The dissertation you write will will be read by at least two members of staff, and marks are awarded to it rather than to the software you have written.

Software does earn marks (~20%)

BUT report is main evidence for quality of software.

Learning Outcomes - On successful completion of this module, the student should be able to:

Assessed by:

1. Carry out a substantial software or hardware development task, or a substantial piece of research in Computer Science, Artificial Intelligence or Software Engineering.
   - Demonstration/presentation, project report

2. Work independently and prioritise different components of the work; manage a large project effectively.
   - Demonstration/presentation, project report

3. Take decisions and justify them convincingly.
   - Demonstration/presentation, project report

4. Orally present work undertaken, and answer questions about it convincingly.
   - Demonstration/Presentation

5. Write a formal report, detailing work undertaken and conclusions reached.
   - Project report
How do you earn the marks?

1. Carry out a substantial software or hardware development task, or a substantial piece of research in Computer Science, Artificial Intelligence or Software Engineering.
2. Work independently and prioritise different components of the work; manage a large project effectively.
3. Take decisions and justify them convincingly.
4. Orally present work undertaken, and answer questions about it convincingly.
5. Write a formal report, detailing work undertaken and conclusions reached.

Insight is engineering insight

Using planning to create a quality product

Don’t ask - What do I need to do to get the marks?

Do ask - How do I make the techniques work effectively?

Report structure (guidelines)

1. Title page.
2. Preamble: Table of Contents; Abstract/synopsis; Acknowledgements.
3. Introduction.
4. Three or four sections to cover: Further background material; Analysis and Specification; Design; Implementation and testing; User interface; Project management; Results; Appraisal.
5. Conclusions.
6. References and/or bibliography.
7. Appendices.

Without insight:

- write the section headings
- write something or other in each section
- use approved style; check spelling etc.
- include diagrams etc.
- make sure citations are clear

With insight:

- These are the tools for saying what needs to be said about your project
- use them effectively and convincingly
What does the report need to say?

- WHAT you did
- and WHY
- and HOW

BUT ... must say these effectively & convincingly
- no waffle
- no dishonesty
- no important points overlooked

Engineering setting
Without insight:

I needed to do some stuff to pass
40 credits of project module
- so here's what I did
ARTIFICIAL!
Engineering process loses its meaning

Engineering setting
With insight:

It would be great to have a product that does ...
In 400 hours I could nominal time for 40 credits
& that's my project.

Engineering setting

grand vision

Think of your project as if you're going to continue developing it into a sellable product. maybe you will
Then the engineering processes have their natural meaning.
The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance.

Iterative and Incremental development is at the heart of a cyclic software development process developed in response to the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between.

For this project a combination of the two was adopted.

Practical software development needs both

- need to design it in
  - depends on nature of project.

E.g. Ask: Where is feedback important?

Difficult in models like waterfall

According to Wikipedia:

The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of

By contrast:

Iterative and Incremental development is at the heart of a cyclic software development process developed in response to the weaknesses of the waterfall model. It starts with an initial

For this project a combination of the two was adopted.

I wrote some code, & then ran it to see what it did, then rewrote it to try to make it do something more sensible & kept going until the deadline.
Philosophy behind Waterfall =

- Think everything through before you code
- Brief, fast implementation phase turns it all into code
- Works for some things
- Worth doing it for them

Iterative =

- Need to try things out to see what works best
  e.g. GUI
- planning, design to facilitate this
  e.g. - rapid prototyping
    - model-view-controller structure makes GUI flexible

Case study: project planning

What report should have said:

\[ x,y,z \ldots \text{components of the project were specified as follows,} \]

\[ \text{but for a,b,c} \ldots \text{a more iterative approach was needed} \]
\[ \text{to enable this it was decided to} \ldots \]

Similarly for risk assessment. Are some components riskier than others?
Report sections as tools: what do they do?

Report structure (guidelines)

1. Title page.
2. Preamble: Table of Contents; Abstract/synopsis; Acknowledgements.
3. Introduction.
4. Three or four sections to cover:
   Further background material; Analysis and Specification; Design; Implementation and testing; User interface; Project management; Results; Appraisal.
5. Conclusions.
6. References and/or bibliography.
7. Appendices.

Grab reader’s attention.
- it’s the first thing they read

What is needed for the project & why?
- say the important things

- the easy things are probably unimportant

- e.g. specification, UML diagrams, use cases:
don’t bother with trivial examples

Describe overall vision
+ how project fits in

What have people done before with a similar vision?
- don’t want to just redo the same thing

- learn from their experience

Testing – How do you test comprehensively for important features?
Report sections as tools: what do they do?

Report structure (guidelines)

1. Title page.
2. Preamble: Table of Contents; Abstract/synopsis; Acknowledgements.
3. Introduction.
4. Three or four sections to cover:
   - Further background material; Analysis and Specification; Design; Implementation and testing;
   - User interface; Project management; Results; Appraisal.
5. Conclusions.
6. References and/or bibliography.
7. Appendices.

Report sections as tools: what do they do?

Very important!
Summarize achievements of project.
Put it in context of overall vision of project.
(Further work ...)

Style
Impression you want to give
= professional
• slightly formal – not slangy
• correct spelling & grammar
• straightforward
• not pompous or jargon
• concise – not padded out
Citations

Wherever you use other people’s work:
- acknowledge it
- make it completely clear it’s not yours
Otherwise you risk suspicion of plagiarism
See Peter Coxhead’s “Writing project reports”

Reusing code

You are not expected to re-invent the wheel; indeed you can legitimately be penalized for doing so. It’s good software engineering practice to re-use code. But you must make clear which parts of your code are taken from elsewhere and which are original. Carefully commenting your code can achieve this.

It is difficult to get this right just with comments.
Best to summarize in the report
- whose software was used
- how much there was

Summary

- Use report to bring out qualities of product
- Know what you are trying to say
- Think of an engineering vision that goes beyond the project
- Show insight into project process
- Style: professional
- Cite correctly