

KINGDOM OF SAUDI ARABIA

Ministry of Education

Taibah University

College of Computer Science and

Engineering

(Male Section)



جامعة طيبة

كلية علوم وهندسة الحاسب الآلي

(قسم الطلاب)

**AI-Powered Requirements Analysis Modeling**

**Graduation Project 1**

**by**

Ali Talal Alahmadi 4100379

Mohammed Hadi Alharbi 4101704

Rami Ramadan Al-Mohammadi 4101758

**A project submitted in partial fulfilment of the requirements for the degree of Bachelor of Science Computer Science**

**Supervised by**

Dr. Mohammad M. Alsuraihi

1st Semester - Academic Year 1445 (2023/2024)

# Abstract

The abstract should identify clearly and succinctly the purpose of the project, the methods used, the results obtained and the significance of the results or findings. The abstract must not exceed one page. Abstract section gives the readers a brief idea about your project, which present in brief your problem statement and how you can solve it.

Indent accordingly when you start a new paragraph. The abstract section should not exceed a single page.

**Keywords** Word1; Word2; Word3; Word4; Word5

# Acknowledgement

The content of this single page is left to the discretion of the student. It is suggested however that the page makes reference to guidance received by the student from his or her supervisor and other members of the college. Reference should also be made to any financial assistance received to carry out the project. Any extraordinary assistance received by the student for example in word processing, data collection, data analysis, and so on, should be properly acknowledged. Example acknowledgement can be found in books, reports and also papers. The acknowledgements should not exceed 250 words.

**Contents**

[Abstract ii](#_Toc151301917)

[Acknowledgement iii](#_Toc151301918)

[List of Figures vii](#_Toc151301919)

[List of Tables viii](#_Toc151301920)

[List of Abbreviations ix](#_Toc151301921)

[1 Chapter 1: Introduction 1](#_Toc151301922)

[1.1 Introduction 1](#_Toc151301923)

[1.2 Problem Definition 1](#_Toc151301924)

[1.3 Project Aim and Objectives 2](#_Toc151301925)

[1.4 Project Methodology 2](#_Toc151301926)

[1.5 Project Timeline 3](#_Toc151301927)

[1.6 Document Organization 5](#_Toc151301928)

[1.7 Summary 6](#_Toc151301929)

[2 Chapter 2: Literature Review 7](#_Toc151301930)

[2.1 Introduction 7](#_Toc151301931)

[2.2 Research Methodology 7](#_Toc151301932)

[2.3 AI 10](#_Toc151301933)

[2.4 Machine Learning 11](#_Toc151301934)

[2.4.1 Neural Networks (NNS) 11](#_Toc151301935)

[2.4.2 Supervised Learning 12](#_Toc151301936)

[2.5 NLP 13](#_Toc151301937)

[2.5.1 Named Entity Recognition 15](#_Toc151301938)

[2.5.2 Parsing 16](#_Toc151301939)

[2.5.3 Pronoun Resolution 18](#_Toc151301940)

[2.5.4 Semantic Analysis of Requirement Using NLP and ML 19](#_Toc151301941)

[2.6 System Development for AI 20](#_Toc151301942)

[2.6.1 NLP application Development 21](#_Toc151301943)

[2.6.1 ML application Development 22](#_Toc151301944)

[2.7 Similar Tools 24](#_Toc151301945)

[2.7.1 Lucidchart and Lucidspark 24](#_Toc151301946)

[2.7.2 Visual paradigm 25](#_Toc151301947)

[2.7.3 StarUML 26](#_Toc151301948)

[2.7.4 Enterprise Architect 27](#_Toc151301949)

[2.7.5 MagicDraw 28](#_Toc151301950)

[2.8 Summary 29](#_Toc151301951)

[3 Chapter 3: System Analysis 30](#_Toc151301952)

[3.1 Introduction 30](#_Toc151301953)

[3.2 Analysis of Existing Systems *(Optional)* 30](#_Toc151301954)

[3.3 Requirements Elicitation 30](#_Toc151301955)

[3.3.1 Functional Requirements 30](#_Toc151301956)

[3.3.2 Non-Functional Requirements 31](#_Toc151301957)

[3.3.3 User Requirements 31](#_Toc151301958)

[3.3.4 Domain Requirements 31](#_Toc151301959)

[3.4 Requirements Specification 32](#_Toc151301960)

[3.5 Developmental (or Research) Methodology 32](#_Toc151301961)

[3.6 Summary 33](#_Toc151301962)

[4 Chapter 4: System Design 34](#_Toc151301963)

[4.1 Introduction 34](#_Toc151301964)

[4.2 Architectural Design 34](#_Toc151301965)

[4.3 Object Oriented Design 34](#_Toc151301966)

[4.3.1 Structural Static Models 34](#_Toc151301967)

[4.3.2 Dynamic Models 34](#_Toc151301968)

[4.4 Data Modeling 34](#_Toc151301969)

[4.5 User Interface Design 34](#_Toc151301970)

[4.6 Summary 34](#_Toc151301971)

[5 Chapter 5: System Implementation 35](#_Toc151301972)

[5.1 Introduction 35](#_Toc151301973)

[5.2 Tools and Languages 35](#_Toc151301974)

[5.3 Mapping Design to Implementation 35](#_Toc151301975)

[5.4 Main/Most Important Codes 35](#_Toc151301976)

[5.5 System Testing 35](#_Toc151301977)

[5.6 Results and Discussion 35](#_Toc151301978)

[5.7 Summary 36](#_Toc151301979)

[6 Chapter 6: Conclusion and Future Work 37](#_Toc151301980)

[6.1 Conclusion 37](#_Toc151301981)

[6.2 Goals Achieved 37](#_Toc151301982)

[6.3 Limitations and Future Work 37](#_Toc151301983)

[7 References 38](#_Toc151301984)

[8 Appendix A 42](#_Toc151301985)

[9 Appendix B 43](#_Toc151301986)

[1 Appendix C 45](#_Toc151301987)

# List of Figures

[Figure 1: Project Methodology 3](#_Toc150802983)

[Figure 2: Methodology Schema 9](#_Toc150802984)

[Figure 3: AI branch. [7] 11](#_Toc150802985)

[Figure 4: Neural networks architecture. [12] 12](#_Toc150802986)

[Figure 5: Supervised learning process [13] 13](#_Toc150802987)

[Figure 7: Word2Vec architecture: (a) CBOW; and (b) skip-gram. [15] 14](#_Toc150802988)

[Figure 6: RNN model architecture. [15] 16](#_Toc150802989)

[Figure 8: Examples of experimental results: (a) original text; and (b) NER results. [15] 16](#_Toc150802990)

[Figure 9: Constituency and dependency structures for the sentence ‘the man hit the ball’. [16] 17](#_Toc150802991)

[Figure 10: An example of a non-projective parse tree. [16] 17](#_Toc150802992)

[Figure 11: difference between "original sentence" and "sentence with resolved Coreference". [18] 18](#_Toc150802993)

[Figure 12: identify potential spans. [18] 19](#_Toc150802994)

[Figure 13: group spans. [18] 19](#_Toc150802995)

[Figure 14: replace pronouns with real-world entities. [18] 19](#_Toc150802996)

[Figure 14: replace pronouns with real-world entities. [25] 28](#_Toc150802997)

# List of Tables

[Table 1: Project plan v1 4](#_Toc149585906)

# List of Abbreviations

HCI Human Computer Interface

# Chapter 1: Introduction

## Introduction

Nowadays, the field of software development is using the capabilities of artificial intelligence to improve the phases of this field, including prediction and decision-making assistance. Artificial intelligence has become instrumental in guaranteeing the high quality of software development phases. Additionally, it has the capacity to significantly reduce the time needed for software development, leading to increased overall productivity. Furthermore, AI can effectively detect and pinpoint errors within the development process, enabling timely rectification. It also contributes to the improvement of specific components within various phases of software development. [1]

In this project, we pursue to merge knowledge from two major disciplines of computer science: Artificial Intelligence and Software Engineering in order to facilitate software development. Particularly, employing the power of artificial intelligence to transform project analysis textual elements: scenarios and requirements into visual modeling representations (Usecases and Class Diagrams).

This chapter highlights the goal and objectives of this project, outlines our methodology to achieve these objectives, gives the plan for the project’s tasks, provides an overview of the subsequent chapters in this report, and finalizes with an overall summary of the chapter.

## Problem Definition

Software maintenance constitutes a pivotal stage within the software development lifecycle, encompassing a substantial portion, varying from 40% to 80% of the total expenses associated with software development. It's worth highlighting that a notable 60% of the overall maintenance expenditure is dedicated to the improvement of existing software functionalities. Consequently, it becomes imperative to meticulously prepare appropriate software documentation at each developmental phase in order to alleviate the financial burdens of maintenance. The crux of reducing maintenance costs lies in enhancing one's grasp of the software system, as understanding a software system accounts for roughly 50% of the time spent in the maintenance phase. To facilitate this understanding, various modeling languages have surfaced, enabling graphical representations that substantially contribute to an enhanced comprehension of software systems. [2]

The project may face issues with excessive resource consumption, such as time or costs, and frequent changes in requirements can lead to project delays and increased expenses. Additionally, sometimes the techniques or tools used in the analysis and design process are ineffective or outdated.

Using artificial intelligence (AI) to analyze requirements and generate drawings or models can be an effective solution for saving time and costs in the design and analysis processes.

## Project Aim and Objectives

The aim of the project is to complete the analysis, modeling, and data gathering for a system that converts textual requirements into use cases and class diagrams, with the aim of expediting and facilitating the work of systems modeling analysts and saving them time.

To achieve this goal, we must achieve the following objectives:

1. To review previous works that can be related to our project domain.
2. To finish the Analysis phase of the proposed system to be built.
3. To do the preliminary Design tasks of the suggested system.
4. To conclude with the lessons learnt and knowledge and experience gained from working on this project.

## Project Methodology

Reviewing previous works is a fundamental step in our project's research phase. By examining existing research and projects in our project domain, we can gain valuable insights into the state of the art, identify potential challenges, and build upon the knowledge and findings of others. This process allows us to ensure that our project is both innovative and informed by the latest developments in the field.

Completing the Analysis phase is a critical milestone in our project's development. During this phase, we thoroughly examine the project requirements, gather user needs, and analyze existing systems and processes. It provides the foundation for all subsequent project activities, ensuring that we have a clear understanding of what needs to be achieved and how to achieve it.

After completing the Analysis phase, we move on to the preliminary Design tasks. This phase involves translating the gathered requirements and insights into a comprehensive system design. We will define the system's architecture, user interfaces, data flows, and functionality. This step is pivotal in transforming our project concept into a concrete plan for implementation.

Concluding the project with a comprehensive review of lessons learned, knowledge gained, and experiences accumulated is essential for continuous improvement. This reflection allows us to identify areas of success and areas for improvement in project management, technical implementation, and collaboration. By documenting our insights and best practices, we can enhance our future projects and contribute to the growth of our team's expertise.



Figure 1: Project Methodology

## Project Timeline

This section presents the timeline plan versions of our project as the following:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **December** | | | | **November** | | | | **October** | | | | **September** | | **Months**  **(2023)** |
| **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **Weeks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Define project objectives and identify the problem | **Tasks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Review related literature and existing systems |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Analysis and document system requirements |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Design the system structure and interfaces |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Documentation |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Presentation |

Table 1: Project plan v1

1. **Defining Project Objectives and Identifying the Problem (1 week):**

This time is needed to understand the project scope and specific objectives, as well as to analyze the problem thoroughly.

1. **Reviewing Relevant Literature and Theories (4 weeks):**

Allowing this time enables thorough research and review of literature related to the project, including studying relevant theories.

1. **Analyzing and Documenting System Requirements (4 weeks):**

More time is allocated to this stage as it involves a detailed analysis of functional and non-functional system requirements and their precise documentation.

1. **Designing System Structure and Interfaces (3 weeks):**

This time allows for the precise design of the system's structure and user interfaces, including working on their details.

1. **Documentation:**

This task spans throughout the project's duration as it involves continuous documentation of the work.

1. **Project Presentation:**

The project should be presented during this time to showcase the results.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **December** | | | | **November** | | | | **October** | | | | **September** | | **Months**  **(2023)** |
| **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **Weeks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Define project objectives and identify the problem | **Tasks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Review related literature and existing systems |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Analysis and document system requirements |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Design the system structure and interfaces |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Documentation |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Presentation |

Table 1: Project plan v2

1. The presentation has been rescheduled to take place between the second and third weeks of December instead of the third and fourth weeks of December. This is the specified date from the university for the presentation.
2. The third task of the project will now commence from the third week of November until the first week of December, instead of from the fourth of November to the second of December. This adjustment is due to the modification in the presentation date.
3. In the third week of November, we will be nearing the completion of system analysis and will start the architecture design, connecting the analysis and design phases.

## Document Organization

* **Chapter 1.** In this chapter we discussed the plan and objectives and identified the problem that the project solves.
* **Chapter 2.** In this chapter, we reviewed the literature and learned about the techniques used. We reviewed and compared systems similar to ours.
* **Chapter 3.** Our focus in this chapter was on identifying functional and non-functional requirements using UML diagrams, as well as identifying user requirements and search methodologies.
* **Chapter 4.** In this chapter, we designed the system, identified the components of the system, modeled the data, and designed the user interfaces.
* **Chapter 5.** A presentation of the project's conclusion was given, along with a proposal for new work to improve the current work, and a statement of whether the original objectives of the project had been met.

## Summary

In this chapter, we obtained a general idea of the challenges we face in analyzing project requirements to transform them into Use Case and Class Diagrams. We will build a model that relies on artificial intelligence to save time.

In the second chapter, we will delve into related work and similar solutions, discussing them in detail and compiling all relevant information.

# Chapter 2: Literature Review

## Introduction

In this chapter, we meticulously explore the current scholarly landscape within our field of study. Our primary objective is to conduct a precise examination of the existing literature and research pertaining to our subject matter. We will study and analyze the methodologies and techniques employed in this context.

Furthermore, we will undertake a comprehensive review and comparative analysis of systems and research akin to our project. Through these reviews and comparisons, we aim to extract valuable insights that contribute to understanding the historical and contemporary developments in our field, and elucidate the patterns, trends, and innovative approaches that have shaped the discourse and research.

By conducting a meticulous evaluation of prior works and engaging in a profound comparison with our research approach, we aim to direct our efforts towards delivering novel contributions and deepening our understanding of our study's domain.

## Research Methodology

Our data collection process involves using the Google Scholar search engine to find diverse scientific literature sources, such as research papers, conference papers, articles, and e-books that can assist us in understanding similar studies and identifying their strengths, weaknesses, similarities, and differences. For citation and referencing, we use digital libraries from IEEE, ACM, the university library, trusted websites (e.g., .org, .edu), and the Saudi Digital Library. Additionally, we use ChatGPT to assist us in translating and comprehending texts.

1. **IEEE** [3]**:** The IEEE style is a numerical style, where citations are numbered according to their appearance order. This number guides the reader to a complete reference in the reference list at the end of the work. The citation number should be placed within square brackets on the same line as the text, before any punctuation, with a space before the square brackets, such as: [x]. Once cited from a source, the same number is reused for all subsequent citations from the same source. [4]
2. **ACM Library Online** [5]**:** The ACM Digital Library is a research, discovery, and networking platform that offers full-text access to all ACM publications, including journals, conference proceedings, technical magazines, newsletters, and books. It also maintains a carefully curated collection of full-text publications from a limited number of publishers. The ACM Guide to Computer Literature is a searchable database dedicated solely to computing literature, encompassing a complex network of relationships among authors, works, institutions, and specialized communities.

After more than a week of researching topics related to our project, we have now compiled the main and subtopics as shown in the following figure.



Figure 2: Methodology Schema

## AI

Artificial intelligence (AI) is the branch of computer science concerned with making computers behave like humans (Computers with the ability to mimic or duplicate the functions of the human brain). [6]

Artificial intelligence systems encompass individuals, processes, hardware, software, data, and the essential knowledge required to develop computer and machine systems that exhibit intelligent characteristics. [6]

**AI conduct:**

* Learn from experience.
* Apply knowledge acquired from experience.
* Handle complex situations.
* Solve problems when important information is massing.
* Determine what is important.
* React quickly and correctly to a new situation.
* Process and manipulate symbols.

**Techniques used in AI:**

1. **Knowledge Representation**: It is the science of translating real-world knowledge into a form that can be used by computers.
2. **Search:** A technique for selecting the best solution from all possible solutions.
3. **Automated Reasoning:** The process of achieving a specific goal based on prior knowledge.
4. **Planning:** The ability to make a good sequence of actions to achieve our objectives.

**AI branch:**



Figure 3: AI branch. [7]

## Machine Learning

Machine science is a branch of artificial intelligence that aims to facilitate human life by replicating human behaviors. It is used to train the machine to train itself and to teach the machine to deal with data accurately. One of its uses is to predict future results based on known or previous data. Some of its types include supervised learning, unsupervised learning, and neural networks. [8] [9]

### Neural Networks (NNS)

It is a group of neurons that bind to each other, and its characteristics are self-learning It consists of three layers the input layer, the hidden layer, and the output layer. The input layer receives variables to connect to the hidden layer which could be more than one layer containing neurons. Convolutional Neural Networks (CNNs) are useful and powerful in computer vision and learning and are also a type of neural network. [10] [11]



Figure 4: Neural networks architecture. [12]

### Supervised Learning

The standard formula for supervised learning is that the learner learns from several categories through the input and output examples given to him. It is the most common technique for solving classification problems. One of its algorithms is Naive Bayes, which solves prediction problems, and Linear regression, which trains models in the trained dataset, and then uses the model to predict new data or test data. [13]

**Supervised Learning Process:**

In this process, each dataset stored or sent to the learning algorithm has a label or name identifying this data. Each label has more than one feature. for example, If we take the dog, it is a label that the features are (big ear, big nose, etc..). [13]

Trained data with features and labels are entered as input to the learning algorithm. In addition, correct outputs are entered and the algorithm is learned by comparing correct outputs to actual outputs to find errors. [13]



Figure 5: Supervised learning process [13]

## NLP

Natural Language Processing (NLP) is a branch of artificial intelligence that involves processing and analyzing text data and includes machine learning to understand and interact with human language. [14]

**How does a computer understand languages?**

Computers are emotionless machines, so we must convert natural languages into numbers. This numerical transformation allows the computer to perform mathematical operations on language data, enabling it to comprehend human language. [14]

**Text Tokenization:**

It is a process in which unnecessary words are removed, and the root (or lemmatization) is applied. Each sentence is divided into a sequence of words for analyzing each word separately. For example, "I like study" would be broken down into "I," "like," "study." [15]

**Word Embedding:**

Term Frequency (TF) is the simplest of its kind and involves counting how often each word appears in a document. For example, two sentences, “The Contractor should prepare” and “The Engineer should submit,” would be mapped to [1the, 1Contractor, 1should, 1prepare, 0Engineer, 0submit] and [1the, 0Contractor, 1should, 0prepare, 1Engineer, 1submit]. [15]

Term Frequency-Inverse Document Frequency (TF-IDF) analyzes the importance of prevalent and widely used words, such as("a," "an," "the"(. For example, the two preceding sentences would be mapped to [0.5the, 1Contractor, 0.5should, 1prepare, 0Engineer, 0submit] and [0.5the, 0Contractor, 0.5should, 0prepare, 1Engineer, 1submit]. [15]

The latest word embedding technique is Word2Vec, which assigns words used in a similar context to be close to each other in the vector space. For example, it places "contractor" and "engineer" in a nearby vector space if they are often used together. There are two types of Word2Vec: Continuous Bag of Words (CBOW) and Skip-Gram. CBOW tries to predict the current word from its surrounding words, controlling the number of surrounding words. Skip-Gram, on the other hand, predicts the surrounding words from the current word. According to developers, CBOW is faster, but Skip-Gram provides better word predictions. [15]



Figure 7: Word2Vec architecture: (a) CBOW; and (b) skip-gram. [15]

### Named Entity Recognition

Construction specifications are a document that outlines the necessary requirements for performing work during a project's construction phase. They typically consist of a description and requirements. Because construction specifications require precision in understanding and adherence during project development, failing to comprehend the project can result in wasting time, human resources, and an increase in costs. [15]

NER, a subfield of machine learning-based information extraction methodologies, categorizes specific words such as names, locations, and objects. It is divided into two sections: structural and semantic information. Each word's category can be easily determined. For instance, names typically begin with a capital letter, and words associated with locations follow prepositions like "in," "on," or "to." Objects are usually nouns. Semantic information stands out for its strength and expandability, automatically identifying usage patterns for each word and obtaining evidence from these patterns through machine learning algorithms. [15]

**Recurrent Neural Network for NER:**

It is a form of deep neural network where networks are connected in a sequential structure. Therefore, it can handle sequential input data that may carry multiple meanings, even within the same context. However, there is a critical issue called the vanishing gradient problem. This means that as the length of the sequence increases, the gradient becomes smaller, significantly reducing the model's learning capacity. [15]

To address this problem, two gates were introduced: the forget gate (ft) and the input gate (it). The forget gate's function is to discard previous information, while the input gate's role is to retain essential information for longer periods by enhancing crucial signals and forgetting the non-essential ones. [15]



Figure 6: RNN model architecture. [15]

**Prediction Result of the NER Model:**

The results consist of automatically tagging each word from the original text, and each word is individually assigned to its category. Despite some incorrect classifications, out of 85 results, 75 appeared to match the scientific identifier. [15]



Figure 8: Examples of experimental results: (a) original text; and (b) NER results. [15]

### Parsing

Describing the problems of structural linguistic analysis of sentences through parse trees, the analysis can be divided into two main types: dependency parsing and constituency parsing. Dependency parse trees rely on direct relationships between sentence structures, while constituency parse trees rely on the analysis of formal grammatical rules. [16]



Figure 9: Constituency and dependency structures for the sentence ‘the man hit the ball’. [16]

Dependency parsing is based on separating dependency relationships and linking different symbols directly. Each symbol depends on the main symbol it is associated with, except for the root symbol of the tree. There are two types of dependency trees: projective and non-projective. Projective trees have no crossing arcs, as shown in the previous figure. Non-projective trees, on the other hand, have intersecting arcs, as shown in the following figure: [16]



Figure 10: An example of a non-projective parse tree. [16]

In the development of a natural language text analyzer capable of working with multiple languages, we face a challenge due to the varying linguistic structures between languages. We propose using the Bidirectional Long Short Term Memory (BiLSTM) neural network model to enhance the analyzer's performance across different languages, as this model can handle complexities arising from long-range linguistic structures. [16]

### Pronoun Resolution

Natural Language Processing (NLP) is one of the most challenging branches of artificial intelligence primarily because natural language is full of exceptions and ambiguities that make it difficult for computers to learn. One approach to simplify this process is to eliminate imprecise expressions that require context for understanding. This makes it easier for computers to learn. Examples of such expressions include pronouns like "he" or "she," which can be replaced with specific names referring to them.

Coreference Resolution (CR) is the task of finding all linguistic expressions in a given text that refer to the same entity. We can address this problem by replacing pronouns with noun phrases after identifying these references and aggregating them. [17]

A screenshot of a phone

Description automatically generated

Figure 11: difference between "original sentence" and "sentence with resolved Coreference". [18]

Anaphora resolution (AR) is one of the cases that significantly differ from coreference resolution. An anaphora resolution occurs in the text when one term refers to another term and specifies the interpretation of another. [19]

Despite the distinctiveness of anaphora resolution from coreference resolution, coreference resolution is widely applicable and covers most cases. [17]

**Misleading pronominal references:**

Some cases can be misleading, such as cases where there is no relationship between the pronoun and other words in the text. Examples of these cases include derived sentences where the pronoun 'it' is redundant, and we can easily extract a sentence with the same meaning without using the pronoun 'it'. [20]

**Steps for Coreference Resolution:**



Figure 12: identify potential spans. [18]

A white background with black text

Description automatically generated

Figure 13: group spans. [18]



Figure 14: replace pronouns with real-world entities. [18]

### Semantic Analysis of Requirement Using NLP and ML

During software development stages sometimes the quality of the analysis of the requirements is poor and can cause the failure of the software project. Measuring the quality of the analysis of the requirements is through, inter alia, understanding the requirements by making the terminology understood by different stakeholders and being clear is not ambiguous. [21]

**NLP activities:**

Activities that can be applied using NLP are the selection of requirements, elicitation of requirements, validation of requirements, creation of model, and discovery of ambiguity. [21]

**some of the categorizations used for NLP:**

Syntactic focuses on word analysis. Semantic, which focuses on the analysis of concepts and meaning of language programming using machine learning techniques. [21]

**Pre-processing approaches:**

Pre-processing is the process of preparing requirements by NLP techniques to convert and analyze requirements. [21] Includes 5 core techniques:

* **lexical analysis**: The first step in processing NL is performing a lexical analysis. The purpose of the lexical analysis is to interpret the meaning of individual words.
* **syntactic analysis**: The output of the lexical analysis serves as input to the syntactic analysis. The goal of the syntactic analysis is to uncover the grammatical structure of a sentence.
* **semantic analysis**: The purpose of the semantic analysis is to determine possible meanings of a sentence. Semantic analysis looks at the meanings of the different words in a sentence.
* **Categorization**
* **Pragmatic analysis**

## System Development for AI

In this section, we will discuss the system development for Artificial Intelligence applications and the stages followed in the development of Natural Language Processing (NLP) and Machine Learning (ML) applications, along with the languages used in development.

### NLP application Development

Developing Natural Language Processing (NLP) applications is crucial in the field of artificial intelligence. This work involves several sequential steps to ensure the creation of an effective application that leverages human language understanding. We will take a look at each step to comprehend it. [22]

1. **Data Acquisition:**

In this stage, we aim to acquire a dataset that reflects the linguistic diversity and complexity relevant to the application's subject. This step forms the foundation for understanding language and its varied usage.

1. **Text Cleaning:**

After obtaining the data, it's time to clean it from linguistic errors and distortions. This step is crucial to ensure accuracy and quality of results in later stages.

1. **Pre-processing:**

This part involves an initial analysis of the data and the application of basic natural language processing techniques. This step is fundamental to comprehend the language structure and prepare the data for training phases.

1. **Feature Engineering:**

This step focuses on utilizing features derived from the data in a way that contributes to the understanding of models and enhances their performance.

1. **Modeling:**

Modeling encompasses the training of models using processed data, refining them to achieve greater efficiency in understanding and processing language.

1. **Evaluation:**

After model training, their performance is evaluated using various tests to ensure their effectiveness and robustness in dealing with a wide range of scenarios.

1. **Deployment:**

Following model evaluation, the application is deployed for widespread use and application in real-world contexts.

1. **Monitoring and Model Updating:**

In this step, the application's performance is monitored post-deployment, and models can be updated based on changes in data or usage requirements.

**Using Python for Natural Language Processing (NLP):**

There are several options for tools and technologies when developing a natural language processing application, but we'll focus on Python code that utilizes the spaCy NLP library. [14]

There are many reasons to use Python and spaCy, including:

**Simplicity:** Python is an easy-to-learn language, making it suitable for newcomers to software development. It allows for writing straightforward code, especially compared to other languages.

**Community Support:** Python benefits from a large and active community. This means that solutions to problems can be found, educational resources are readily available, and assistance is just a question away.

**Libraries:** Python is renowned for its extensive libraries, particularly in the field of artificial intelligence. These libraries simplify the development of natural language processing applications (such: NLTK, spaCy, TextBlob, genism).

### ML application Development

Developing machine learning applications is an exciting challenge in the modern tech world. These steps represent the journey developers go through to build smart applications based on the learning capabilities of automated systems. we'll overview the main steps in developing these applications and how to achieve effective integration between the process components.

1. **Data Acquisition:**

Developers begin by examining and gathering data that the system will learn from. This data forms the foundation for understanding the application's context and training the model.

1. **Data Analysis and Cleaning:**

Developing machine learning applications requires a thorough examination of the data, ensuring its cleanliness and freedom from errors to ensure quality training.

1. **Feature Selection:**

Identifying key features in the data that will contribute to the system's understanding of information and achieve outstanding performance.

1. **Data Splitting:**

Dividing the data into sets for training and testing ensures an effective evaluation of the model and avoids overfitting problems.

1. **Model Selection:**

Modeling encompasses the training of models using processed data, refining them to achieve greater efficiency in understanding and processing language.

1. **Model Training:**

The model is trained using data, learning and benefiting from patterns and details within the data.

1. **Performance Evaluation:**

The model's performance is assessed using test data to ensure its effectiveness and efficiency in dealing with new cases.

1. **Improvement and Adjustment:**

Based on the evaluation, developers adjust enhance the model's performance and efficiency in facing different challenges.

1. **Application Deployment:**

After ensuring the model's quality, the application is deployed for public use and interaction with users.

1. **Performance Monitoring and Updating:**

Developers continue to monitor the application's performance after deployment, making necessary updates to keep up with changes and ensure continuous improvement.

**Libraries in python:**

1. **scikit-learn**
2. **TensorFlow**
3. **Keras**
4. **PyTorch**
5. **XGBoost**

## Similar Tools

### Lucidchart and Lucidspark

The Lucid platform includes Lucidchart and Lucidspark products, working seamlessly to enhance productivity, clarity, and guidance at every stage of the workflow. [23]

**Lucidspark:**

You and your team are aided in turning your ideas into reality with no limit to what you can do on the infinite whiteboard. Some of its key uses include: [23]

1. Conducting brainstorming sessions
2. Guiding project teams
3. Facilitating collaborative work sessions
4. Creating customer journey maps

And among its features are notes, timer, participant colors, breakout boards, voting on team-favorite ideas, and sorting them to organize your board. [23]

**Lucidchart:**

After discussing the ideas you've generated in Lucidspark, create strong visualizations that all team members can understand. [24]

One of its most important uses is: [23]

1. Building diagrams such as UML, BPMN, ERDS.
2. System engineering planning and network planning.
3. Comparing the current and future states.
4. Allocating resources for the project.
5. Designing low-fidelity prototypes for user testing.

One of its features:

1. Drawing maps more efficiently by automatically generating maps from the data at your disposal.
2. Direct import data into your maps.

**Collaborative AI:**

Collaborative AI enhances collaborative thinking processes, improves team innovation sessions, and supports design thinking. [23] Some uses of Collaborative AI include:

1. **Idea Generation:** Using artificial intelligence to generate new ideas from a set of keywords or a specific topic. [23]
2. **Idea Sorting:** Employing artificial intelligence to analyze a series of notes and categorize them. [23]
3. **Idea Summarization:** Using artificial intelligence to provide a summary of identified notes. [23]

### Visual paradigm

Visual Paradigm is a software application designed for software development teams, used for modeling business information systems, and managing development processes.

One of its AI-powered tools:

1. **code generation:**

This functionality generates code from UML class diagrams, facilitating the transformation of design into executable code. Additionally, you can reverse this process, allowing the conversion of code into a UML model for model extraction or design modification. [25]

It supports various programming languages such as Java, C++, C#, Python, PHP, and others. [25]

One of the most popular IDEs that supports this feature includes Eclipse, Netbeans, IntelliJ IDEA, Visual Studio, and Android Studio. [25]

1. **Intelligent pattern recognition:**

Design pattern is considered a part of the diagram that can be reused in various diagrams to illustrate the Intelligent pattern recognition tool. First, we need to know what is meant by defining design pattern.

**defining design pattern:**

It is a tool like Intelligent Pattern Recognition, but it does not use artificial intelligence. It allows you to define a pattern and reuse it in your project or share it with your team. To apply the pattern, you need to manually define and save it as a ready-to-use pattern file. [26]

**The difference between Intelligent Pattern Recognition and defining design pattern:**

Defining design pattern involves manually defining and saving the pattern, while Intelligent Pattern Recognition utilizes artificial intelligence to automatically analyze and comprehend patterns without human intervention.

1. **Real-time Diagram validation:**

This is a process where changes in the diagram are immediately examined as the user makes any modifications. It checks the validity of the changes and guides the user to any potential errors in real-time.

### StarUML

It is an open-source software modeling application primarily used to support the Unified Modeling Language (UML). [27]

**Some features of StarUML:**

1. **Design Speed:**

Enables users to create elements and link them quickly and efficiently. [27]

1. **Integration with Programming Languages:**

Supports programming languages such as Java, C++, C#. [27]

1. **Versatile Usage:**

Considered useful in modeling diverse applications and systems. [27]

**Among the features that leverage artificial intelligence:**

1. **Creating Use Cases and Class Diagrams:**

Artificial intelligence is employed to guide users during the creation of use cases and class diagrams, providing guidance on relevant elements related to use cases and class diagrams.

1. **Suggesting Relevant Elements:**

Artificial intelligence is utilized to suggest relevant elements based on context.

1. **Detecting Errors and Inconsistencies:**

Artificial intelligence is employed to detect potential errors in the model, contributing to the enhancement of analysis quality by identifying and correcting any contradictions or errors in the design.

### Enterprise Architect

It is an application for software modeling used to design and build software systems, model business processes, depict systems, and current operations. [28]

**Features include:**

1. Model Management
2. Project Management
3. Model Version Control
4. Generate RTF and HTML Documents (Reports)

**Among the features that leverage artificial intelligence:**

1. **Predictive Modeling:**

Utilizes artificial intelligence to analyze data and forecast future trends, aiding in guiding strategic decisions by understanding the potential future of the project.

1. **Risk Assessment:**

Uses artificial intelligence to analyze information and estimate potential risks, assisting in identifying and mitigating potential risks before they impact the project.

1. **Automated Testing:**

Uses artificial intelligence to execute tests automatically, ensuring quality and continuity.

### MagicDraw

MagicDraw facilitates the execution of the software development life cycle, unlike other modeling and architectural environments that rely on UML. [29]

It allows for a focus on business and process modeling, gathering requirements, and design, providing independence from any specific software development process. It is not tied to any stage in your project; you can start from any point, such as requirements definition or maintenance. [29]

And among its features:

1. **Ease of use and quick start:**

It provides a well-designed user interface that allows users to quickly get accustomed to MagicDraw without the need to spend a long time understanding the elements. [29]

1. **Collaboration and team development:**

By using Teamwork Cloud, several people can work simultaneously on the same model. [29]

**Among the features that leverage artificial intelligence:**

1. **Smart Requirements Management:**

Artificial intelligence is used to automatically link requirements with UML diagrams, ensuring an accurate and updated representation of project requirements.

1. **Conflict Analysis:**

MagicDraw can analyze models and identify conflicts between requirements and different models.

1. **Process Coordination Improvement:**

By using artificial intelligence-supported features, teams can enhance the coordination of development processes, contribute to improving model quality, and ensure compliance with requirements.

## Summary

After reviewing the previous systems in this chapter, such as AI and ML, especially in the field of Natural Language Processing (NLP), and examining similar systems related to our project.

we will be ready in Chapter 3 to analyze the system and analyze both functional and non-functional requirements.

# Chapter 3: System Analysis

## Introduction

In this chapter, we will focus on defining the system and user requirements using simple diagrams. We will use UML tools to explain how the system operates and its requirements. We will specify both functional and non-functional requirements and discuss the development methodology. We will also analyze an existing system.

## Analysis of Existing Systems *(Optional)*

This section may contain any of the following information; document review, data collection, user interviews, or questionnaires.

## Requirements Elicitation

Requirement analysis is a phase of software development that is the process of studying and identifying the user's needs for system building that solves a particular problem and includes elicitation, and specifications. Elicitation of requirements is the collection of user and customer requirements to determine system function and requirements. The elicitation process must include all persons directly and indirectly involved in the project. After the elicitation process, we extracted functional and non-functional requirements, user requirements and domain requirements [30].

### Functional Requirements

The requirements determine the system's behavior, what it should and shouldn't provide, and how it interacts with inputs [31].

1. The system should allow the user to input requirements or scenarios.
2. The system must allow the user to choose the type of diagram, whether it is a use case diagram or a class diagram.
3. The system should allow the user to download the generated image after processing the requirements.

### Non-Functional Requirements

It describes the characteristics of the system and the constraints imposed on the services provided by the system such as time constraints and other constraints. It also describes specific standards for some of the system's services [31].

1. **Performance:** The system must be fast, ensuring that the processing of requirements and drawing (use case or class diagram) does not exceed (write the time). The resulting diagrams should have an accuracy of no less than (write the percentage).
2. **Usability:** The system should be user-friendly and adaptable to various devices to provide a seamless experience. The system should perform its functions with minimal steps.
3. **Availability:** The system should be available 24 hours a day, 7 days a week.

### User Requirements

The services provided by the system are determined to meet users' needs and expectations about the performance of the system. Their documentation is in the form of user stories, use cases, and scenarios [31].

* 1. The system must display a description to write the requirements or scenarios correctly in case of introducing incorrect requirements.
  2. The system must provide practice for the user by setting steps with an explanation of each step to know how to use the tool.
  3. The system must ensure the security of the data used during the generation of diagrams.

### Domain Requirements

Constraints on the system from the domain of operation or environment [31].

1. The system should be able to understand the natural language of users' requirements.
2. The system must accurately understand the relationships between objects and represent them in diagrams.
3. The system should be able to analyses the context of requirements such as relationships and interactions between the actors and use cases.

## Requirements Specification

**Use case diagrams and description:** Use-cases are a scenario-based technique in the UML which identify the actors in an interaction and describe the interaction itself. A set of use cases should describe all possible interactions with the system.

## Developmental (or Research) Methodology

In our project, we used the waterfall methodology for project execution. In the first step, the project plan. In the second step Literature Review. in the third step, analyzed the system. in the fourth step, system design.

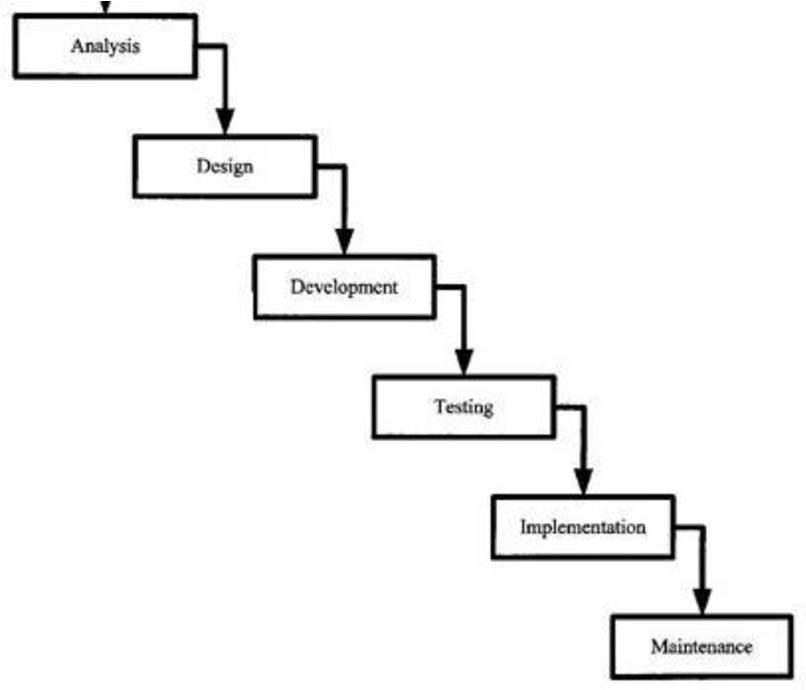


Figure 14: replace pronouns with real-world entities. [32]

* In the first step, we identified the problem and the project's objective.
* In the second step, we explored relevant topics related to our project such as AI, ML, NLP, and another in chapter 2 through books, articles, and electronic libraries.
* In the third step, we analyzed the system and extracted functional and non-functional from similar systems. We also created use case diagrams and descriptions.
* In the fourth step, which involves system design, we implemented architectural design, object-oriented design, and interface design.

## Summary

# Chapter 4: System Design

## Introduction

## Architectural Design

The architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system, i.e. it involves identifying major system components and their communications.

## Object Oriented Design

### Structural Static Models

The diagrammatic representation depends on the modeling language that you use. For example, class diagram, DFD, ERD, etc.

For research-based or simulation project you probably need the (detail) flowchart, experimental design, etc.

### Dynamic Models

The student may present Activity diagrams, State chart diagrams or Sequence diagrams. For example, a sequence diagram may be used to add detail to use-cases by showing the sequence of event processing in the system (shows the sequence of interactions that take place during a particular use case or use case instance). The sequence diagrams are used to model the interactions between the actors and the objects within a system.

## Data Modeling

The databases design or the developed algorithms should be presented here.

## User Interface Design

## Summary

# Chapter 5: System Implementation

The students of CS/IS491 should remove this chapter and prepare it only when they take CS/IS492 in the second semester.

## Introduction

## Tools and Languages

In this section you should state the tools and/or languages that you intend to use and its justifications.

## Mapping Design to Implementation

Detail explanation is required on how the discussion on system design (in Chapter 4) relates with the actual implementation.

## Main/Most Important Codes

The student should include (at least three) selected most important codes with the explanation. Please exclude the code on interfaces, connection to the database, etc.

## System Testing

The system testing can cover one or more of the following test types:

* Unit tests.
* Verify that all system components are integrated properly, and that actual processing situation will be handled correctly (integration test).
* System and release testing.
* Demonstrate that users can interact with the system successfully (usability tests).

## Results and Discussion

The student presents his results and interprets them by making links between the required solution and the results obtained. The student writes arguments and explanations, justifications and logical deductions so that the reader is led to understand his position. All this interpretation approach must be based on facts (existing data and results) and on technical and scientific knowledge of the student.

## Summary

# Chapter 6: Conclusion and Future Work

## Conclusion

In the conclusion the student should make a brief review of the problem, objectives, assumptions and the methodology used, and summarize the solutions and the obtained results.

## Goals Achieved

This section describes to which degree the final outcome supports the original objectives of the project (partially, fully achieved, or exceeded expectations). Here you will summarize the achievements and deficiencies of your project. You may also state what you would/could have done, if you had had more time or if things had worked out differently.

## Limitations and Future Work

In this section the student should set out the limits of his work and suggest new works to be undertaken to complement or improve the present work.

# References

|  |  |
| --- | --- |
| [1] | H. Sofian, N. A. M. Yunus and R. Ahmad, "Systematic Mapping: Artificial Intelligence Techniques in Software Engineering," *IEEE Access,* vol. 10, pp. 51021-51040, 2022. |
| [2] | R. S. e. Bashir, "UML models consistency management: Guidelines for software quality manager.," *International Journal of Information Management,* vol. 36, no. 6, pp. 883-899, 2016. |
| [3] | T. w. l. t. p. o. f. t. a. o. technology, "The world's largest technical professional organization for the advancement of technology," IEEE. |
| [4] | university of BATH, 2017. [Online]. Available: https://www.bath.ac.uk/publications/library-guides-to-citing-referencing/attachments/ieee-style-guide.pdf. |
| [5] | ". R. M. A. D. Library. [Online]. Available: https://dl.acm.org/. |
| [6] | G. F. Luger., Artificial intelligence: structures and strategies for complex problem solving., Addison-Wesley Publishing CompanyUnited States, 2008. |
| [7] | M. O. I. Bashir, "ResearchGate," [Online]. Available: https://www.researchgate.net/figure/Different-branches-of-AI-13\_fig1\_357512563. |
| [8] | 2. W. A. Hany H Ammar1, Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems, 2013. |
| [9] | S. Shafiq, A. Mashkoor, C. Mayr-Dorn and A. Egyed, "A Literature Review of Using Machine Learning in Software Development Life Cycle Stages," *IEEE Access,* vol. 9, pp. 140896-140920, 2021. |
| [10] | H. a. Y. N. A. M. a. A. R. Sofian, "Systematic mapping: Artificial intelligence techniques in software engineering," *IEEE Access,* vol. 10, pp. 51021-51040, 2022. |
| [11] | Z. a. L. F. a. Y. W. a. P. S. a. Z. J. Li, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems,* 2021. |
| [12] | JayeshBapuAhire, "Data Science Central," [Online]. Available: https://www.datasciencecentral.com/the-artificial-neural-networks-handbook-part-1/. |
| [13] | V. Nasteski, "An overview of the supervised machine learning methods," *Horizons. b,* vol. 4, pp. 51-62, 2017. |
| [14] | Y. Vasiliev, Natural language processing with Python and spaCy: A practical introduction., No Starch Press , 2020. |
| [15] | G. L. S. C. &. H. O. S. Moon, "Automated construction specification review with named entity recognition using natural language processing," *Journal of Construction Engineering and Management,* vol. 147, no. 1, p. 04020147, 2021. |
| [16] | S. &. C. C. Jaf, "Deep learning for natural language parsing," *IEEE Access,* vol. 7, pp. 131363-131373, 2019. |
| [17] | R. P. S. C. E. &. T. R. Sukthanker, "Anaphora and coreference resolution: A review," *Information Fusion,,* vol. 59, pp. 139-162, 2020. |
| [18] | P. M. Marta Maślankowska, "neurosys," [Online]. Available: https://neurosys.com/blog/intro-to-coreference-resolution-in-nlp. |
| [19] | C. Manning, "Stanford lecture (CS224n) by Christopher Manning," 2019. [Online]. Available: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1162/handouts/cs224n-lecture10-coreference.pdf. |
| [20] | S. G. L. &. H. C. Loáiciga, "What is it? Disambiguating the different readings of the pronoun 'it.," *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* pp. 1325-1331, 2017. |
| [21] | B. Arendse, "A thorough comparison of NLP tools for requirements quality improvement," 2016. |
| [22] | S. Vajjala, B. Majumder, A. Gupta and H. Surana, "Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP Systems.," O'Reilly Media, 2020. |
| [23] | "lucid collaborative ai," innovation training, [Online]. Available: https://www.innovationtraining.org/lucid-collaborative-ai/. |
| [24] | "lucidchart vs lucidspark," innovation training, [Online]. Available: https://www.innovationtraining.org/lucidchart-vs-lucidspark-when-why-and-how-to-use-them-both/. |
| [25] | "UML/Code Generation Tool," visual paradigm, [Online]. Available: https://www.visual-paradigm.com/features/code-engineering-tools/. |
| [26] | "Defining design pattern," visual paradigm, [Online]. Available: https://www.visual-paradigm.com/support/documents/vpuserguide/26/36/6246\_definingdesi.html. |
| [27] | "starUML in 2022," Pat research, [Online]. Available: https://www.predictiveanalyticstoday.com/staruml/#content-anchor. |
| [28] | "Introduction to Enterprise Architect," sparx systems, [Online]. Available: https://sparxsystems.com/enterprise\_architect\_user\_guide/13.0/. |
| [29] | "MagicDraw," Dassault Systemes, [Online]. Available: https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/. |
| [30] | J. T. Catanio, "Requirements analysis: A review," *Advances in Systems, Computing Sciences and Software Engineering,* pp. 411-418, 2006. |
| [31] | S. Ian, Software Engineering, 11th ed., Addison-Wesley,, 2015. |
| [32] | S. Balaji and M. S. Murugaiyan, "Waterfall vs. V-Model vs. Agile: A Comparative Study on SDLC," *International Journal of Information Technology and Business Management,* vol. 2, no. 1, pp. 26-30. |

# Appendix A

For the graduation project CS492 report, you should prepare an appendix explaining file structure on the CD submitted with it. The appendix must also contain information on how the code should be run (i.e. the user guide or manual). Other appendices may include documents such as: the checklist of examiners’ comments, questionnaire, selected experimental data, schedules, testing strategy or risk management plans. Do not include the source code as an appendix (submit it on a CD). Do not include voluminous appendices (these should also be submitted on a CD, if necessary).

Students who have successfully completed their graduation project, CS492 oral examination and made all the revisions and corrections required by the examining committee, must submit one bound copy to the Project Committee.

# Appendix B

This section shows examples of figure, table and equation. We assume that the figure, table and equation appear in Chapter 2, therefore their numberings will be preceded by number 2. Whenever possible please place the figures and tables alongside with the captions, immediately following the first text that refer to it. If this is not possible, then the figure or table should appear in the following page. More than one figure or table can be placed in a single page. However, it is advisable to disperse the figures and tables throughout the report. Please ensure that the figures and tables do not run across pages.

process3.emf

Figure 2.1: Basic movement process model.

Figure 2.1 shows that before and after spacing should be equal to 12pt. The figure caption is under the figure. Figures are center justification. If you start a paragraph with the word “Figure” then use capital F, otherwise small f.

Tables are similar to the figures, but the difference is that the table caption is above the table, for example, table 2.1 shows an example of way-finding path table concept.

Table 2.1: An example of way-finding path table concept [2].

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Source** | **Destination** | **Cost** | **MidP-1** | **MidP-2** | **MidP-3** | **MidP-4** | **MidP-5** | **MidP-6** |
| Gate1 | Gate2 | 1 | - | - | - | - | - | - |
| Gate1 | Gate6 | 2 | Room1 | - |  | - | - | - |
| Room4 | Room7 | 3 | Room3 | Room5 | - | - | - | - |
| … | … | … | … | … | … | … | … | … |

References are numbered in square brackets, “[” and ”]”. We use IEEE system in our citation above, see the examples in the reference section.

Each equation in a chapter is to be numbered consecutively using a decimal system appearing flush with the right-hand margin. For example:

Y = mx + b (2.1)

The numbers in parentheses are the chapter number and equation number respectively. Every new symbol used in the report text for the first time must be explained. When a large number of special symbols are used, it is permissible to collect them in a table or in a special appendix.

# Appendix C

The following describe the details of the required report format.

**Paper**

Standard A4 size; Weight: 90 Grams

Width: 8.27"; Height: 11.69"

**Fonts, Type Styles**

Font Size = 12 (Normal Text)

Font = Times New Roman

Heading 1 (Font Size) = 22 (Bold), Font = Times New Roman

Heading 2 (Font Size) = 16 (Bold), Font = Times New Roman

Heading 3 (Font Size) = 14 (Bold), Font = Times New Roman

**Margins**

Top = 1.0" Bottom = 1.0"

Left = 1.25" Right = 1.0"

**Spacing**

Line Spacing = 1.5

Paragraph Spacing = 6 pts (before) and 0 pts (after)

**Indentation**

Indent all quotations comprising 4 or more lines by 5 spaces from left.

**Page Numbers**

Except for the title page, number all pages which come before the first page of the body chapters consecutively with lower case roman numerals (i, ii, iii, iv…).

The first page with Arabic numeral (1, 2, 3, and so on) starts from the page of the introduction but it is mentioned on page 2 onwards. Mention page numbers on the bottom right of the page. The first page of each section or chapter will not carry the page number, however the page number will be counted for the proceeding page.

**Headers**

The header will comprise the title of the project report (together with the project logo if you wish). On every odd page will appear the title of the report while on the even pages the title of the chapter or section will be mentioned. The first page of every section or chapter shall not carry the header.