

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](#_Toc147775175)

[Acknowledgement iii](#_Toc147775176)

[List of Figures vi](#_Toc147775177)

[List of Tables vii](#_Toc147775178)

[List of Abbreviations viii](#_Toc147775179)

[1 Chapter 1: Introduction 1](#_Toc147775180)

[1.1 Introduction 1](#_Toc147775181)

[1.2 Problem Definition 1](#_Toc147775182)

[1.3 Project Aim and Objectives 2](#_Toc147775183)

[1.4 Project Methodology 2](#_Toc147775184)

[1.5 Project Timeline 3](#_Toc147775185)

[1.6 Document Organization 5](#_Toc147775186)

[1.7 Summary 5](#_Toc147775187)

[2 Chapter 2: Literature Review 6](#_Toc147775188)

[2.1 Introduction 6](#_Toc147775189)

[2.2 Research Methodology 6](#_Toc147775190)

[2.3 AI 8](#_Toc147775191)

[2.3.1 Text to image Generation 8](#_Toc147775192)

[2.3.2 Conditional image Generation 8](#_Toc147775193)

[2.3.3 Multi modal Generation 8](#_Toc147775194)

[2.4 Machine Learning 8](#_Toc147775195)

[2.5 NLP 8](#_Toc147775196)

[2.5.1 Named Entity Recognition 8](#_Toc147775197)

[2.5.2 Parsing 8](#_Toc147775198)

[2.5.3 Pronoun Resolution 8](#_Toc147775199)

[2.5.4 Syntactic and Sematic Analysis 8](#_Toc147775200)

[2.6 Requirement Analysis 8](#_Toc147775201)

[2.6.1 Semantic Analysis of Requirement Using NLP and ML 8](#_Toc147775202)

[2.7 AI System Development 9](#_Toc147775203)

[2.7.1 AI application Development 9](#_Toc147775204)

[2.7.2 ML and Framework Development 9](#_Toc147775205)

[2.7.3 Intelligent Automation Development 9](#_Toc147775206)

[2.8 Similar Tools 9](#_Toc147775207)

[2.9 Summary 9](#_Toc147775208)

[3 Chapter 3: System Analysis 10](#_Toc147775209)

[3.1 Introduction 10](#_Toc147775210)

[3.2 Analysis of Existing Systems *(Optional)* 10](#_Toc147775211)

[3.3 Requirements Elicitation 10](#_Toc147775212)

[3.3.1 Functional Requirements 10](#_Toc147775213)

[3.3.2 Non-Functional Requirements 10](#_Toc147775214)

[3.3.3 User Requirements or Domain Requirements 11](#_Toc147775215)

[3.4 Requirements Specification 11](#_Toc147775216)

[3.5 Developmental (or Research) Methodology 11](#_Toc147775217)

[3.6 Summary 11](#_Toc147775218)

[4 Chapter 4: System Design 12](#_Toc147775219)

[4.1 Introduction 12](#_Toc147775220)

[4.2 Architectural Design 12](#_Toc147775221)

[4.3 Object Oriented Design 12](#_Toc147775222)

[4.3.1 Structural Static Models 12](#_Toc147775223)

[4.3.2 Dynamic Models 12](#_Toc147775224)

[4.4 Data Modeling 12](#_Toc147775225)

[4.5 User Interface Design 12](#_Toc147775226)

[4.6 Summary 12](#_Toc147775227)

[5 Chapter 5: System Implementation 13](#_Toc147775228)

[5.1 Introduction 13](#_Toc147775229)

[5.2 Tools and Languages 13](#_Toc147775230)

[5.3 Mapping Design to Implementation 13](#_Toc147775231)

[5.4 Main/Most Important Codes 13](#_Toc147775232)

[5.5 System Testing 13](#_Toc147775233)

[5.6 Results and Discussion 13](#_Toc147775234)

[5.7 Summary 14](#_Toc147775235)

[6 Chapter 6: Conclusion and Future Work 15](#_Toc147775236)

[6.1 Conclusion 15](#_Toc147775237)

[6.2 Goals Achieved 15](#_Toc147775238)

[6.3 Limitations and Future Work 15](#_Toc147775239)

[7 References 16](#_Toc147775240)

[8 Appendix A 17](#_Toc147775241)

[9 Appendix B 18](#_Toc147775242)

[1 Appendix C 20](#_Toc147775243)

# List of Figures

[Figure 1: Project Methodology 3](#_Toc148628244)

[Figure 2: Methodology Schema 7](#_Toc148628245)

[Figure 3: RNN model architecture 10](#_Toc148628246)

[Figure 4: Word2Vec architecture: (a) CBOW; and (b) skip-gram 11](#_Toc148628247)

[Figure 5: Examples of experimental results: (a) original text; and (b) NER results. 12](#_Toc148628248)

[Figure 6: Constituency and dependency structures for the sentence ‘the man hit the ball’. 12](#_Toc148628249)

[Figure 7: An example of a non-projective parse tree. 13](#_Toc148628250)

# List of Tables

[Table 1: Project plan v1 4](#_Toc148105216)

# List of Abbreviations

HCI Human Computer Interface

# Chapter 1: Introduction

## Introduction

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. Moreover, machine learning techniques within AI make it possible to simplify software requirements, ultimately enhancing the efficiency and effectiveness of the development process. [1] [2]

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 (Fernández-Sáez, Genero, Caivano, & Chaudron, 2016). 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 (Dzidek, Arisholm, & Briand, 2008). [3]

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** | **2** | 1 | **Weeks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Define project objectives and identify the problem | **Tasks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Review RELATED literature AND theories |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 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.

## 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 review the literature and learn about the techniques used. We review and compare systems similar to ours.
* **Chapter 3.** Our focus in this chapter is 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 design the system, identify the components of the system, model the data, and design the user interfaces.
* **Chapter 5.** After designing the system, our focus in this chapter is on implementing it, identifying tools and programming languages, and testing it.
* **Chapter 6.** A presentation of the project's conclusion and a proposal for new work to improve the current work, and a statement of whether the original objectives of the project have 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 gathering process involves using Google Scholar to find various scientific literature sources such as research papers, conference papers, articles, and e-books that can help us understand similar studies and identify their strengths, weaknesses, similarities, and differences. For citation and referencing, we utilize the IEEE and ACM Digital Libraries.

1. **IEEE:** The Institute of Electrical and Electronics Engineers is the world's largest technical professional association, with active participation in research, paper authoring, conferences, and critical local and global discussions on relevant technological topics. IEEE hosts over 1,600 conferences and events worldwide annually and publishes approximately one-third of all technical publications in electrical engineering, computer science, and related electronics fields. [4]
2. **ACM Library Online:** 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. [5]

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.

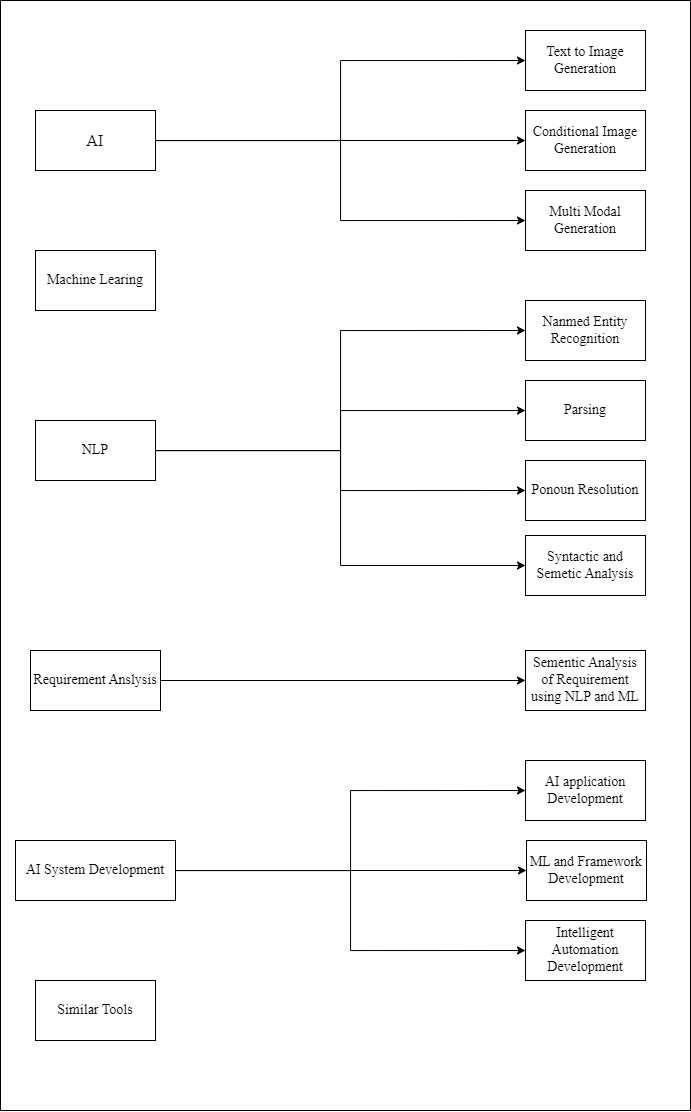


Figure 2: Methodology Schema

## AI

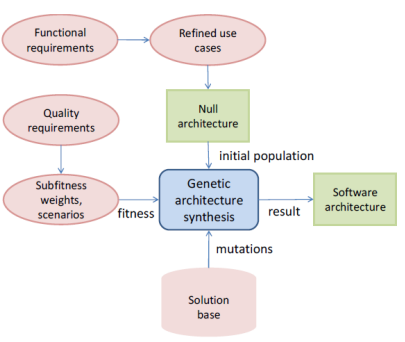
Artificial intelligence (AI) is the branch of computer science concerned with making computers behave like humans (Computers with the ability to mimic or duplicate the functions of the human brain).

### Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems

The software intensive systems we develop these days are becoming much more complex in terms of the number of functional and nonfunctional requirements they need to support. The impact of low quality can also have a catastrophic impact on the mission of these systems in many critical applications. Moreover, the cost of software development dominates the total cost of such systems. Research in applying artificial intelligence techniques to software Engineering have grown tremendously in the last two decades producing a large number of projects and publications. A number of conferences and journals are dedicated to publish the research in this field. The AI techniques are proposed in order to reduce the time to market and enhance the quality of software systems. Yet many of these AI techniques remain largely used by the research community and with little impact on the processes and tools used by the practicing software engineer. The recent survey papers published in this field are mainly targeted to the research community. They are driven by the specific AI techniques used rather than the software engineering activities supported. They are also focused on a specific software engineering process such as software designConditional image Generation

**Software coding and testing**

Techniques learned from AI research make advanced programming much simpler, especially with regard to information flow and control as a result of advances in knowledge representation. In the following we focus on the AI techniques used in supporting the tasks of coding and testing.

****

**Coding**:

Software engineers can apply AI techniques to help automate or assist the programming process.

**Use of AI to help assist the programming process:**

The main idea here is to create an expert system to assist software engineers during software development

**Use of AI to help automate the programming process:**

The idea here is to have a completely automated program synthesis. This is done by having human specialists write a complete and concise specification of the desired software; so that, a system can generate "functions, data structures, or entire programs" directly from the specifications**.**

**Testing:**

Software testing remains an expensive task in the development process and one of the main challenges concerns its possible automation. AI techniques can play a vital role in this regard.

### REQUIREMENTS ENGINEERING AT A GLANCE: COMPARING GORE AND UML METHODS IN THE DESIGN OF AUTOMATED SYSTEMS

The early stage of requirements engineering is crucial for designing intelligent automated systems, especially if no formal analytical solutions have been arrived at. In such cases, requirements analysis is not possible and the design process would be more suitable for rework. Planning and scheduling problems can be addressed using artificial intelligence (AI) and constitute an important area of ​​machine intelligence – along with machine learning.

## Machine Learning

## 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. [6]

**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. [6]

**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. [6]

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.

### 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. [7]

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. [7]

**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. [7]

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. [7]



Figure 3: RNN model architecture

**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." [7]

**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]. [7]

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]. [7]

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. [7]



Figure 4: Word2Vec architecture: (a) CBOW; and (b) skip-gram

**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. [7]



Figure 5: Examples of experimental results: (a) original text; and (b) NER results.

### 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. [8]



Figure 6: Constituency and dependency structures for the sentence ‘the man hit the ball’.

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: [8]



Figure 7: An example of a non-projective parse tree.

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. [8]

### Pronoun Resolution

### Syntactic and Sematic Analysis

## Requirement Analysis

### Semantic Analysis of Requirement Using NLP and ML

## AI System Development

### AI application Development

### ML and Framework Development

### Intelligent Automation Development

## Similar Tools

Highlight the uniqueness of your work and its relation to others.

## Summary

# Chapter 3: System Analysis

## Introduction

It is necessary to study and analyze the current system to understand the shortcomings and the problems, and generate solutions to solve the problems.

## Analysis of Existing Systems *(Optional)*

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

## Requirements Elicitation

The requirements are the descriptions of the system services and constraints.

### Functional Requirements

* + Statements of services the system should provide how the system should react to particular inputs and how the system should behave in particular situations.
  + May state what the system should not do.

### Non-Functional Requirements

* + Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  + Often apply to the system as a whole rather than individual features or services.
  + Use metrics for specifying non-functional requirements (see Figure 1).



Figure 3.1. Metrics for specifying non-functional requirements [1].

### User Requirements or Domain Requirements

## 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

The description of the developmental (or research) methodology is to explain clearly and accurately "How" the student will solve the problem. This section may explain the applied development process model. The description of the methodology comprises the following:

• Description of the working procedure used (list of steps);

• The conditions necessary to achieve each step (e. g., equipment, software, tools, instruments, parameters, etc.);

• Describe the processing of data collected;

• Identify the constraints imposed on the project.

## 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] | 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. |
| [3] | 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. |
| [4] | T. w. l. t. p. o. f. t. a. o. technology, "The world's largest technical professional organization for the advancement of technology," IEEE. |
| [5] | ". R. M. A. D. Library. [Online]. Available: https://dl.acm.org/. |
| [6] | Y. Vasiliev, Natural language processing with Python and spaCy: A practical introduction., No Starch Press , 2020. |
| [7] | S. Ian, Software Engineering, 11th ed., Addison-Wesley,, 2015. |

# 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.