GUIDELINES ON WRITING A COMPUTER SCIENCE PROJECT

1. PURPOSE AND INTRODUCTION

The purpose of this document is to provide guidelines on writing a Computer Science graduate project work. It is not intended to be used in writing a project describing theoretical research work.

A graduate project work represents the culminating experience resulting from your graduate study. Your project work is the most important artefact you create in earning your Higher National Diploma (HND). It will persist in perpetuity, long after your graduation.

It is the project work that proves that you have mastery in the subject matter. The project work demonstrates that you are capable of **finding solutions to significant problems**. It shows that you can perform critical analysis and make sound technical decisions based on the findings. Most importantly, the project work is a proof that you can describe the project related activities and results in a well written scholarly publication.

i. Your project work is kept by the Computer Science Department. It is available for inspection by anyone, throughout the world. Each graduate project work bears the signature of this department. Therefore, your project work must be written to a standard consistent with published technical work in professional publications, such as: conference proceedings, publications, and scholarly journals.

ii. A successful demonstration of the software product you have produced is clearly very important; a project work is not complete without such a demonstration. However, the demonstration is seen only by the Defence Panel. In reality, you earn your degree with the project report (documentation), not with the demonstration. Many students spend more time and energy in getting the demonstration ready and not nearly enough in writing the project report (documentation). That is a poor choice that often results in delayed graduation. It is important that you schedule enough time for writing the project Documentation.

iii. By the time you start writing the Project Documentation, you should have acquired sufficient writing skills in English. The preparation section, on the following page, provides some ideas on how you might accomplish this.

2. PREPARATION

i. Writing your graduate project work is no simple task. It takes months of preparation and thorough hard work. You need to work closely with your supervisor in getting the project ready forreview and defense. It is not **uncommon** for a student to produce 4 to 5 drafts before arriving at a copy ready for distribution to the defense panel.

ii. The project must be written in grammatically correct English and be easy to read. Do not expect your supervisor to copy edit your work. S/he is there to give you guidance on technical issues on project writing, such as: project outline and topics to be covered. S/he is not there to provide you lessons in writing English. If the draft you submit to the supervisor is not of reviewable quality, then s/he may return it for you to revise and resubmit.

iii. Do not ask the reviewer/Supervisor or any external consultant to write the project for you. That constitutes academic dishonesty. Any level of academic dishonesty can have severe consequences, including the need for you to start your project over with on a new topic and a new proposal.

iv. Carefully consider the word processing and other utility tools, such as grammar and style checkers, that you will be using to write the thesis. Arrange to learn the techniques for using the tool effectively. Tool issues are addressed further in section 7.

3. PROJECT OUTLINE - BRIEF CREATE AN OUTLINE

Plan out the project chapters and create an outline listing the chapters you will have in the project. A suggested list of chapters appears below. This is preliminary; you can change the chapter list as the project develops. Suggestions for what should be covered in these chapters appear in **Section 5, DETAILED STRUCTURE AND CONTENTS OF THE PROJECT.**

3.1 **Abstract**: A summary of the objectives and accomplishments. Typically 1 page long.

3.2 **CHAPTER ONE: INTRODUCTION:** Describe the background of the project work. Establish the context. Discuss why this problem is important.Describe the problem that you set out to solve and the solutions you have achieved. Briefly describe the development process you will follow.

3.3 **CHAPTER TWO:LITERATURE REVIEW:** Provide a survey and a critical review of related prior work.

3.4 **CHAPTER THREE: ANALYSIS AND REQUIREMENTS:** Describe the problem analysis, enhanced with an analysis model in UML. Specify the resulting set of system level and software level requirements.

3.5 **CHAPTER FOUR: DESIGN:** Describe the architectural design and the detailed design enhanced with UML model diagrams. Describe your rationale for the design decisions with supporting data collected from trade-off studies. Describe the specific tools and techniques used in subchapters.

3.6 **CHAPTER FIVE: IMPLEMENTATION & TESTING:** Describe the implementation approach. Describe software reuse, design patterns, special coding techniques, etc. Describe special tools used, if any. Describe the testing approach. Describe sample test plans and test results.

3.7 **CHAPTER SIX:CONCLUSION & RECOMMENDATION:**

3.8 **APPENDICES:**

4. MASTERY

4.1. Successful completion of a graduate Project demonstrates that you have the ability to analyse and develop solutions to a problem of significant complexity and stature. Simply producing a software application, using a run-of-the mill method and an ad-hoc process, does not demonstrate mastery worthy of a graduate project. Through your project, you must demonstrate mastery of the current software engineering and computer science disciplines. Use of current techniques and technologies in completing the project work is important.

4.2. In writing the project, you need to describe the problems and the solutions in an organized and clear manner. You should use standard software engineering and computer science nomenclature. If in doubt, consult the IEEE Standard Glossary of

Software Engineering Terminology (No 610.12-1990) or the Software Engineering Book of Knowledge (SWEBOK). Both of these references are available online.

4.3. Today, UML is accepted as the modelling language of choice in both computer science and software engineering. Therefore, use of standard UML is strongly recommended. Avoid using your own ad-hoc drawing conventions. If you don't know UML, then learn it. Attend a class, or take one of the many online tutorials available in the Internet. At the computer Science department, UML modelling is introduced in **CSD 215**; advanced UML is covered in CSD 204. Produce model elements that adhere to the core UML syntactical and semantic rules. This will allow the reader to read and interpret the meanings of the drawings in the analysis and design models.

4.4. Strongly recommend using a modern, UML capable CASE to create the UML model diagrams. A CASE tool will make it easy to create the model diagrams, maintain a consistent model, and will (often) guard against breaking UML syntactical and semantic rules. There are many free CASE tools available, such as: Visio-2010, ArgoUML, Visual Paradigm, and Poseidon. Find one CASE tool that you like, learn it well, and use it capture the analysis and design models. Use it as a CASE tool, not as a drawing tool, i.e., follow the rules of UML. Many CASE tools provide a capability for generating reports describing the model information captured in the tool. With some editing, you can incorporate these reports into your project.

5. OUTLINE, DETAILED STRUCTURE AND CONTENTS OF THE PROJECT

In general, a graduate project should have the following chapters and sections. Some chapters are mandatory; others will depend upon the nature of the work. Each chapter should elaborate on one major concept, such as prior work, design, implementation, testing, tools used, etc.

Abstract

Although it appears first, plan on writing the abstract last. The abstract is the most difficult part of the project to write, and it is the part most readers of the project will read first. The abstract should be very well written. It should be *clear, easy to read, and to-the point*. The abstract conveys the most important messages regarding your project, such as: what you set out to do? How did you do it? What results were obtained? ***You will have a much better shot at writing a good abstract after you have completed all the other parts of the project.***

CHAPTER ONE: INTRODUCTION

This chapter is mandatory and, at a minimum, should cover the following topics:

1.1. Background: Set the scene by providing a **background** for the work. Why is this work important or interesting?The student will present a clear idea of what needs led up to proposing this project.

1.2. Problem Statement/Description: Introduce the reader to the particular problem your project is attempting to solve. Talk about the "nature" and "scope" of the problem. You should also include a statement of the specific problem or problems that the project addresses. (What is the underlying issue that led to proposing this project? Is there a strong need for the application, or is there a gap in the knowledge of how to apply a given algorithm or methodology?)

1.3. Objectives: Here, you include a statement of the general objectives in doing this project. What is the Project proposing to accomplish here? (Are you trying to apply a new technology? Are you trying to see if a new algorithm works? Or . . . what?). the Project Objectives is divided into two;

- **1.3.1. Global/Main Objective:** The main objective is an overall statement of the thrust of your study
- **1.3.2. Specific Objectives:**Most projects have multiple objectives. For ease of cross referencing, it is a good idea to state these in a numbered list. Each objective should contain only one aspect of the Study.*Always use action oriented words or verbs when writing objectives*. *In general, the objectives stated in the project should match those stated in the project proposal. If there are substantial differences then file a revised proposal.*

1.4 Scope: The *Scope Statement* is an essential element of any project. It is used as a written confirmation of the results your project will produce and the constraints and assumptions under which you will work.

1.5 Justification: Project Justification is about trying to explain why we need to implement a particular solution to the problem we have narrated above. We need to tell the readers why this is the best solution to address the problem.

1.6 Methodology: Write a summary of the overall approach. Include brief descriptions of the development process, design, implementation, and testing approaches.The tools and technologies you used should be mentioned here but described and discussed in later, in a chapter dealing with the technical details.

1.7 Organization of The Project: Provide a synopsis of what the other chapters contain. These descriptions should be very brief, one or two sentences for each chapter. *By the time the reader has finished reading the Introduction s/he should have a clear understanding of the problem you set out to address and how it has been solved.*

CHAPTER TWO: REVIEW OF LITERATURE

This chapter is mandatory and is different from the background provided in the Introduction. The background provides general information.

2.1 Introduction: this section should introduce your readers to the content of this chapter. The literature review focuses on issues that are more specifically related to the work in your project.

2.2 Review of theories/concepts: Describe the development process you followed. To demonstrate your mastery in software engineering and computer science, your project should follow a standard software development process, rather than an undefined or ad hoc process. *Generally, a process framed on the agile development philosophy works well for graduate projects in software engineering and computer science.*

2.3 Review of existing system: Describe similar work done by others in the past and described in the literature. If you cannot find prior work in the literature, then it is most likely that the work you are describing is too simple to qualify as a graduate project.

Your project needs to demonstrate that you have done a literature search and completed a critical analysis of the relevant literature describing prior work in the field. Demonstrate this by writing some discussions on what others have done, what they have achieved, and limitations of their work. If they exist, then provide reviews of prior work in the literature, this shows that you have done a comprehensive literature search.

NOTE:

i. *Do not copy and paste text from the literature; paraphrase the contents in your own words.*
ii. *References must be cited here in the introduction and everywhere else in the thesis. Do not just provide a number like [23]. Say something about the work.*

Example: Jones and Bartlet reports that use of Agile processes reduced mean development time in graduate projects by approximately 12% [23]. Kissinger opined that these results need verification with a wider sample. He pointed out that the Jones and Bartlet study was based on results from only six projects [32]. A follow-on study by Jones and Bartlet included over 30graduate projects and substantiated the results reported in the original study [24]. Anindependent study by Reifer, involving 27 industrial projects, claims development cost saving of 13% attributable to the use of Agile methods [42].

TECHNICAL CHAPTERS (CHAPTER THREE AND CHAPTER FOUR)

In general, the chapters described in the following subsections are expected in a project describing a graduate project. Some of the listed chapters may not be applicable to your project, and additional chapters covering special topics may be needed. Seek guidance from your Supervisor on the chapters your project should contain. A good approach is to describe each major concept/task in a separate chapter, and describe minor related concepts in sections/subsections within the chapters.

CHAPTER THREE: ANALYSIS AND REQUIREMENTS

3

3.1 Introduction **and requirement analysis:** Introduce your readers to the operations of the current or existing system. Provide analysis models, not just words. Some suggested model elements are: use cases, activity diagrams, sequence diagrams, and domain models.
The analysis models should express the system architecture and the top-level behavioral requirements. Don't provide a superficial model with just one or two context level use case diagrams.

3.2 **Proposed System:** Given a description of the new or proposed and how it differs from the existing or current system

3.3 **Requirement Gathering:** Describe how you did requirements elicitation, conducted the analysis, and arrived at the specified requirements.

3.3.1 **Functional Requirements:** Describe a detailed list of things the system must be able to do

3.3.2 **Non- functional Requirements**: Describe the properties or qualities that the system must have.

3.4 **Model of the existing system**

3.5 **Chapter summary/Conclusion:**

CHAPTER FOUR: SYSTEM DESIGN

4.1 Introduction to Design: Generally describe the architectural design model and the detailed design model. Always, discuss the alternatives considered and the rationale for the choosing the solutions you adopted.

4.2: Detailed Design: Describe the architectural and detailed design models in a disciplined manner using both text and comprehensive design models, ideally expressed in UML. Use of UML is highly recommended over using ad hoc or older modelling notations. Suggested UML design modelelements are: class diagrams, interaction diagrams, structured classes, components, subsystems, and deployment models. Produce the model diagrams with a modern CASE tool, not drawing tools.

Provide a comprehensive design model with sufficient design information, not just one or two top level model diagrams. Note that to describe a design adequately you must describe both its static view and the dynamic view. The static view includes elements such as: classes with inheritance and aggregation, structured classes, interfaces, components, subsystems, and deployment. The dynamic view includes: activity diagrams, sequence or communication diagrams, and the state model, when appropriate. Remember that, in most projects, the design model is the main aspect of your work, and it deserves a good deal of your attention.

4.2.1 Interface Design

4.2.1.2 Input /Output Requirements

4.2.1.2 Files and Databases: Describe how files are organized (Sequentially, directly, or by another method). Also describe he format of the records making up the data files. Also discuss the table structure definitions and relationships as well as security for the database design.

4.3 Conclusion

CHAPTER FIVE: IMPLEMENTATION & TESTING

5.1 System Implementation: Describe the overall strategy for implementation tasks, such as incremental builds, risk mitigation measures. Discuss the reasons why you chose the specific programming language, development tools, testing tools, and the implementation platform.

Discuss strategies for reuse of existing products and components. Use of design patterns in the implementation demonstrates sophistication in the subject matter and is highly encouraged. Generally, you do not need to provide source code in the project, unless that code is central to your project, e.g. if you created new design patterns and need to describe the logic of those design patterns using code. However, note that describing design logic using detailed design models demonstrates a higher level of expertise than using code to do the same.

5.2 Testing& Validation: Describe how testing and validation tasks were performed.

5.2.1 Unit Testing: Describe the plans and strategies used in unit testing

5.2.2 Integration Testing: Describe the plans and strategies used in Integration testing

5.2.3 System Testing: Describe the plans and strategies used in system testing. Address regression testing if possible. Describe the test plans and provide test procedures for testing the critical functions.

Describe the test tools you used. Whenever possible, involve someone else, such as friends and colleagues, in the testing and verification process, and include their comments and observations.

If your project serves an external customer then you must involve end users, selected by the customer, in the testing process. Examples of such projects are: community service projects, project from your place of work, or projects with an external sponsor. For graduate projects involving the end users in the testing serves as an acceptable validation process.

5.3 Conclusion

CHAPTER SIX:CONCLUSIONS & RECOMMENDATIONS

6.1: Conclusion: In the conclusion chapter summarize the problem you set out to solve and describe what you have achieved. Address how you have met the original objectives of the project (i.e. proposal contents). Always refer back to the main body of the project for the detailed descriptions; the conclusion section should not contain detailed descriptions of the problems or the solutions.
6.2 Lessons Learned: Refer back to the problems you encountered and how you overcame those, or found workarounds.
6.3 Recommendations: Discuss potential future work.

APPENDICES

Use Appendices to present material that will interrupt the flow if included in the main body of the project. Typical contents of appendices include:

- Code
- data tables
- detailed analysis and design models
- If a user manual is called for, then provide it in an appendix.

BIBLIOGRAPHY

1. Every citation made in the body of the project must appear in the Bibliography. Similarly, every item listed in the Bibliography must be cited in the body of the project.
2. Your supervisor may use the list of references as a yard stick to assess how well you have researched the field before setting out to do your project. The supervisor may look for completeness and also accuracy of the references. Error in the bibliography will need to be corrected before a project is approved.
3. Follow a single standard method for citing and listing both the print references and the online references. There are many different formats for citing and listing references, such as: APA, MLA, ACM style, IEEE style, etc. The department has approved the Harvard style of referencing for you projects.
Note that there is a standard method for listing online references, listing just the URL is notSufficient.
4. Today, most print publications, such as journal articles and conference papers, also exist in the web, typically in digital libraries or in the author's web sites. When listing references to suchprinted sources, provide references to the original printed source, not the online sources.
In general, publications that are not peer reviewed, such as blogs, are not credible sources of reference. Citations from Wikipedia are marginally acceptable but should be avoided if possible.
This is because the Wikipedia review process is not as well controlled as in professional journals and in proceedings from conferences organized by professional institutes. Technical publications from well-established and recognized organizations like IBM, Microsoft, Apple, Oracle, Motorola, etc. are generally acceptable.
You should have read every referenced item you list in the bibliography and every item in the bibliography should be appropriately cited in the body of the thesis.
7. If you used the Citation and Bibliography Tools, then you can auto generate the bibliography.This will save you a lot of work and time.

OTHER SECTIONS

The following sections are highly desirable, because they greatly improve the readability of the thesis.
i. Glossary of terms.
ii. List of abbreviations.

2. Provide headings for all chapters, sections and subsections and prefix each heading with asection number. This is easy to do if you use a style based template in Microsoft Word or Open Office Writer.
3. For improved readability the headings should be left justified rather than centre justified.
4. Number all figures and tables. It is best to use a localized numbering scheme where the figure/table numbers are prefixed with the section number. This makes it easy to locate the item when referenced from elsewhere within the document.

4.1. Examples:
The two first figures in section 2.0 will be numbered Figure 2.0-1 and Figure 2.2-2.
Similarly Figure 3.1-3 would be the 3rd figure in section 3.1. Table 3.3-3 would be the 3rdtable in section 3.3.
5. Provide captions for all figures and tables.
Example: Figure 2.3.1: The System Context Diagram.
Itis common for figure captions to appear below and table captions to appear above.
6. If you are using Word or Open Office Writer, then use ***styles*** to mark the figure and tablecaptions. This will enable you to auto-generate the List of Figures and the List of Tables.
7. Figures, tables, and their associated captions should be centrally justified.
8. If you are providing a list of items, then use a numbered list, not bulleted list. It is easier to reference an item in a numbered list.
9. Follow a consistent method for citing references such as: the APA, MLA, ACM style, or IEEE style.
9.1. If you are using Microsoft Word, or Open Office Writer, then consider using the **Citation and Bibliography Tool**. You will then be able to generate the bibliography automaticallyin APA or MLA style.
All figures and tables should be referenced in the body of the project using their captions.
12.1. Be mindful of the language you use.
When referencing a figure, write, **“The initialization logic is illustrated in Figure 3.2-1: System Initialization”** rather than **“Figure 3.2-1 illustrates the initialization logic”.** Why? A figure, by itself, does not “illustrate anything”,rather “something is illustrated in a figure”. Use a similar language when referencing a table.
13. If you need to include diagram or illustration copied from some other document, then cite the source with a reference number in the Figure caption and also in the text that introduces the diagram or illustration.
13.1. Example: Harper reports the results from a study that examined the relationship between software structural complexity and software quality [31]. The relationship is illustrated in Figure 3.4-1: Design Complexity and Residual Defect Density.
13.2. The figure caption would read
Figure 3.4-1: Design Complexity and Residual Defect Density [31].

14. Do not copy and paste diagrams that are blurry. Redraw the diagram instead. You should cite references, even if you redraw the diagram, unless you make substantial changes. This guideline applies to all cut-and-paste items, including graphs, and tables.
14.1. Do not include any art work, such as photographs, that are blurry, or difficult to interpret.

DEVELOPMENT PROJECT REPORT

A sample table of contents for the final report for a development project is shown below

PRETEXT

i. Approval Cover Page
ii. Abstract
iii. Acknowledgement
iv. Keyword
v. Table of Content (including Tables, Figures and Appendices)

1. CHAPTHER ONE: INTRODUCTION

1.1 Description of the Problem
1.2 Project Objectives
1.3 Justification
1.4 Development Environment (software and hardware)
1.5 Operational Environment (software and hardware)
1.6 Project management
1.7 Team management

2. CHAPTHER TWO: LITERATURE REVIEW

2.1Review of theories
2.2 Review of existing system
2.3 Chapter summary

3. CHAPTHER THREE: SYSTEM ANALYSIS AND DESIGN

3.1ANALYSIS
3.1.1 Introduction to Requirements Analysis
3.1.2 Characterization of the current system
3.1.3 Requirement Gathering
3.1.3.1 Functional Requirements
3.1.3.2 Non-functional
3.1.4 Model of the existing system

3.2DESIGN
3.2.1 Introduction to design
3.2.2 Model for the proposed system
3.2.3 Prototypes
3.2.4 Interface design
3.2.5 Database design

4. CHAPTHER FOUR: IMPLEMENTATION AND TEST (PLAN AND RESULTS)

4.1 Introduction to Coding and implementation
4.2 Code testing

4.3 Usability testing
4.4 Browser testing (if necessary)
4.5 Implementation technique

5. CHAPTER FIVE CONCLUSION, SUMMARY AND RECOMMENDATION

5.1 Findings
5.2 Goals achieved
5.3 Lessons learned
5.4 Recommendations for enhancements (further works)

REFERENCES (citing all references used)

APPENDICES

a) User manual
b) Maintenance manual