A Short Guide to Writing Your Report

UXCFXK-30-3

Digital Systems Project

Abstract

This guide is intended to help you produce a good Digital System Project report. It gives advice on how to gather relevant material, how to organise it into a suitable form, and how to then turn it into a written project report. It also describes the conventions that should govern the structure of the report and suggests some descriptive devices that you can use to make it more effective. A summary of the guidelines is given at the end. The appendix lists the rules governing presentation details.

Table of Contents

Introduction

This guide is intended to hep you produce a good final year project report. A good report is one that presents your project work *concisely* and effectively. It should contain various materials relevant to the work you have undertaken in respect of your project. It should be organised into a logical framework, and it should be supported by written material that follows well-established academic conventions in a consistent fashion.

The purpose of the project is to integrate various aspect of the taught material from entirety of your degree, and to demonstrate your academic research skills and your professional analysis, design, and implementation skills. It gives you the opportunity to conduct in-depth work on a substantial problem to show individual creativity and originality, to apply where appropriate, knowledge skills and techniques taught throughout the degree programme to further oral and written communication skills, and to practice investigative, problem-solving, project management and other transferable skills. The management and execution of the project is your responsibility, but you should seek and take advantage of advice from your supervisor.

When you choose a project, you should do so carefully, to reflect the focus of the degree programme you are enrolled in, your personal interests (the project needs to keep you interested for the whole the academic year) and the ability of the academic staff to support you throughout your project. Projects vary widely in the problem they address and the products they deliver at the end.

An important point to remember is that the report should describe *your own* work. Large chunks of bookwork describing standard material are unnecessary. You should simply refer to such material where necessary – assume that your reader is a competent computer systems theorist or practitioner.

Your project supervisor will guide you on what it is reasonable to expect a project in your chosen topic to deliver. However, all students are required to justify all decisions made at every stage of research and the development of appropriate deliverables, including the choice of approach.

Gathering Material

This section outlines the kinds of material you need to collect before you can begin writing in earnest. Most of the necessary material will consist of your own ideas and experiences gained while carrying out the project, and your approach to solving the problem you have decided to address. For the background study or literature review you will also need references to various resources such as key books and papers, policy documents, Internet resources, related software, etc.

While working on the project you may find it helpful to keep a notebook handy and record all relevant information. Typically, such information will include:

- References such as papers, books, websites with full bibliography details;
- Lessons learned, for inclusion in the "reflective" part of your report;
- Notes from meetings or interviews with
    - Your supervisor;
    - Potential end-users and other stakeholders;
    - Technical experts;
- and so on.

Also, we recommend that you keep a diary of all your project-related activities and supervision meetings. This will show the progress made during the life of the project and will provide a record of how you spent your time. You will need to provide evidence of your meeting logs at the Project-in-Progress day.

In general, you should supplement the material you generate yourself with relevant material from other sources. A good project report will show that you are aware of relevant work that other people have done and include this in your Literature Review section (See Section XX). You should include relevant references to such work in your project report. References to work in periodicals, i.e., magazines and journals, and conference proceedings may be more useful than references to textbooks, as periodicals and conferences are usually more specialised and up to date. References to technical manuals and national and international standards should also be included, where appropriate. You may also cite web sites as sources, if suitable. However, keep in mind that web sites may often contain incomplete or wrong information and in general textbooks or papers are a better reference and show that you have done a more extensive literature review than just searching for some keywords on the Internet.

Report Structure

You should consider, at the beginning of your project, what you need to do to solve the problem you have chosen to address. This will then inform choices about the structure of your report; your written report needs to be both a "narrative" (telling the story of your project) and an "argument" (providing a logical justification of the steps you have undertaken to solve your chosen problem). Once you have started to gather material you can begin to arrange it in a form which can then be refined into the final project report, though the outline chapter headings shown below will serve as a good guide in the early stages of your work.

All good project reports whatever their subject, follow certain well-established conventions and have a similar overall shape. They generally consist of a main body surrounded by other information (presented in appropriate formats) that support it in various ways. Some of these are mandatory, others are optional.

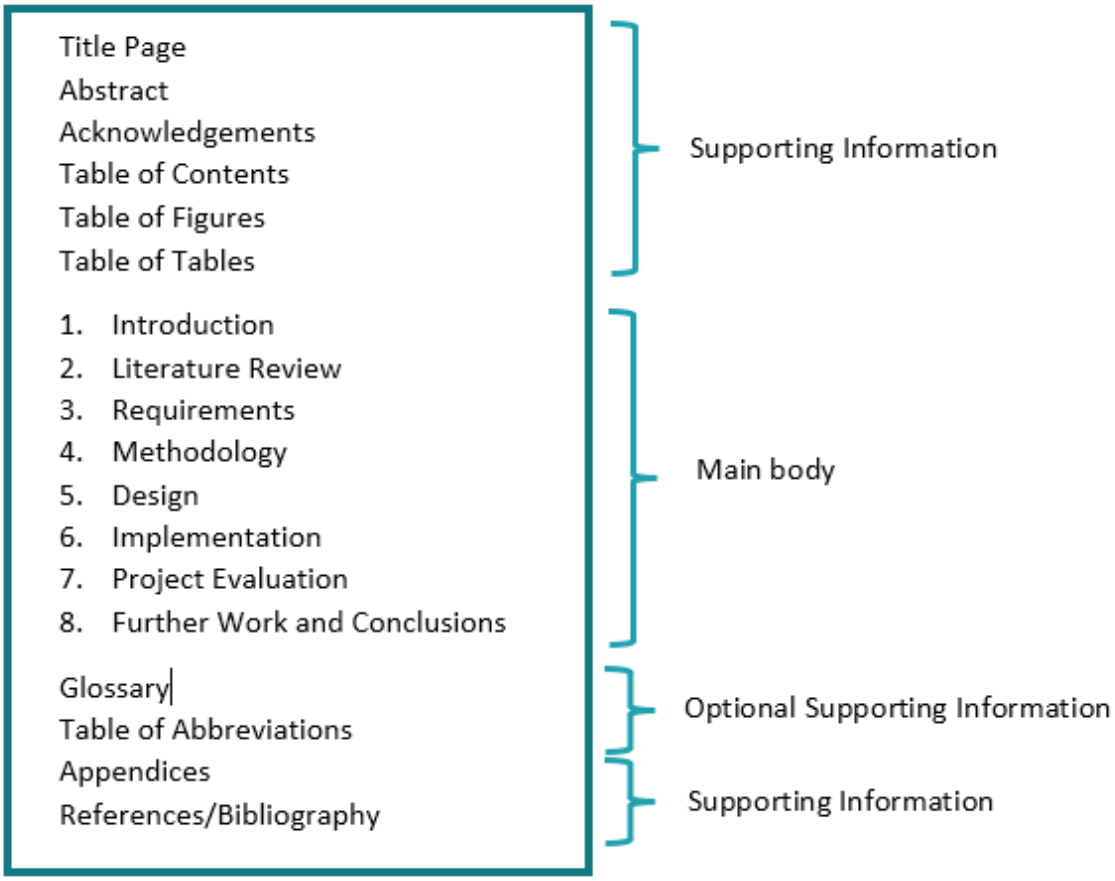


Figure 1: Suggested Report Structure

We look at each of the general sections of the report structure in more detail below. You can use this characteristic structure as a rough template for organising the material. However, often it may be of advantage to adjust the suggested structure to your project instead of sticking to the template. Consult your supervisor for advice. It is also a good idea at this stage to plan roughly how long each

part should be, to make sure that the length and overall balance are about right. You can then construct each part to produce a first draft of the main body.

Title Page

Your project title should state the topic of your project. It should provide enough (but not too much) information to capture the reader's attention. The title page should also include the author's details.

Abstract

The abstract is the **first thing** that someone reads in a report to get a general idea of what the project is about. Therefore, it plays a very important role.

The abstract should provide a short description of your project. Normally it is no more than one or two paragraphs **summarising your work**. As it is a summation of the project, it is generally a good strategy to write this at the end of your project.

Acknowledgements

The Acknowledgements section is where you recognize and thank everyone who helped you with your project. It's a way to display your appreciation to them in a public and permanent forum.

Table of Contents

To make your life easier you can use Word's styles to automatically create the table of contents as I have done for this document (see figures below). It is recommended you use Word's automated tools to ensure a good quality report. (Note, you are free to use other document tools, such as LaTeX, however, these are not documented here.)

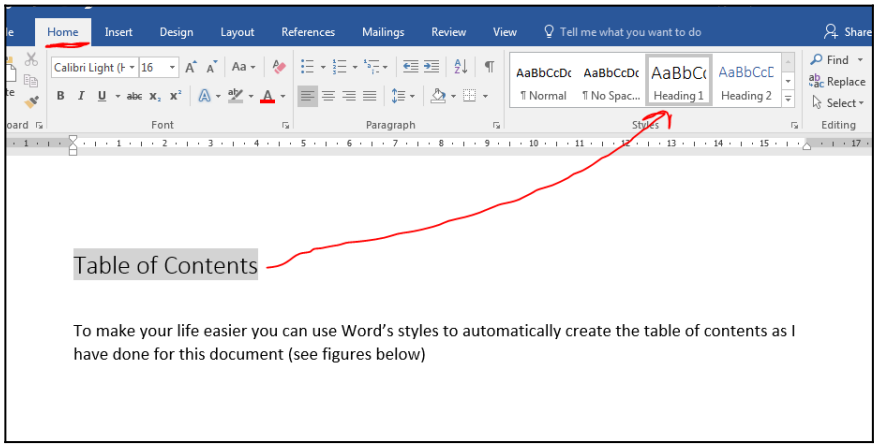


Figure 2 Selecting the Style Heading 1 for my chapter's Title

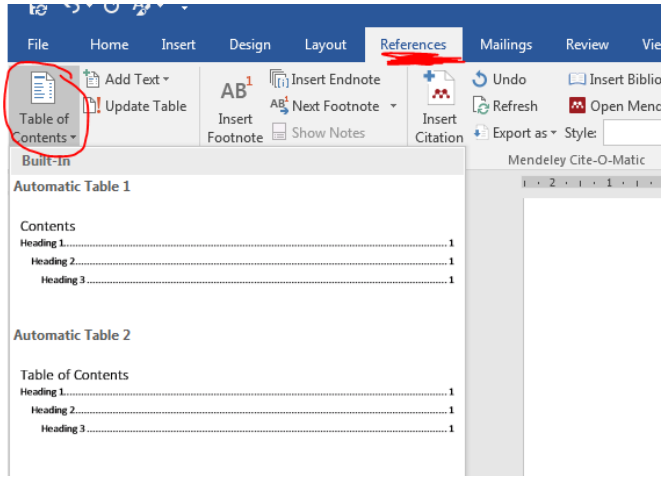


Figure 3 Creating the Table of Contents

Table of Figures

In this section you should provide the table of figures. You can also create this automatically if you have used the Word's captions to identify your figures (right click on your image and then insert caption).

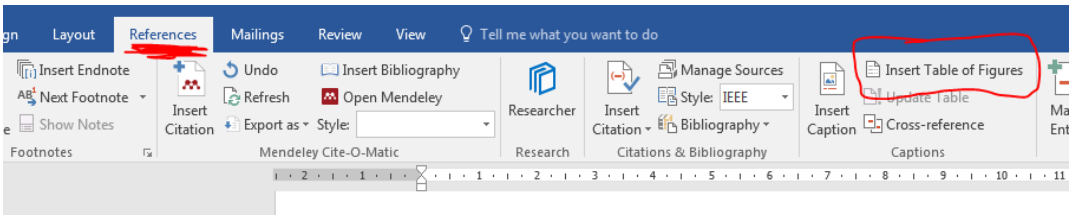


Figure 4 Inserting table of figures.

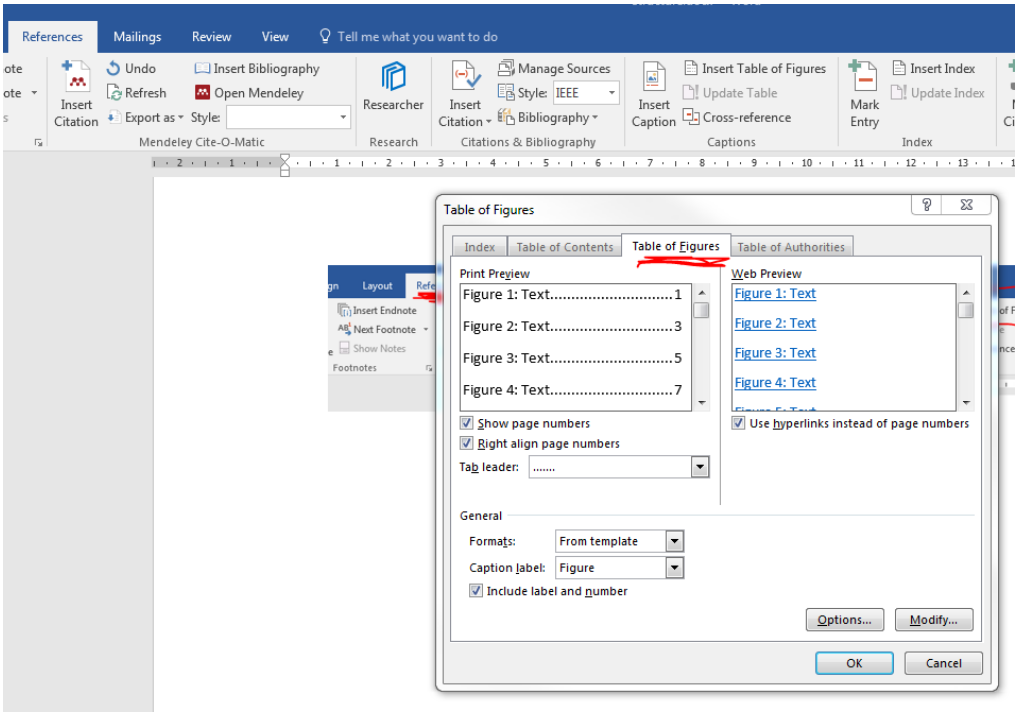


Figure 5 Inserting table of figures

Table of Tables

To automatically create a table of tables (provided that you have used Word's captions for tables; right click on table and then insert caption) you have to follow the same steps as in the table of figures: Click on **Cross-reference** and from the drop-down list select **Table**.

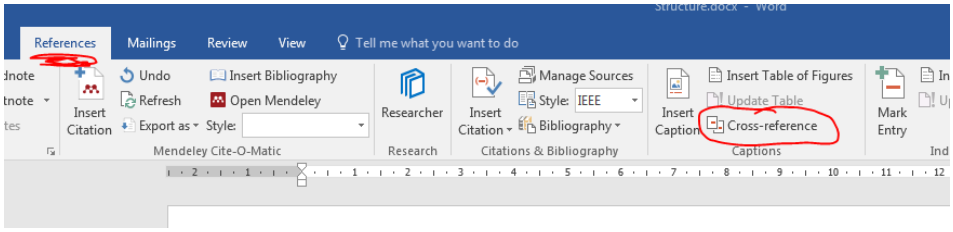


Figure 6 Creating table of tables

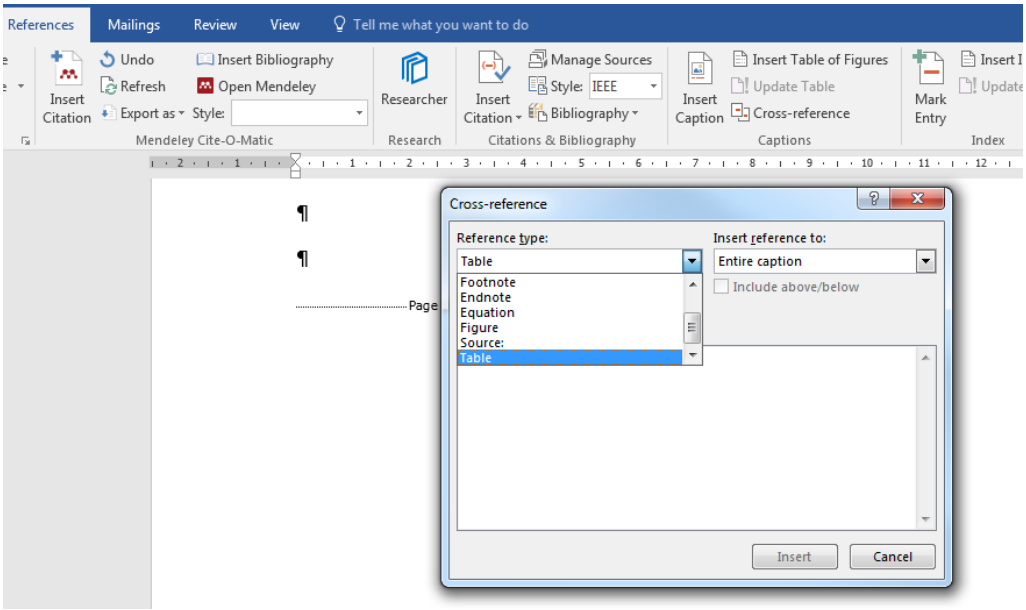


Figure 7 From the drop-down list select Table.

Introduction

A good introduction should tell the reader what the project is about without assuming special knowledge and without introducing any specific material that might obscure the overview. It should anticipate and combine main points described in more detail in the rest of the project report. Also, importantly, it should enthuse the reader about the project, to encourage them to read the whole report.

This chapter should provide a **short description of your project**, including the **aims and objectives** and the **scope** of the project, and an outline of your **report's chapters**. Ideally, your introduction chapter should answer these questions:

- What is the real-world problem you are looking at?
- Why is it important?
- What are your aims or goals of the project?
- What did you learn from the literature review? (provide an outline of your findings)
- How did you approach the problem?
- Outcome of the project?

At the end of the Introduction Chapter, you should provide an outline of your report's chapters. For instance:

Chapter 2 provides a critical review of relevant work/tools/software ……. Chapter 3 includes … In Chapter 4 we provide the … etc.

Literature Review

The purpose of the Literature Review section is to provide the typical reader with information that they cannot be expected to know, but which they will need to know to fully understand and appreciate the rest of the report. It should explain why the project is addressing the problem described in the report, indicate an awareness of other work relevant to this problem.

The remainder of this section is based upon chapter five of the book "**Projects in Computing and Information Systems**" by Christian Dawson.; chapter five is titled: Literature searching and literature reviews. You can access it online from the **Reading List** section on Blackboard.

Writing the literature review:

The goal of your work on the literature is to write a chapter in your final project report that provides a "coherent argument that leads to the description of a proposed study" (Rudestam and Newton, 1992:47).

This argument, or "story", will refer to the past and current literature in the field. It will involve a critical evaluation current approaches and discussion of current omissions.

You will not be able to achieve these aims if you merely read the literature that you find. You need to **read the literature critically** (see next section) and **evaluate it** as you go along.

Literature reviews evolve over time in an iterative fashion and should cover the following areas (Dawson, 2005):

- "Arrange the relevant literature in an area"
- "Critically evaluate past and current research in the field"
- "Identify your project within a wider context; Justify the existence of your project by identifying a gap in the field and showing how your project will fill that gap" (remember: this is about identifying its wider SIGNIFICANCE)

As such, the output of the literature review ideally sculptures the project's requirements.

Critical evaluation:

The main point of critical evaluation is **what you think about what you are reading**.

You have found some apparently relevant literature – **journal papers, conference papers, books, reports**. Now you need to critically evaluate them. This doesn't mean just finding faults with the paper or book; it means you should consider the following points (and others - see Dawson's book):

- "How does the article **fit** into and **support** the context of your project?" (Dawson, 2005) I.e. what is its general relevance to your work?
- "Can you use the results from the article in **your own work**?"
- "What can you **gain** from the article – ideas, techniques, useful quotes, etc?" (Dawson, 2005). I.e. what specifically does the paper have that you can adapt and use in your work?
- "Do conclusions follow logically from the work that has been presented? Are the arguments logical? Do they follow from one another?" I.e. is this a high quality paper?
- "What do you feel about what has been written? Do you agree with statements that are made? Are there counter-arguments?". If you disagree, you must say why you disagree and point to evidence that supports your viewpoint, or to weaknesses in an author's data or argument.

“Taking all these points into consideration, you will see that critical awareness of your chosen subject means a lot more than just understanding it and being able to regurgitate parts of it” (Dawson, 2005).

“You should be aware of its:

- Boundaries
- Its limitations
- Contradictions,
- Developing areas
- And dead ends”

Literature Searching

“A literature search is a ‘**systematic** gathering of **published** information relating to a subject’ (University of Derby, 1995)” (Dawson, 2005)

- Try to focus upon only books and papers, etc. that are relevant to your work. This will become easier over time as you find and digest more material and begin to make more sense of your field and to know what is and isn’t relevant.
- When you are trying to assess whether or not a work is worth reading, ask the following questions:
    - Does the title seem relevant?
    - Does the abstract identify the work as relevant?
    - What about the table of contents?
    - Do the key words relate to your work?
    - Is the author well known in the field (you will begin to learn this with time)

Literature sources include

- Books (often peer reviewed, so good because high probability of high quality)
- Journal papers (nearly always peer reviewed; some journals carry more weight than others)
- Conference papers (sometime not peer reviewed, so less authoritative and reliable)
- Databases like Inspec
- PhD, MSc and MPhil theses
- Company white papers
- Manuals
- Sources on the internet
    - Treat with caution
    - scholar.google.com is a search engine that searches academic papers and filters dross
    - Other useful sites include:
        - IEE Computer Society (www.computer.org)
        - DBLP Bibliography http://dblp.uni-trier.de
        - IBM Systems Journal (http://www.research.ibm.com)
        - UWE Library Search (https://www.uwe.ac.uk/study/library)

“Note interesting quotes and references as you go along; reference correctly from the start; have a system to organise and catalogue the material that you read; read recognised leaders and original theorists in the field; start with a broad search before you focus” (Dawson, 2005)

Managing information

“Arrange photocopied articles and your own notes into suitably labelled plastic wallets or folders”

- Create an index system that includes information on every paper or book or other source that you read. (It should include details of title, author, journal name, etc. plus a one paragraph précis of its content and an indication of its relevance to your work)
- You may like to try an electronic indexing system. Check the UWE Library pages on referencing tools - https://www.uwe.ac.uk/study/study-support/study-skills/referencing/referencing-tools
- Record references in the correct format e.g. using the UWE Harvard system, from the start of your project. Going back and “*fixing it later*” will introduce errors and be a larger job than you initially think.
- Mark-up each article with a précis and use highlighters to mark key sentences and paragraphs

Requirements

This chapter should include the description and presentation of your project's requirements. You need at least 10 functional and 10 non-functional requirements. (Note, these numbers are a minimum and it is reasonable to expect more.) Requirements should be realistic, detailed, and measurable/testable. You should also prioritise your requirements using a method such as the MoSCoW (**M**ust have, **S**hould have, **C**ould have, **W**on't have this time) method of prioritisation.

The remainder of this section is based on **Chapter 4: Requirements Engineering** of the book "**Software Engineering**" by Ian Sommerville 2016. You can access this book online from your reading list on Blackboard. (It is strongly recommended to read this chapter).

The requirements for your software system describe the **services** that your system should provide and the **constrains** on its operation. The requirements reflect the needs of customers.

User and System Requirements

User Requirements

These are high-level abstract requirements. User requirements are statements of what **services** the system is **expected to provide** to the system users and the **constrains** under which it must operate. Their expression can be done both via natural language and diagrams. They vary from broad statements to detailed descriptions of the system's functionality.

System Requirements

They provide a more detailed description of the software system's functions, services and operational constrains. The system requirements document (also called functional specification) should define exactly what is to be implemented. It can be part of the contract between the developer and the buyer.

The system requirements provide more specific information as to what is to be implemented.

User Requirements	System Requirements
- Not interested in the details. Provides a broad description of services and constrains. - Mainly for managers	- Provides more details. Describes specific services, functions, and constrains. - Mainly for system end-users, software developers etc.

For the purposes of your Final Year Project you don't need to distinguish user from system requirements. Your requirements need to provide adequate details (20 requirements in total – 10 functional and 10 non-functional – should be enough) on how the system will operate describing the system's services, functions and operational constrains. Your requirements need to be testable.

Functional and Non-Functional Requirements

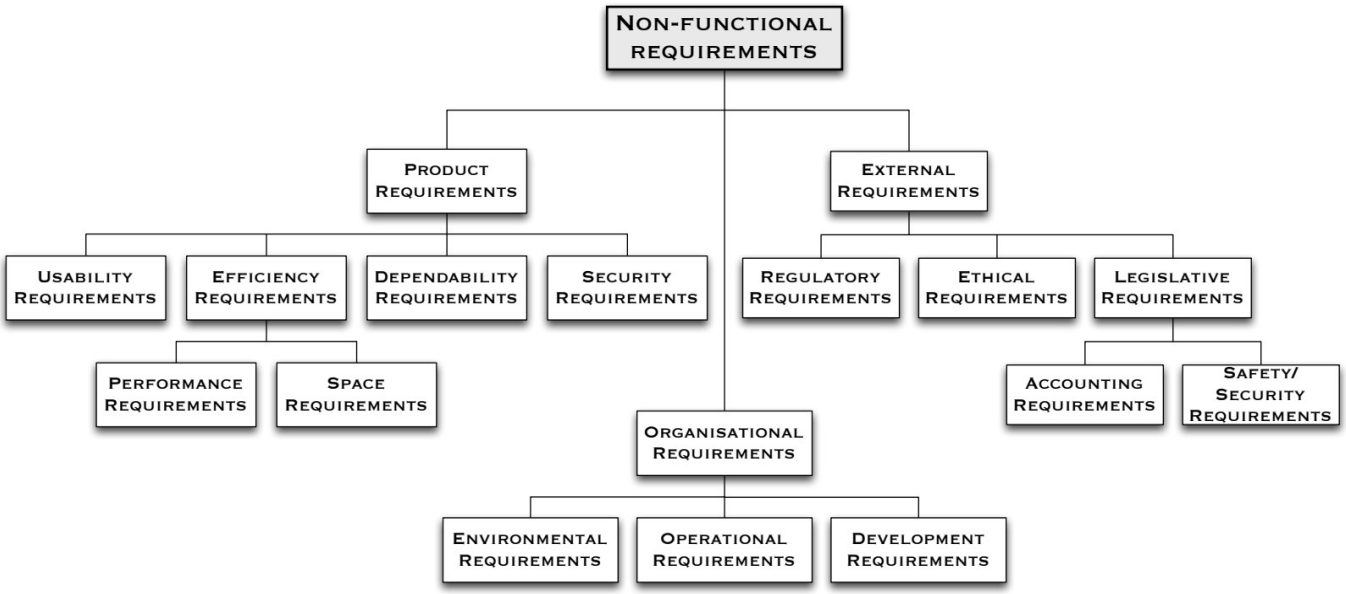
Functional: These are statements of services the system should provide, how the system should react to specific inputs and under particular situations. They can also include what the system should not do.

An example of functional requirements:

> **FR1:** A user shall be able to search the appointments list for all clients
>
> **FR2:** The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
>
> **FR3:** Each user shall be uniquely by his or her eight-digit employee number

Non-functional: These are constraints on the services or functions offered by the system. They include timing constrains, constrains in the development process and constrains imposed by standards. In most cases they apply to the whole system rather than its parts.

Example: Limiting access to the system to authorised users for security purposes is a non-functional requirement. However, this may generate functional requirements such as the need to include user authentication.



Elicitation and Analysis of Requirements

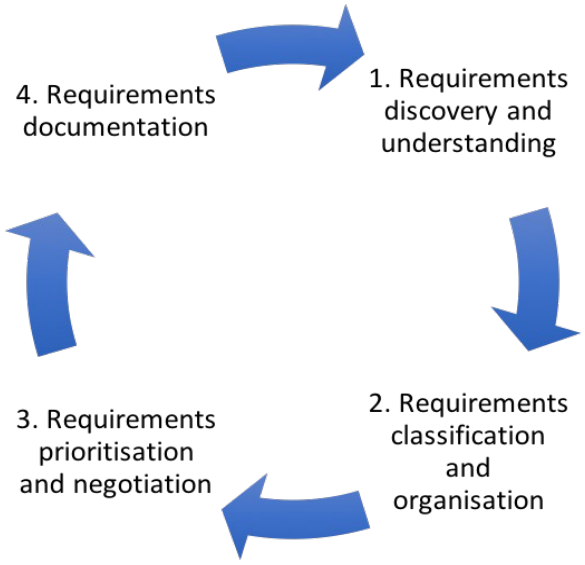


Figure 8: Elicitation and analysis process

To understand the current system and its environment you need to work through the followings stages:

- Identify the stakeholders: people with a stake in the new system
- Discover each stakeholder's requirements
- Investigate the environment, in particular the current process in which the current computer-based system plays a part
- Model the process in a process model e.g., role activity diagram or UML activity diagram

To discover each stakeholder's requirements for a new system, and to find out about the environment, use one or more of the following techniques:

- Interviews
    - Individual
    - Focus group
- Questionnaires

Requirements Specification

- Natural Language specification
- Structure Specification
- Use Cases

It is not sufficient just to list a set of requirements. You must describe the reason they exist in your requirement analysis and prioritise them (e.g., MoSCoW prioritisation or similar). Ideally, some of your requirements need to be linked with the output of your research.

Use Cases and UML diagrams should be used to provide additional context to the analysis of your requirement.

Finally, don't forget that the design of your software tests needs to take into account your requirements (e.g. software tests should check whether the requirements have been met and your test should cross-reference your requirement IDs).

Methodology

A short description of the methodology you followed. For more information about the various methodologies please read **Chapter 2** "**Software Processes**" and **Chapter 3** "**Agile Development Software**" from **Software Engineering** book written by Ian Sommerville (2016) – see reading list section on Blackboard.

Design

This chapter should describe and discuss in detail all design steps and use appropriate diagrams to show the classes, sequence flows, state machines, user case diagrams, database design, test design, etc.

You should read:

- **Chapter 6:"Architectural Design",**
- **Chapter 7: "Design and Implementation",**
- **Chapter 8: "Software Testing",**
- **Chapter 17: "Distributed Software Engineering" (if applicable) and**
- **Chapter 18: "Service-oriented architecture" (if applicable)**

from the "**Software Engineering**" book written by Ian Sommerville (2016) – see the Reading List on Blackboard.

You should also read the **UML2** book found in the reading list on Blackboard. UML will help you create useful diagrams.

Some very general guidelines:

High-level Design
System architecture
User interface design (possibly)
Object oriented design (possibly)
Database design (possibly)
Environment design (shows external CBSs and the relationship of these to the student's CBS)

Low-level Design
Developing structural models • classes • relationships between classes (inheritance, uses)
Developing dynamic models • Show interactions among objects (sequence diagram, collaboration diagrams) • **Change of state in objects (activity diagrams)**
Develop user-interface design • User-interface design • Communication with rest of the computer-based system
Detailed design and implementation • Reuse of existing code • Writing new code
Testing • Stages • Test design

Useful diagrams: class diagram, flow diagram, sequence diagram, state diagram, use case diagram, database design, test design

Implementation

The Implementation section is like the Design section in that it describes the system, but it does so at a finer level of detail, down to the code level. This section is about the realisation of the concepts and ideas developed earlier. It can also describe any problems that may have arisen during implementation and how you dealt with them.

Do *not* attempt to describe all the code in the system, and do *not* include large pieces of code in this section. Complete source code should be provided separately. Instead pick out and describe just the pieces of code which, for example:

- are especially critical to the operation of the system.
- you feel might be of particular interest to the reader for some reason.
- illustrate a non-standard or innovative way of implementing an algorithm, data structure, etc.

You should also mention any unforeseen problems you encountered when implementing the system and how and to what extent you overcame them. Common problems are:

- difficulties involving existing software, because of, e.g.,
    - its complexity,
    - lack of documentation.
- lack of suitable supporting software.
- over-ambitious project aims.

A seemingly disproportionate amount of project time can be taken up in dealing with such problems. The Implementation section gives you the opportunity to show where that time has gone.

This chapter should answer the following questions:

- How did you implement the project and why? (Demonstrate technical skills; no need to provide all the code; could provide some screenshots of the product and code snippets as discussed above)
- Have the aims and objectives of the project been met?
- Software testing
- Reflection on the test's results

You should describe how you demonstrated that the system works as intended (or not, as the case may be). Include comprehensible summaries of the results of all critical tests that were carried out.

You must also critically evaluate your results in the light of these tests, describing its strengths and weaknesses. Ideas for improving it can be carried over into the Future Work section. Remember: no project is perfect, and even a project that has failed to deliver what was intended can achieve a good pass mark, if it is clear that you have learned from the mistakes and difficulties.

Project Evaluation

You should use this chapter to reflect on all aspects of the project.

We believe that it is important that we reflect upon our performance in order to identify "transferable learning", that can be carried over into future activities. Reflection should focus on what Argyris calls "double loop learning"; this is where we identify, not relatively "simple skills", such as the mastery of a new programming language, but the impact of what we have done on the assumptions, concepts and ideas we used to make decisions about our work. For example, a "reflective practitioner" would try to identify the characteristics of the problem that has been addressed, and consider whether assumptions or decisions about the relevant approach to solving that problem had been appropriate, in order to make a better decision in relation to problems that might be encountered in the future

You should reflect on the project's outcome, the difficulties you faced and how you overcame them.

- Discuss **limitations**. Reflect on the tests' results.
- Any room for **improvement**? May lead into Further work?
- **Reflect** on all parts of the project (research, requirements, implementation etc.)
- **Reflect** on the way you used the **supervisor's feedback** (both from the Project-in-Progress day and from your meetings).

Conclusions and Further Work

The Conclusions section should be a summary of the aims of project and a restatement of its main results, i.e., what has been learnt and what it has achieved. An effective set of conclusions should not introduce new material. Instead, it should briefly draw out, summarise, combine and reiterate the main points that have been made in the body of the project report and present opinions based on them.

The Conclusions section marks the end of the project report proper. Be honest and objective in your conclusions.

It is quite likely that by the end of your project you will not have achieved all that you planned at the start; and in any case, your ideas will have grown during the course of the project beyond what you could hope to do within the available time. The Future Work section is for expressing your unrealised ideas. It is a way of recording that "I have thought about this", and it is also a way of stating what you would like to have done if only you had not run out of time[1]. A good Future Work section should provide a starting point for someone else to continue the work which you have begun.

[1] Remember to take into account Hofstadter‟s Law:
"Everything takes longer than you think, even when you take into account Hofstadter‟s Law."

Glossary and Table of Abbreviations

If you use any abbreviations, obscure terms or esoteric acronyms in the project report then their meaning should be explained where they first occur. If you go on to use any of them extensively then it is helpful to list them all in a table at the end so that readers can quickly remind themselves of their meaning.

References / Bibliography

We previously stated that you should relate your work to that of other people. Other work explicitly cited should be listed in the Reference section and referred to in the text using UWE Harvard format. It is important that you give proper credit to all work that is not strictly your own, and that you do not violate copyright restrictions.

It may be desirable to provide a Bibliography section separately from the reference section. In general, references are those documents/sources cited within the text. The bibliography lists documents which have informed the text or are otherwise relevant but have not been explicitly cited.

References should be listed in alphabetical order of author's surname(s) and should give sufficient and accurate publication details. For example,

> Chikofsky, E.J. and Cross, J.H., (1990) Reverse engineering and design recovery: A taxonomy. *IEEE software*, 7(1), pp.13-17.

Please read the UWE Library guide to referencing at https://www.uwe.ac.uk/study/study-support/study-skills/referencing for how to list references and cite these from within your report.

Appendices

Appendices should go at the end of your report. Appendices are where you present material which you want to include in the report, but which would seriously obstruct the flow of ideas if put anywhere in the main body. This could be extensive technical details or mathematical proofs, derivations of formulae, etc. required to support a point you are making in the report. Other documents you have written, such as user manuals, technical manuals or formal specifications should go here too.

The appendices should not contain any of the source code for your software. This will be submitted using a link to GitHub or GitLab.

Appendices should be headed by letters in alphabetical order, i.e. Appendix A, Appendix B, etc.

Using Descriptive Devices

In this section we will mention some well-established descriptive devices which you can use in your project report to improve its quality.

Cross-references

Cross-references are just references to other parts of the same document. For example,

```
This module contains procedures for operating on variables of
type WINDOW (see Section 2.2).
```

Section numbers will change if sections are added or deleted. Good typesetting or word processing software provides suitable mechanisms to automatically number sections and create such references such that they will always refer to the intended section. Make sure you know how your chosen software does this and select the right software to make this simple. If you use software that does not support cross-references or uses an overly complicated system, it is a good idea to wait until the report is almost complete before putting in any cross-references.

Backward references to sections earlier in the project report can make explicit connections between parts of the document that may not be connected obviously. Forward references can be used, for example, to reassure the reader that you are not going to leave them stranded after you have introduced a new idea without explaining it. For example,

```
This procedure uses the Volestrangler algorithm (to be
described in Section 4.3).
```

Note that too many forward references are probably an indication that the report could be organised better.

Footnotes

Many word processors have facilities for handling footnotes. By all means use them, in particular when you want to make a comment which is not strictly relevant, or which would upset the flow of ideas in the text. If the comment is closely related to the text, you may consider including it in parenthesis instead.

Lists

Traditionally, collections of items are listed within the text using the adverbs “firstly”, “secondly”, etc. Often, though, it is clearer to tabulate these items, particularly if there are many of them. The simplest way of doing this is to use a “*bullet*” list[2]. Various examples of bullet lists appear throughout this guide. Sometimes there is a need to nest one list inside another. To distinguish the two lists, the inner one can be indented and have a different symbol. Lists with more than one degree of nesting tend to appear confusing and therefore we do not generally recommend them. Listed items can also be keyed using numbers, letters, or other labels.

[2] “Bullet” is the name for the ● symbol, although other similar symbols are also available.

Figures

A project report that uses figures (i.e., diagrams or other pictorial techniques such as tables) to illustrate ideas will probably be easier to digest than one that does not. We therefore recommend that you use figures wherever appropriate3.

Be careful though. When drawing diagrams try to keep to a standard graphical notation that has been introduced during your studies, or that you have seen published widely, and use it consistently. Computer Science, unlike most other professions, has few established conventions governing the use of diagrams and this means that diagrams can sometimes make ideas more obscure rather than clarifying them.

If you feel you have to invent your own notation, remember that the best ones are usually the most economical, i.e., they use only a few different kinds of symbols. Also, you must explain the precise meaning of your symbols in a key. A very common mistake is to use arrows to illustrate relationships between items without declaring what that relationship is.

Graphics editors (i.e., picture processors) can be extremely useful, particularly if you have a great deal of drawing to do or if there is a lot if commonality among the drawings (because cut and paste operations can then be used with great effect). However, some artefacts are difficult to produce using standard software applications, and in such cases, it is quite acceptable to present hand-drawn diagrams. To include these into your report you may use a scanner or even take a photograph (or multiple partial photographs that you merge afterwards) of the artefact and include it in your report as any other computer-generated image

All figures should be labelled and captioned, for example,

```
Figure 3.10: Sub-System Architecture.
```

The label can then be used to refer to the diagram within the text, e.g.

```
See Figure 3.10.
```

All diagrams must be explicitly referred to somewhere within the text.

Similar to sections and subsections the labels may change if you insert additional figures or change the structure of the report. Again, good typesetting software will support automatic label generation and keeping the references to the figures consistent.

For some reports it may also be useful to distinguish between figures and tables and use separate labels for them (e.g., Figure 3.1 and Table 3.1 are two separate elements, sometimes also referred to as floats). Figures are diagrams, drawings, images, etc. while tables list information in a tabular layout, e.g., program running times for specific inputs

Literal Text

It is important when writing about software systems to distinguish in the text between the ordinary natural language you are using and the program code or other literal text. If you are using a word processor which offers both proportionally spaced and fixed width character fonts then there is a straightforward way of doing this. Program code and other literal text can be written in a fixed width

font such as “Courier New” while the natural language text can be written with a proportionally spaced font such as “Times New Roman”. For example:

The procedure `draw_circle (p:POINT, r:REAL)` draws a circle of radius `r` at point `p` on the screen.

Other similar kinds of text, UNIX commands for example, can be treated in the same way. Some typesetting systems also offer to include “verbatim” text, which you can use to insert small code examples, examples of the output of a program, etc. They are also typeset in a fixed width font. Using a fixed width font means that the code appears in the document much as it would do on a console. If you only have fixed width characters available on your word processor then put program code etc. into *italics* or **bold** text.

Note that using more than a few different character fonts, styles or sizes can make text look very untidy. Generally we recommend to use, e.g., a serif font for the main text (or a sans-serif font, if you prefer), a fixed-width font for literal texts as above, and optionally one sans-serif font for headings and captions (this can also be the same font used for the main text). Emphasis can be indicated by italics or stronger using bold text. If you use more fonts you should have a very good reason for this to support the content.

Conclusions

These are our main recommendations:

- Record all relevant information generated by the project:
  - use a notebook,
  - keep a diary,
  - log debugging sessions.
- Gather further material from publications or other external resources.
- Organise the material into sections agreed with your supervisor,
  - e.g.“Introduction”, and so on.
- Turn this material into written prose to form the project report’s main body.
- When writing the main body
  - keep your readership in mind;
  - identify commonality;
  - use sections and subsections;
  - follow stylistic conventions.
- Where appropriate use
  - cross-references,
  - references,
  - figures and other descriptive devices.
- Produce all required supporting structures according to convention, after completing the main body, and include this material in appendices to avoid disrupting the flow of your narrative.
- For examples to follow, look at
  - textbooks from reputable publishers,
  - the way this guide is written.
- Discuss an outline of the project report with your supervisor before you begin to write up; this will help you to plan your project. However, we strongly recommend that you write up your work as much as possible as you carry out your project, rather than leaving the writing to last.

We hope you will find this guide to be of value in completing your project. If you have any comments or wish to suggest good sources of advice that you have found, please let your project supervisor know so that we can update these guidelines.

Good Luck!!!

Document Revision History

Version	Date	Changes
1.0	27th September 2021	Initial Release

References

Dawson, C. (2005) *Projects in computing and information systems: a student's guide*, Addison – Wesley

Rudestam, K.E. and Newton R.R. (1992) *Surviving your dissertation*, SAGE Publications, London

Sommerville, I. 2016, *Software engineering*, Tenth, Global ed, Pearson, Harlow, Essex, England.

University of Derby (1995) *Literature searching for computing*, University of Derby, internal library publication.

Appendix A: Typesetting Guidelines

Length: The report should be between 8000-10000 words. with a tolerance of +10%. See the University's policy on word count for further details for what is and isn't included in the word count.

Font Size: Reports should use 11pt or 12pt typefaces for the body text.

Line Spacing: Reports should have single line spacing such as this guide. The report should be economical on paper. It should not, for example, contain excessive amounts of white space. Only the major sections need to begin on a new page.

Submission: Reports should be typeset with some word-processing system, e.g. LaTeX, OpenOffice, LibreOffice or Word. The final project report should be presented as a PDF or DOCX file with any other documentation in the appendices of the report. Artefacts produced for the project that are to be processed by other software such as a compiler or interpreter should be submitted along with the source code using a GitHub or UWE GitLab link.