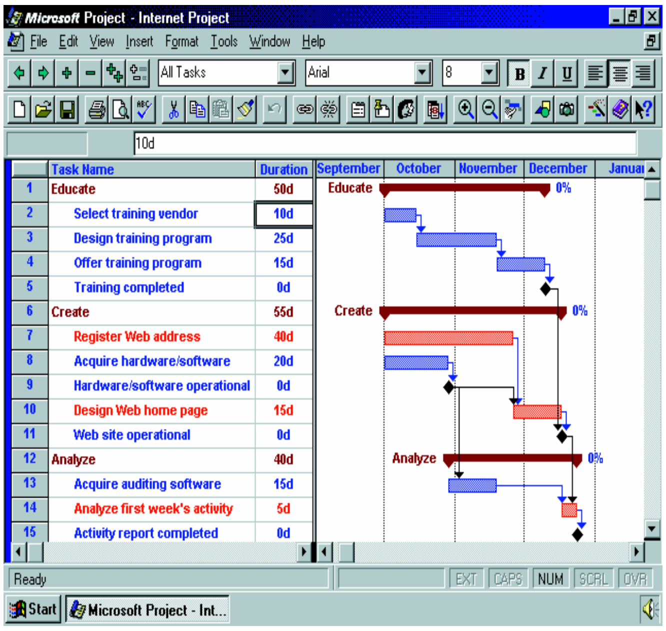

Microsoft Project - Ann
File Edit View Insert Format Tools Window Help
2 10d Tue 10/1/98 Mon 10/14/9
Design training program 3 25d Tue 10/15/9 Mon 11/18/9
Offer training program 4 15d Tue 11/19/9 Mon 12/9/98
Training completed 5 0d Mon 12/9/98 Mon 12/9/98
Register Web address 7 40d Tue 10/1/98 Mon 11/25/9
Design Web home page 10 15d Tue 11/26/9 Mon 12/16/9
Web site operational 11 0d Mon 12/16/9 Mon 12/16/9
Acquire hardware/software 8 20d Tue 10/1/98 Mon 10/28/9
Hardware/software operational 9 0d Mon 10/28/9 Mon 10/28/9
Acquire auditing software 13 15d Tue 10/29/9 Mon 11/18/9
Analyze first week's activity 14 5d Tue 12/17/9 Mon 12/23/9
Activity report completed 15 0d Mon 12/23/9 Mon 12/23/9
Ready
EXT CAPS NUM SCRL OVR
Start
Microsoft Project - Ann
HiJaak PRO

Figure 1: An Example of a PERT Chart. The Critical Path is shown in red. (Source: *Computing Essentials*, T. O'Leary & L. O'Leary, McGraw Hill, 1999)


Microsoft Project - Internet Project
File Edit View Insert Format Tools Window Help
All Tasks
Arial
8
10d
September October November December Januar
Educate 0%
Create 0%
Analyze 0%
Ready
EXT CAPS NUM SCRL OVR
Start
Microsoft Project - Int...

	Task Name	Duration
1	Educate	50d
2	Select training vendor	10d
3	Design training program	25d
4	Offer training program	15d
5	Training completed	0d
6	Create	55d
7	Register Web address	40d
8	Acquire hardware/software	20d
9	Hardware/software operational	0d
10	Design Web home page	15d
11	Web site operational	0d
12	Analyze	40d
13	Acquire auditing software	15d
14	Analyze first week's activity	5d
15	Activity report completed	0d

Figure 2: An Example of a GANTT Chart. The Critical Path is shown in red. (Source: *Computing Essentials*, T. O'Leary & L. O'Leary, McGraw Hill, 1999).

4. Documenting Your Project

During the course of the year, you have to write three reports. These are the Project Specification, the Interim Progress Report, and the Final Report. These reports are important for several reasons. First, they are formal components of the assessment exercise. In other words, they contribute towards the overall mark you will receive for your work. Second, and equally important, these report are valuable milestones for you: they help you focus on achieving concrete outcomes as your project progresses. The documentation of these outcomes is a difficult and time-consuming process: do not underestimate the importance or the magnitude of the task.

4.1 Project Specification

The first report you have to write is really an extended definition of what the project is all about. Note that this is a 'project specification' and not a 'system specification'. That it, it must address not just the system which will be created as a result of the project, but also the entire development process by which it will be created. Of course, it will also include a major section on the system specification but it must also address other issues such as the requirements elicitation, the development lifecycle, the tasks that must be undertaken to a achieve successful conclusion, a description of the problem domain, a description of the solution domain (*i.e.* environment in which the system will operate), and the theoretical foundations for the system.

Since this report must be produced approximately six weeks into your project, it is not likely that your system specification will be extremely detailed. In addition, the schedule of project tasks will probably be only good estimates and will have to be refined as the project proceeds.

The ultimate goal of this report is to present a clear and explicit definition of the required system, together with the steps that you will take to realize this system.

The following is an outline of a typical Project Specification Report.

Title Page

- ❑ Specific Title of the Project (*e.g.* "Automatic Test Pattern Generation for Digital Circuits")
- ❑ General Title (*i.e.* "Project Specification")
- ❑ Degree (*e.g.* B.Eng. in Computer Engineering)
- ❑ Author (name and student identification number)
- ❑ Institution (*i.e.* Etisalat College of Engineering)
- ❑ Supervisor
- ❑ Date

Table of Contents

Section 1. Introduction

1.1 Brief summary of the problem being addressed.
1.2 Overview of the target domain for the final system (where is it going to be used?)
1.3 Overview of the technical area, *i.e.* background technical and theoretical context.
1.4 Summary of the system functionality (what is it going to do?)
1.5 Overview of the report: what material will you be covering and how is it arranged?

Section 2. System Requirements

2.1 Required system functionality: focus on the functionality that a user requires of the system, rather than on how the system will deliver that functionality.
2.2 List of criteria that define a successful project: expected outcomes, required system behaviour, and especially performance metrics.

Section 3. Theoretical Foundations: The Engineering Model

3.1 Introduction
3.2 Details of theoretical model
- ❑ Mathematical/computational model
- ❑ Discrete or other approximations
- ❑ Limitations and assumptions
- ❑ Possible algorithm options

Section 4. System Specification

4.1 Functionality provided by the system
4.2 System interfaces, inputs, and outputs
4.3 System models:
    Functional decomposition
    Entity-Relationships
    Data-Flow Model
    Behavioural Model - State Transition Diagram
4.4 Specification of user interface
4.5 Failure modes and action on failure
4.6 Target architecture

Section 5. Task Analysis and Schedule of Activities

5.1 Task decomposition
5.2 Project schedule
5.3 Task specification: for each task, identify goals, inputs, outputs, estimated effort and duration, and task dependencies.

Section 6. Project Management

6.1 Meetings with supervisor
6.2 Major risks and contingency plans
6.3 Principal learning outcomes

References

Appendices

- ❑ Project description from the project list

4.2 Interim Progress Report

The Interim Report should focus primarily on presenting the progress that has been made in achieving the initial goals. It takes the project specification report as its baseline. Any amendments to the project specification should be highlighted and discussed in full. Similarly, any deviation from the schedule should be identified and all amendments to the schedule should be addressed: explaining the need for the change, the nature of the change, and any knock-on effects. The interim report should provide a complete summary of all project outcomes to date. These include project management documents, system analysis documents, theoretical development, system design, implementation, and early results.

The following is an outline of a typical Interim Report.

1. **Goals of the Project**

   Give a short summary of the main objectives of the project and the expected results.

2. **Synopsis of the System Specification**

   Provide an abstract of the first report (*i.e.* the Initial Specification), setting out the main features of the system or project and addressing at least system functionality, interfaces (electronic and/or human), and signal/information representations. This synopsis might be listed as a series of bullet points.

3. **Overview of Task Specifications and Project Schedule**

   List the primary tasks and sub-tasks required to carry out the project and an overview of the project schedule, giving the timings (start date, duration, amount of effort) of each task.

4. **Review of Tasks**

   Provide a *comprehensive* review of the status of each task and sub-task, setting out at least:
   - The status (not started, on-going, complete, behind schedule, ahead of schedule ...);
   - Problems encountered and identified solutions;
   - Anticipated problems and possible solutions;
   - Impact on the project schedule.

5. **Summary of Changes to the Specification**

   List and discuss the changes which have been made to:
   - System functionality;
   - Tasks and sub-tasks;
   - Project schedule.

6. **Interim Results**

   If possible, provide examples of any interim results you have achieved.

7. **Short Term Plans**

   Identify the next steps you will take in the project.

4.1 Final Report

Your final report is a critical part of your project. It defines what you have done and why you have done it. It is also one of the chief ways that your project is examined and assessed and it on the basis of the project that you will receive a large proportion of your marks.

In writing your Final Report, you will need to decide on its *content* and on its *structure*. We will look at both of these aspects in turn, beginning with the content. We will conclude with a few guidelines on good writing practice.

4.1.1 Content of the Report

Clearly, the content of the report is going to vary from project to project and it is difficult to make any strict recommendations. However, we can make a few general observations about the content of your report. We return again to the definition of engineering at the beginning and highlight the phrase:

"Knowledge … is applied with judgement …"

Your final year project provides you with an opportunity to demonstrate your ability to use your judgement. This means that you must show your skill at

- Assimilating
- Synthesizing
- Critically Appraising

all material relevant to the engineering project.

Your main opportunity to display your talents at assimilation and synthesis comes when you describe the background material you read, the requirements, specification, and design phases of your work. Needless to say, synthesis means that you must write the text yourself, expressing your understanding of the material in your own words.

Resist the temptation, no matter how strong, to copy sentences or paragraphs (or whole sections) from other books or articles. Copying is not synthesis and it demonstrates neither your assimilation of material nor your understanding of it. If you do come across a sentence or paragraph which is so good that is just has to be used, then do so and include it as a direct quotation, providing a reference to the original source.

It is extremely important that you also appraise, or assess, your work critically, *i.e.* with objectivity and with a view to seeing how it could be improved. Such honest criticism does not mean you will receive fewer marks; in fact, it is likely that you will receive more. Typically, you exercise your talents of critical appraisal at the end of the report in a discussion chapter but you can also exercise it throughout the report wherever it seems appropriate. Note that this exercise of critical appraisal is different from the testing processes of verification, validation, and evaluation, which refer to the functionality of the system you have designed. The critique you write applies less to the system and more to the overall objectives, methodologies, and findings of the project.

4.1.2 Structure of the Report

The structure of your report is critical to the impact you make in your writing. Remember that you are trying to convey a message to the reader and, since this message is likely to be quite complicated, you must assist him or her by making your points clearly and in a logical order. Think of it as telling an exciting story: you want to tell enough early on to attract the reader's interest but not too much otherwise he will become confused. You want to build up to a climax, incrementally revealing more and more of your message, but doing so in a way which builds on what you have already said. This is what we mean by a logical structure: breaking up the 'story' into a sequence of messages which follow naturally one from the other and which lead to an interesting conclusion (i.e. a conclusion you couldn't have guessed from reading page 1).

Because all engineering projects involve (or should involve) the exercise of a fairly standard engineering methodology (requirements, specification, modelling, realization, testing & evaluation), you can, if you wish, adopt a fairly standard structure. Note well, however, that this does not mean that you just simply have to fill in the gaps in a general report template: this standard structure simply provides you with a place to start as you begin to design the final structure and content of your report. You will still have to do quite a lot of work to make it fit your own project. A typical outline of a final year project report is provided on the next page.

You should design your own report to the level of detail given in the proposed structure, adapting it to your own particular needs. Note that you should do this *before* you start writing anything. It's just like designing a piece of software: the sooner you start coding, the longer it will take (and the more likely it is to be wrong). Try to achieve modularity and independence amongst your chapters and sections. At the same time, remember you are trying to convey a convincing message to the reader. Again, it's like telling a good story: you have to keep the reader interested and he has to be able to follow the story-line (which means there has to be one: make sure there is). Keep the thread of the story going continuously, from section to section, and from chapter to chapter by providing link sentences or paragraphs: at the end of a chapter, for example, remind the reader of the important messages, tell him why they are important, and then say what you need to look at next, and why, in order to continue with the story. That's your cue for the next chapter.

In general, your final report should be between 40 and 80 pages long, not counting appendices and front matter. The report should be printed on A4 paper, single-sided, using either in space-and-a-half or double line spacing, and leaving left and right margins of approximately 25mm. You are required to submit three copies (one for the Library, one for your supervisor, and one for the Academic Director). You should also make a fourth copy for your own records. Note that copyright of the projects rests with the Etisalat College of Engineering as do all intellectual property rights associated with the project. In essence, this means that the report is confidential to the College and may not be copied, published, or otherwise disseminated without the prior written permission of the College management.

Typical Structure of a Final Year Project Report

Title Page

- ❑ Specific Title of the Project (*e.g.* “Automatic Test Pattern Generation for Digital Circuits”)
- ❑ General Title (*i.e.* “Final Year Project Report”)
- ❑ Degree (*e.g.* B.Eng. in Computer Engineering)
- ❑ Author (name and student identification number)
- ❑ Institution (*i.e.* Etisalat College of Engineering)
- ❑ Supervisor
- ❑ Date

Abstract

- ❑ What is the subject matter of the report: what did you do?
- ❑ Motivation: why did you do it?
- ❑ Significance: what are your findings and conclusions?
- ❑ The abstract should be approximately 200 words long. It normally takes at least four revisions to achieve a good abstract.

Table of Contents

- ❑ Chapters
- ❑ Sections

Acknowledgements

- ❑ Help from friends, colleagues, and staff.
- ❑ Support from Etisalat
- ❑ Support from Parents, etc

Chapter 1. Introduction

1.1 Goals & Objectives: *what* was to be achieved?
Motivation: *why* undertake the project?
Method: *how* was it undertaken or carried out?

1.2 Overview of the technical area, *i.e.* background technical context

1.3 Overview of the report: what material will you be covering and how is it arranged?

Chapter 2. System Overview

2.1 Requirements

2.3 System Specification

Chapter 3. Theoretical Foundations: The Engineering Model

3.1 Introduction

3.2 Details of theoretical model
- ❑ Mathematical model
- ❑ Discrete or other approximations
- ❑ Limitations and assumptions

Chapter 4. System Analysis: System Model and System Architecture

4.1 Functional Decomposition
4.2 Entity-Relationships
4.3 Data-Flow Model
4.4 Behavioural Model - State Transition Diagram

Chapter 5. Design Issues

5.1 Introduction
- ❑ Different design options

5.2 Physical Realization
- ❑ Circuit Design
- ❑ Algorithms
- ❑ Mechanical Superstructure

5.3 Assessment of Design

Chapter 6. Implementation

6.1 Software development platform

6.2 Code design

Chapter 7. Testing

7.1 *Verification*:
- ❑ Did I construct the correct system (*i.e.* does it satisfy the requirements)?

7.2 *Validation*:
- ❑ Did I do construct it correctly (*i.e.* does it meet the specification)?
- ❑ Test scenarios (simple and complex)

7.3 *Evaluation:*
- ❑ How does it compare with other systems
- ❑ Qualitative assessment of performance
- ❑ Quantitative assessment of performance
- ❑ Metrics for measuring performance

Chapter 8. Discussion

8.1 Summary of work done
8.2 Critical appraisal of work done
8.3 Proposal for enhancement or re-design

References

Appendices
- ❑ Software listings
- ❑ Circuit diagrams
- ❑ Mechanical schematics
- ❑ Mathematical proofs
- ❑ Summary of Project Management
- ❑ Original System Specification
- ❑ Interim Progress Report

4.2 Presentations

During the course of your project, you will be required to give three presentations:

1. a short summary of your systems specification mid-way through Semester 1
2. a progress report early in Semester 2
3. a final presentation on your finished project at the end of Semester 2.

You have learned much about presentation skills during your time in the College and it wouldn't be appropriate to attempt to review everything you have been taught already. However, a few pointers may help you give a professional and impressive presentation.

- ❑ Don't depend too much on Powerpoint slides: your speech is the presentation and the slides support you (not the other way around).

- ❑ Take your time: pause frequently. Sometimes, the best thing to say is nothing. Short one-second rests create dramatic impact and also give your audience time to assimilate what you've said. Of course, you also have to maintain continuity and flow; otherwise people forget what you are talking about. It's a question of balance.

- ❑ Use a microphone (and practice using it before your presentation).

- ❑ Arrive early and make sure you know where all the equipment is. Know how to use it.

- ❑ Look at the audience, not at your slides.

- ❑ Project your voice (but don't shout).

- ❑ Smile: enjoy giving your presentation.

- ❑ Be confident: you've done some great work – here is your opportunity to get credit for it.

- ❑ The people in the audience *are* on your side (though sometimes they disguise it well!) They want you to succeed. If they ask you a question you don't understand, say so and ask their help. Ask them to explain, and ask nicely. If you still don't understand, don't bluff. Admit your ignorance and suggest ways of how you will overcome that lack of knowledge.

- ❑ Nobody knows everything; but that's no excuse for not trying to know everything. A knowledgeable person knows enough to do his job well, a wise person knows that he doesn't know everything, and an intelligent person knows how to find out what he doesn't know. Be knowledgeable, wise, and intelligent: be an engineer.

4.3 A Very Short Guide to Good Writing[†]

Good writing is difficult. It takes practice and a willingness to revise your work, many times. One of the best ways to learn how to write well is to read. Almost any reading material will do, as long as it's well written. For technical writing, you should at least read several previous project reports, conference papers, journal papers, magazine articles. The popular scientific press, e.g. Scientific American or New Scientist, employs a particularly simple and effective form of written expression. Try to emulate their style.

So why is writing well so difficult? The goal of writing is to convey a message to the reader. However, writing, and reading, are sequential processes. Therefore, you have to construct the meaning of your message, piece by piece, in a linear time-line. However, the meaning you intend to convey may emerge from many sources, not all related in a nice orderly fashion. This creates a problem for the writer: how to order the messages contained in each sentence effectively. The job of a writer is to make the sequence of pieces as mutually-relevant as possible so that the story or message builds naturally, each piece adding to the previous one. When the pieces are presented out of order (*e.g.* parenthetical expressions in the wrong place, or two related sentences split by a third) the impact on the reader is lessened. And remember the golden rule in writing: keep it simple.

The following are some pointers to help you in your task.

- ❑ Use short sentences and make sure the sentences are complete.

  A complete sentence has a subject followed (usually) by a verb, and then an object. For example: "The compiler identified two syntax errors." Of course, we can add other words to enhance the descriptiveness and richness of the sentence: "The C++ compiler identified two subtle syntax errors", and we can even additional phrases (which will normally have a subject-verb-object structure of its own). For example: "The C++ compiler identified two subtle syntax errors but, unfortunately, it was unable to find the semantic errors in my program."

  Remember that, if you remove all of the ancillary words, you should be left with a valid sentence; if not, then you haven't written the sentence correctly. It's a good idea to check all your sentences this way.

- ❑ Good writing strikes a balance between short sentences and longer more descriptive ones. Just as in oral communication, the full stops mean pauses: too many pauses and the text sounds disconnected, too few and it can be hard to follow the story line. Strike a balance but favour brevity over complexity.

- ❑ If you use pictures and diagrams (and you should), make sure each one has a self-contained explanatory caption. Never refer to a picture or diagram in the main text without saying what it is. For example, never say "Figure 2.3 shows the results of the noise test" and then carry on to another topic. Help the reader. Summarize the content of the figure in a short sentence: "Figure 2.3 shows the results of the noise test which demonstrate the robustness of the system to Gaussian noise with a standard deviation of 2.3 or less." If you have copied the figure from a book or article you must cite the source.

[†] Several of these guidelines are adapted from "The Elements of Style" by W. Strunk & E.B. White, Macmillan 1979.

- Make the paragraph your unit of construction. Each paragraph should bind one or more sentences about a given subject or idea. If the subject or idea changes, start a new paragraph.

- Omit needless words. Unnecessary words distract the reader. Don't write, "This is a system the performance of which is very useful". Instead, write, "This is a useful system".

- Write in a way that comes naturally. Speak the sentence. If it sounds correct, trust your ear and use the sentence. If it sounds unnatural, rewrite it.

- Avoid fancy words; they don't impress anyone.

- Be clear in your expression. If the idea you are trying to convey is getting lost in a sea of words and phrases, draw a line through the sentence and start again.

- Don't take short-cuts. Explain what you mean. Don't leave the reader to struggle trying to figure out what is the real meaning of your carefully constructed but concentrated sentence. He may conclude there is none. Explain all acronyms the first time you use them.

- Revise and rewrite. It is highly unlikely that you will manage to find the best way of expressing an idea with your first attempt. Nonetheless, make that attempt and then be prepared to revise it, again and again.

Remember, if you want to learn how to write a good report, you need to do two things: you need to read other good reports and you need to practise your own writing.

5. Project Assessment and Marking

Projects are assessed according to several criteria and at several points during the year. This assessment will be based both on your report and on presentations. The assessment criteria are as follows:

Section		Percentage of Total	Mark Awarded (0-100)	Date
Project Execution (15%)	Project Specification	5 %		
	Interim Progress	10 %		
Final System (45%)	System Design	15 %		
	System Implementation	20 %		
	Testing and Evaluation	10 %		
Project Report (30%)	Organisation and Clarity	10 %		
	Technical Content	10 %		
	Conclusions and Future Work	10 %		
Final Presentation (10%)		10 %		
TOTAL*		100 %		

*** Classification**:

80% - 100%	Project is an outstanding piece of work of a standard that should lead to an internationally recognized publication. Presentation is reaching professional standard.
70% - 79%	Project is excellent and may contain publishable material. Presentation is excellent.
60% - 69%	Project and presentation are very good. All design aims are met.
50% - 59%	Project and presentation are good. Most design aims are met.
40% - 49%	Minimum core of design aims has been met. Presentation is satisfactory.
0% - 39%	Most design aims are not met and implementation does not work. Presentation is not satisfactory.

The project is assessed by both the Project Supervisor and the Second Marker according to all of the above criteria. You must provide sufficient information in your project report to allow the Second Marker to make an appropriate assessment. If you don't provide the information, you can't be awarded marks.

Students may also be required to give a demonstration of their project and they may also be interviewed by the External Examiner.