



**DESIGN AND IMPLEMENTATION OF AN INTELLIGENT REQUIREMENTS ENGINEERING
TOOL FOR INTERNET OF THINGS APPLICATIONS IN AN AGILE ENVIRONMENT**

by

BOKANG SEITLHEKO Student no 217167845

Thesis submitted in fulfilment of the requirements for the degree

Master of Engineering: Electrical engineering

in the Faculty of Engineering and build environment

at the Cape Peninsula University of Technology

Supervisor: Dr L. Mwansa

Bellville

December 2021

CPUT copyright information

The dissertation/thesis may not be published either in part (in scholarly, scientific, or technical journals), or (as a monograph), unless permission has been obtained from the University

DECLARATION

I, Bokang Seithleko, declare that the contents of this thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.



Signed

16 February 2022

Date

Abstract

The decomposition of agile epics into user stories manually complicates sprint planning. If epics are poorly understood, they contribute to the threats regarding the sprint's completion. Performing the decomposition manual is laborious and complex and wastes resources in extensive projects. Natural language processing techniques present viable techniques that can automate the reduction of agile epics.

This study explored and attempted to automate the decomposition of epics to their finest granularities, user stories and tasks using natural language processing (NLP). To decompose epics, we extracted and learned the essential parts of the linguistic structure of epics using NLP. The automation of agile epics refinement liberates the product owners from repetitive tasks and focuses more on managerial roles. The results of the decomposed epics were assigned to the task assignment model that uses the Hungarian algorithm to form sprints where team members were allocated tasks to attain a minimum time frame to complete the sprint.

Furthermore, we then present our solution as a smart agile project management tool (SAPMT) that integrates the NLP techniques and Hungarian algorithm to assist project managers in the aspects of epic agile requirements decomposition and tasks assigned. The use of NLP has presented significant results in the generation of user stories and tasks from epics. The algorithm obtained an average accuracy of 89.25%, Precision of 100%, the recall of 77.25%, and the F1 Measure of 87%. The tool SAPMT was implemented using a python framework called Flask and presented a robust graphical user interface.

ACKNOWLEDGEMENTS

I wish to thank:

- God above all.
- Dr L. Mwansa, my supervisor, for marvellous supervision and continuous suggestions through the entire thesis, and for granting me the opportunity to work on this project. Furthermore, he profoundly shaped the way I approach questions.
- My family for giving me the support and words of encouragement to complete this work.

The financial assistance of the Lesotho government sponsor National Manpower Development Secretariat (NMDS) and CPUT bursary towards this research are acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to both sponsors.

DEDICATION

I dedicate this thesis work to my brother who is no longer with us, Sekoati Seithheko who has always believed in my ability to be successful in the academic world. You're gone but your belief in me has aroused my tenacity to complete this work.

Table of Contents

| | |
|--|-----------|
| DECLARATION | II |
| ACKNOWLEDGEMENTS | IV |
| LIST OF FIGURES | X |
| LIST OF TABLES..... | X |
| TERMS AND CONCEPTS..... | XII |
| ABBREVIATIONS | XII |
| 1. CHAPTER 1..... | 13 |
| 1.1 INTRODUCTION | 13 |
| 1.1.1 <i>Overview of Scrum</i> | 14 |
| 1.1.2 <i>Scrum artefacts</i> | 15 |
| 1.1.3 <i>Planning</i> | 16 |
| 1.2 BACKGROUND TO THE PROBLEM | 17 |
| 1.3 PROBLEM STATEMENT..... | 18 |
| 1.4 AIM AND OBJECTIVES | 19 |
| 1.5 OBJECTIVES | 19 |
| 1.6 CONTEXT OF RESEARCH | 19 |
| 1.7 RESEARCH QUESTIONS..... | 19 |
| 1.7.1 <i>Main Research Questions</i> | 19 |
| 1.7.2 <i>Sub Research Questions</i> | 19 |
| 1.8 DELINEATION | 19 |
| 1.9 CONTRIBUTIONS | 20 |
| 1.10 METHODOLOGY | 20 |
| 1.11 THE ORGANISATION OF THE DISSERTATION | 21 |
| 2. CHAPTER TWO..... | 22 |
| 2.1 INTRODUCTION | 22 |
| 2.1 AGILE SOFTWARE DEVELOPMENT | 22 |
| 2.2 EPICS | 23 |
| 2.3 USER STORIES..... | 23 |
| 2.3.1 <i>Quality criterion</i> | 24 |
| 2.4 NATURAL LANGUAGE PROCESSING | 25 |

| | | |
|------------------------------|--|-----------|
| 2.4.1 | <i>Conference resolution</i> | 26 |
| 2.4.2 | <i>Chunking</i> | 26 |
| 2.4.3 | <i>Lemmatization</i> | 27 |
| 2.4.4 | <i>Named Entity Recognition (NER)</i> | 27 |
| 2.4.5 | <i>Part-of-Speech (POS) tagging</i> | 28 |
| 2.4.6 | <i>Sentence segmentation</i> | 29 |
| 2.4.7 | <i>Stemming</i> | 29 |
| 2.4.8 | <i>Syntactic parsing</i> | 29 |
| SCRUM GROOMING PROCESS | | 29 |
| 2.4.9 | <i>Grooming PB</i> | 29 |
| 2.5 | REGULAR EXPRESSIONS | 31 |
| 2.6 | SPLIT THE USER STORY IN A MANAGEABLE TASK USING TRADITIONAL APPROACHES | 32 |
| 2.6.1 | <i>Horizontal and vertical slicing</i> | 32 |
| 2.7 | SPLIT THE STORY USING NLP | 32 |
| 2.8 | STATE OF ART: GENERATING AGILE REQUIREMENTS ARTEFACTS | 34 |
| 2.8.1 | <i>Generating test cases using nlp</i> | 35 |
| 2.8.2 | <i>Applying deep neural network to generate trace links</i> | 36 |
| 2.8.3 | <i>Summary of applying NLP in requirement engineering</i> | 36 |
| 2.9 | TASK ASSIGNMENT | 36 |
| 2.9.1 | <i>Crowdsourcing for task assignment</i> | 37 |
| 2.9.2 | <i>Task assignment based on Hungarian</i> | 39 |
| 2.10 | SUMMARY | 40 |
| 2.11 | CONCLUSION | 43 |
| 3. | CHAPTER THREE | 44 |
| 3.1 | INTRODUCTION | 44 |
| 3.1.1 | <i>Aim</i> | 44 |
| 3.1.2 | <i>Input</i> | 44 |
| 3.1.3 | <i>Output</i> | 44 |
| 3.2 | METHODOLOGY | 44 |
| 3.3 | ENHANCED SCUM PROCESS | 45 |
| 3.4 | USER STORY TEMPLATE | 45 |
| 3.5 | DATASET | 46 |
| 3.6 | ATTRIBUTES OF THE RESEARCH | 46 |
| 3.6.1 | <i>Grooming PB with NLP</i> | 46 |
| 3.6.2 | <i>Hungarian algorithm</i> | 46 |
| 3.7 | SUMMARY | 48 |

| | | |
|-----------|---|-----------|
| 4. | CHAPTER FOUR..... | 49 |
| 4.1 | INTRODUCTION | 49 |
| 4.2 | DESIGN APPROACH | 49 |
| 4.3 | DESIGN GOALS..... | 50 |
| 4.4 | SYSTEM ARCHITECTURE | 50 |
| | 4.4.1 <i>components of architecture</i> | 51 |
| | 4.4.2 <i>NLP engine 3</i> | 51 |
| | 4.4.3 <i>Extracting tasks from user stories</i> | 60 |
| 4.5 | TASKS ASSIGNMENT TO THE DEVELOPERS. | 60 |
| 4.6 | IMPLEMENTATION DETAILS | 62 |
| | 4.6.1 <i>Web application</i> | 63 |
| | 4.6.2 <i>Database</i> | 64 |
| 4.7 | LIBRARIES USED | 64 |
| 4.8 | INPUTS AND OUTPUTS | 65 |
| | 4.8.1 <i>Product Owners inputs</i> | 65 |
| | 4.8.2 <i>Developers' inputs</i> | 65 |
| 4.9 | TASK'S SELECTION MODEL..... | 65 |
| | 4.9.1 <i>Process 1</i> | 65 |
| | 4.9.2 <i>Hungarian algorithm</i> | 66 |
| | 4.9.3 <i>Process 2</i> | 66 |
| 4.10 | OUTPUT | 66 |
| 4.11 | SUMMARY | 66 |
| 5. | CHAPTER FIVE..... | 67 |
| 5.1 | INTRODUCTION | 67 |
| 5.2 | METHODOLOGY | 67 |
| | 5.2.1 <i>Example of evaluation</i> | 67 |
| | <i>Input</i> | 67 |
| 5.3 | EXTRACTED USER STORIES AND TASKS..... | 70 |
| 5.4 | TASK ASSIGNMENT PROCESS | 70 |
| | 5.4.1 <i>Process 1</i> | 70 |
| | 5.4.2 <i>Applying Hungarian algorithm</i> | 71 |
| | 5.4.3 <i>Process 2</i> | 71 |
| 5.5 | SPRINT ITERATIONS | 71 |
| | 5.5.1 <i>Iteration 1</i> | 71 |
| 5.6 | CASE STUDIES | 72 |
| | 5.6.1 <i>Case study 1: ATM</i> | 73 |

| | | |
|-----------|--|-----------|
| 5.6.2 | <i>Case study 2: Ecommerce</i> | 75 |
| 5.6.3 | Case study 3 | 77 |
| 5.7 | OUTPUT OF ITERATIONS AND TASK ASSIGNMENTS..... | 79 |
| 5.8 | RESULTS..... | 80 |
| 5.9 | DISCUSSIONS..... | 84 |
| 5.9.1 | <i>SAMPT performance</i> | 84 |
| 5.9.2 | <i>Lack of dataset</i> | 84 |
| 5.10 | THREADS TO VALIDITY..... | 85 |
| 5.11 | CONCLUSION..... | 86 |
| 6. | CHAPTER SIX | 87 |
| 6.1 | INTRODUCTION..... | 87 |
| 6.2 | SUMMARY OF FINDINGS..... | 87 |
| 6.3 | CONCLUSION..... | 87 |
| 6.4 | FUTURE RESEARCH AND RECOMMENDATIONS..... | 88 |
| | BIBLIOGRAPHY | 89 |
| 7. | APPENDICES | 94 |
| 7.1 | APENDIX A..... | 94 |
| | APENDIXB..... | 98 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1.1:An overview of the Scrum process..... | 15 |
| Figure 2.1:Analysed linguistic structure of an epic requirement | 23 |
| Figure 2.2:NER sentence results..... | 27 |
| Figure 2.3:POS tags for sentence..... | 29 |
| Figure 2.4: tasks involved in PB grooming..... | 30 |
| Figure 2.5: Crowdsourcing process (Shi <i>et al.</i> , 2020) | 39 |
| Figure 3.1: Proposed Scrum process..... | 45 |
| Figure 3.2: Hungarian algorithm flowchart | 47 |
| Figure 4.1:Proposed high-level architecture | 50 |
| Figure 4.2:Outline the design process of decomposing epics | 52 |
| Figure 4.3: programmatic dependency output | 54 |
| Figure 4.4:Dependency graph visualized by displacy | 54 |
| Figure 4.5:Code snippet for partial sentence generation..... | 56 |
| Figure 4.6:Replacing verb with its lemma | 56 |
| Figure 4.7:Activity diagram decomposes user story and task | 57 |
| Figure 4.8: Determine the presence of object in generated phrase..... | 58 |
| Figure 4.9:The output of the dependency graph..... | 59 |
| Figure 4.10:The user interface of the developed tool..... | 63 |
| Figure 4.11:Database schema of developed tool..... | 64 |
| Figure 5.1:Graphical dependency output..... | 69 |

LIST OF TABLES

| | |
|--|----|
| Table 2.1:Conference resolution results..... | 26 |
| Table 2.2:NER keywords meaning..... | 27 |
| Table 2.3:Universal Part-of-Speech | 28 |
| Table 2.4:Regular expression meaning..... | 31 |
| Table 2.5: Frequent linguistic structure from (Müter <i>et al.</i> , 2019) | 33 |
| Table 4.1:Task extracted from ATM text..... | 61 |
| Table 4.2:Available developers | 61 |
| Table 4.3:TO-DO Table (iteration 1) Hungarian..... | 61 |
| Table 4.4:After applying Hungarian algorithm | 62 |
| Table 4.5:Iteration 1 | 62 |
| Table 5.1:Generated user stories and tasks from IBM's payroll system text..... | 70 |
| Table 5.2:Square matrix for task assignment | 71 |

| | |
|---|----|
| Table 5.3:Hungarian results for iteration 1 | 72 |
| Table 5.4:Decomposed epics results from ATM text..... | 74 |
| Table 5.5 Hungarian matrix input 2 | 75 |
| Table 5.6: Iteration-Table (iteration 2) | 75 |
| Table 5.7:Extracted user stories and tasks from Ecommerce text..... | 76 |
| Table 5.8:The Hungarian matrix input 3 | 77 |
| Table 5.9:Iteration-Table (iteration 3) | 77 |
| Table 5.10:generated user stories and tasks from agile samurai textbook | 78 |
| Table 5.11:Hungarian input 4 | 78 |
| Table 5.12:Iteration-Table (iteration 4) | 79 |
| Table 5.13: Iterations-table | 79 |
| Table 5.14:Task assignment | 79 |
| Table 5.15:Classification of sentences that will correctly create user story (Pereira, 2018)..... | 80 |
| Table 5.16:Case study results..... | 83 |
| Table 5.17:Avarage performance of the tool | 83 |
| Table 7.1:Saved stories and tasks | 95 |

Terms and concepts

| | |
|------|---|
| AI | Artificial Intelligence |
| APM | Agile Project management |
| ASD | Agile Software Development |
| CPUT | Cape Peninsula University of Technology |
| IoT | Internet of Things |
| LSTM | Long Short-Term Memory |
| IR | Information Extraction |
| NLP | Natural Language Processing |
| PB | PB |
| PO | Product Owner |
| RNN | Recurrent Neural network |
| SM | Scrum Master |
| USs | User Stories |
| US | User story |

Abbreviations

Greek letters

| Symbol | Meaning |
|----------|---------|
| α | Alpha |
| β | Beta |

1. CHAPTER 1

INTRODUCTION

1.1 Introduction

As technology advanced, there is an exponential increase in the modelling or automation of business processes in a smart way to support complex decision making as to increase production and quality. This disruptive change is brought by the rise of artificial intelligence (AI) and big data technologies in the field of academic research and software industries at large. The introduction of AI in software engineering and project management have been used to support critical decision making in the field of requirements engineering (Wang, 1997; Lin *et al.*, 2015).

Due to the gradual increase of user requirements in the rapidly changing environment, there is a need to adapt to the flexible technology that can support decision making and manage these requirements. The influx of requirements places substantial pressure on the product owner (PO). If not managed well, the backlog can be valueless to the customers. Accordingly, advanced, and scalable solutions such as NLP are required to enhance the functionality of agile project management (APM) tools. These techniques were identified as viable solutions over human intelligence because they are consistent, reliable and efficient and does not exhibit mood swings like human beings (Nayak and Dutta, 2018).

Agile development has revolutionised the software development process and has been prominent ever since its inception among practitioners and researchers. The rapid development of agile has captured researcher's attention in the past decades due to the ability to accommodate the change in requirements along project's schedule without affecting the project schedule and cost (Sharma and Hasteer, 2017). Scrum is an agile methodology which is the most prominent framework due to its ability to divide the project into small manageable modules (Diebold *et al.*, 2015; Sharma and Hasteer, 2017; Khabbazian *et al.*, 2018; Ralph, Sedano and Péraire, 2019). Therefore, this thesis has adopted the Scrum framework to manage the agile projects within the following activities: grooming product backlog (PB) and task assignment.

Managing projects with Scrum methodology is an indispensable part of agile software development (ASD) domain. However, there is still a room for improvement when it comes to requirements elicitation process and task assignment models. Over the last decade, USs are selected as the representation of user requirements in agile software development (ASD.) They are described as the semi-structured and concise representation of user requirements transcribed using natural language (NL) (Kassab, 2015). These stories are transcribed concisely to enable the fast progression of software development while

maximising the business value. Being concise ascribe to the flexibility and adoption of this notation in a dynamic environment such as agile software development where just enough documentation is mandatory.

The widely practiced USs template by practitioners is given as follows: As < actor >, I want to < action > so that < business value or reason > which is easy to comprehend and employ. Albeit easy to use, there are few complications. Some stories are large enough to fit the sprint and this brings about adverse implications on the projects; most stories become partially complete during their development and prolong the project's timeframe. These large stories are referred to as epics. To address this challenge, the preliminary rule of thumb is to decompose or refine these stories using manual efforts. Decomposing is the reduction of an epic or large story into small manageable stories.

The accuracy of the sprint planning lies in the heart of the user story's complexity and inherent risk. Small USs give the development team the confidence to select them over epics during sprint planning because there are no unanticipated emergent details. Additionally, USs bring about adequate architecture, and their efforts are easy to estimates. Furthermore, decomposed stories have a higher probability of being completed on time during their execution than the larger stories. Consequently, they have the potential to provide effective sprints. Therefore, the decomposing of USs play a vital role in sprint planning. However, it is still a challenge to automate the refinement of epics in an agile environment especially in the context of Scrum methodology.

This thesis addresses the problems associated with the decomposition of epics while using manual efforts by introducing an automated NLP solution. The solution fine-tune the large USs to their finest granularity state namely tasks. These tasks are then saved into the databases and act as the input for tasks assignment model called the Hungarian algorithm.

1.1.1 Overview of Scrum

The primary Scrum process is initiated by collecting requirements from various users who are appointed to have meeting sessions with the PO. PO is accountable for creating PB based on the USs' importance. PB is a list of tasks that need to be implemented. Refinery of product backlog items (PBI) is accomplished by the team comprised of Scrum master, PO, and DT. High valued USs are selected from the PB to the sprint backlog where tasks are executed within the period of 1-4 weeks. The end products are tested before reaching the clients to ensure that all requirements have been compensated. **Error! Reference source not found.** depicts the scrum process.

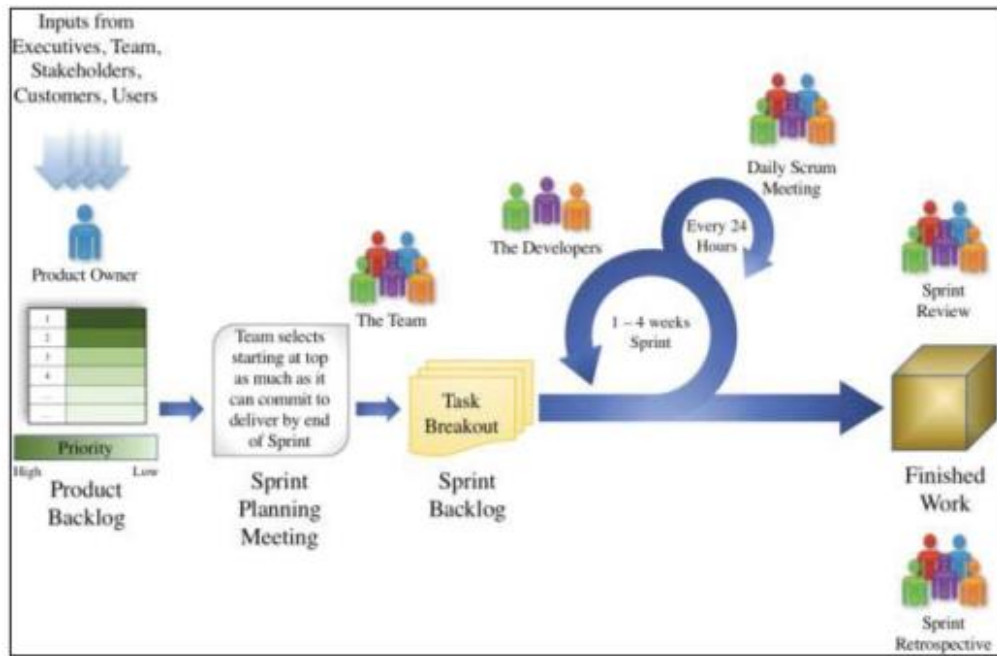


Figure 1.1: An overview of the Scrum process

1.1.2 Scrum artefacts

There are three available scrum artefacts being PB, sprint backlog, and burndown charts. However, this thesis will exclude the burndown charts.

- Product backlog: represents itself as the initial Scrum artefact practised by agile methodologies to capture the elicited requirements epics or USs from users in an orderly and prioritised fashion. A well-recognised representation of USs is transcended to communicate the requirements in agile projects. The typical notation used to present stories entails three attributes: persona, the action, and business value (Vinet and Zhedanov, 2011). These USs can be in the form of functional or non-functional requirements. Stories are transcribed in the form of NL and estimated by using either story points or ideal time.
- Sprint backlog: it's the second artefact of Scrum where the selection of qualifying PBIs from the definition of the ready domain are fit to sprint backlog to formulate the next sprint. These PBI often comes in the form of USs and the entire process is governed by the DT.

1.1.3 Planning

To meet the intense demands and rapid development of projects, the delivery of the product must also be quick. The expedition of development places substantial risk to the project being developed. The development of projects with numerous requirements like the internet of things (IoT) applications demands the agile framework to deliver what the customer or stakeholders need on time and within the budget. Having requirements which are large can hinder the success of the sprint. Therefore, there is a need to identify the most important requirements that align with the next sprint goal via project planning. Planning in Agile is dedicated to managing limited resources. In Scrum, initial planning is occurring at sprint 0.

Sprint planning has a positive influence to foster product quality. The implemented tasks that have value to the next sprints are identified by the PO. PO works hand in hand with the customers. Therefore, the sprint consequently reflects customers' desire because they have a say on what needs to be implemented next. The customer's level of satisfaction and trust to the team is tremendous as he or she fully involved in the sprint planning. Furthermore, sprint backlogs are ordered sequentially concerning the priority values (Liu *et al.*, 2019).

1.2 Background to the problem

Initially, sprint planning is an iterative task carried amongst the agile team SM, DT, and PO. Most work here is completed by SM and other team members have no-input. The meeting would spend hours trying to figure out what to deliver on the next sprint. Here, the PO's responsibility is to specify or assign what needs to be delivered by identifying epic with high importance or business values with high priority. This method is laborious and its labour intensive. Sometimes POs are hesitant to spend hours with the team doing sprint planning (Kniberg, 2015). This causes serious problems because scope and priority are defined by the PO (Kniberg, 2015) inherently causing the sprint to suffer. The new developments of agile planning were born and thanks to the advancements of technology which aid in the acceleration of automation of agile processes and project management tools.

There has been a prior success of sprint planning techniques and tools in the existing literature that automated the processes (García, Cancelas and Soler-Flores, 2014, 2015; Choetkiertikul *et al.*, 2016; Ramirez-noriega *et al.*, 2016; Perkusich *et al.*, 2017; Khabbazian *et al.*, 2018; Ahmed *et al.*, 2019). Since our research focuses on requirements decomposition, the literature on chapter 2 infers that the use of traditional methods like vertical, horizontal splice are still used in Agile methodologies when referring to the US refinement (Taibi *et al.*, 2017).

The development of sprint planning tools that can advocate in decision making on aspects such as automated agile artefacts generation have gained popularity in the field of academics as to assist the agile teams to have the seamless workflow while increasing production. Although there are more sophisticated tools like Azure from Microsoft and Jira, these tools do not offer services such as the decomposition of USs. But they rather support requirements management, task assignments, prioritise tasks etc.

This thesis builds on the dissertation by Pereira (2018) and extended the functionality of their tool by introducing two tasks being (1) The decomposition of USs to tasks and task assignment to developers by applying Hungarian algorithm. The implementation of their design was divided into two segments, first pipeline and second pipeline. This was done to enhance the software performance in terms of speed and reliability. We have also adopted this segmentation approach in our study to enhance the tool's performance. Moreover, we translated Pereira, (2018) dissertation's implementation from java to python language. The rationality to select python was due to its extensive use in AI applications, variety of libraries available online and it's easy to use and comprehend.

1.3 Problem statement

In the centric world of technology where processes are automated, the decomposition of epics into USs is accomplished using human skills. Performing a task manually is laborious and waste resources. For extensive projects where requirements are numerous, this places substantial pressure on the PO, the PO may provide low-quality requirements. The decomposition of epics requires good communication skills and expertise. In start-up companies with a lack of professionals, the complexity of USs may be measured inaccurately.

Due to lack of project requirement resources, inexperience system architects and mainly inadequate practice of applied agile techniques project are still subjected to failure (Taherdoost and Keshavarzsaleh,2015). A large sum of money, estimated to be 322 billion USD, was wasted as a result of bad software engineering techniques (Klotins, Unterkalmsteiner and Gorschek, 2016). According to current software engineering literature, the causes of disappointments are usually project environment, ambiguous requirements, and a lack of a complete set of right agile methodologies (Taherdoost and Keshavarzsaleh, 2015). According to Mohagheghi and Jorgensen (2017), software development with agile methodology and fixed scope had a poor success rate of 58% whereas agile approach with flexible scope had an 87% success rate.

Due to the lack of tools that automates the refinement of epics and task assignment model, sprint planning tools are still prone to ambiguous requirements documentation. Furthermore, the present software products on the market are technologically basic, expensive, and unfit to address 4th Industrial Revolution challenges such as scalability. Furthermore, the repositories of these tools do not capture the insights from preceding similar projects, therefore tasks are repeated. According to MIHALACHE (2017), these tools offer poor PB management.

1.4 Aim and Objectives

Aim: The aim is to design and implement an intelligent tool for IoT application requirements specification into stories for the Scrum team.

1.5 Objectives

1. To groom PB with AI techniques
2. To assign stories / tasks to developers by using optimisation algorithm

1.6 Context of research

This research thesis falls within the discipline of modern software engineering and applications of artificial intelligence in APM with Scrum. Modern software engineering is taking new direction especially in the automation of business processes and model their solutions in the very adaptive manner. For instance, automation of processes such as requirements elicitation with software agents (AI), software testing, etc.

1.7 Research Questions

1.7.1 Main Research Questions.

1. How can NLP be used to decompose agile epic stories into manageable stories and tasks?
2. What is an effective technique used to assign tasks to developers in an agile environment?

1.7.2 Sub Research Questions

1. What are distinguishable linguistic features that an Agile epic must allow decomposed into small manageable stories?
2. To what degree can NLP be used to decompose epics in terms of accuracy and performance?

1.8 Delineation

- As outlined in section 1, the main goal of this research is to design and develop an intelligent tool for IoT application requirements in an agile environment. Therefore, this thesis was populated with suitable AI technologies from NLP and Hungarian algorithms were embedded within the web technologies like Flask framework (python micro-framework used for web development)
- While generating the USs, this thesis excluded the text written in passive mode.
- There are various sources of information that can be utilised for USs, task generation and task assignment. USs can be generated from SRS specification written in NL, However, in this thesis, USs are generated from a text which composed of requirements with large functionality written in English language only. Moreover, the developed tool was implemented only in python programming language.
- Acceptance criteria (AC) for the generated stories will not be part of this thesis.

- The agile framework called is Scrum was used for case studies. Any alternative agile techniques like Extreme Programming (XP), Large Scaled Scrum (LeSS) and traditional methodologies were not part of this research.
- The research was conducted at the premises of Cape Peninsula University of Technology (CPUT) which is in South Africa, Cape Town.

1.9 Contributions

This thesis presents a smart project management tool that advocates in decision support during the initial sprint planning which is sprint 0. It is a web-based tool that was intended to provide two functionalities, (1) groom the PB by reducing the size of complex requirements and (2) assign the decomposed tasks to developers.

- Address the deficiency of Agile requirements decomposition in project management tools
- Proposed a novel based grooming backlog method to lessen the PO's duties
- Contributed to the knowledge in the APM sector as the results produce the publication of a journal paper.

1.10 Methodology

This section emphasises the description of technologies used to accomplish the developed tool and how these techniques were incorporated into the Scrum process to devise the enhanced Scrum methodology. To be more specific, we integrated NLP techniques with the Hungarian algorithm where NLP is accountable for the decomposition of Agile epics while Hungarian addresses the task assignment problem. The output of the NLP pipeline acts as the input to the Hungarian algorithm where the Hungarian's results generate iterations or sprints.

Furthermore, the description of data used to validate the developed tool and the origin data used will be detailed into this chapter. The author generated the dataset comprised of epic requirements based on automated teller machine (ATM) and E-commerce case studies as an additional dataset to the one adapted from (Pereira, 2018).

We ultimately describe how Hungarian can attain minimum task assignment by demonstrating its functionality through mathematical formulas and flowchart.

1.11 The organisation of the dissertation

This thesis is organised as follows: Chapter 2: presents a comprehensive literature review of how to design a suitable APM tool based on the objectives outlined in section 1.5. Furthermore, it also discussed the efficiency of applied techniques in relation to the outlined objectives. Moreover, we conclude this chapter by identifying the existing research gap.

Chapter 3: presents the description of the methodology that compelled this work to design the architecture in chapter 4. Here all interoperability between components in different phases of Scrum is maintained to construct the proposed Scrum process. Furthermore, the decomposition of epics into small manageable stories using NLP was briefly outlined. More information under this section regarding how NLP was used to automate the grooming process will be discussed in more detail in chapter 4. Moreover, identify the structure of the template suitable for the automatic generation of the USs. We end this chapter with a detailed description of the Hungarian algorithm.

Chapter 4: illustrates the implementation of the proposed methodology and conducts experiments to verify and validate the proposed tool.

Chapter 5: analyses and discuss the results obtained in chapter 4 with the relevant literature.

Chapter 6: concludes the thesis. In this chapter, results are compiled, and conclusions are drawn. This led to limitations and recommendations.

2. CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

It is imperative to review the state-of-art for every application development. This chapter contains the review of academic sources used to address the automation of requirements refinement and task assignment. Different scholarly papers were collected, analysed, and extracted insights to address the mentioned problems. We first start by describing ASD, epics, USs, techniques utilised to generate Agile artefacts, and lastly, approaches that accommodates task assignment in an Agile environment are discussed.

2.1 Agile Software development

ASD is an iterative and stochastic development method that anticipates fluctuations in requirements. ASD has revolutionised the software development process and has been prominent ever since its inception among practitioners and researchers. The rapid development of agile has captured researchers' attention in the past decades due to its ability to accommodate a change in requirements along the project lifetime without affecting the project schedule and cost (Sharma and Hasteer, 2017). The inception of Agile in the industry was by popularised Agile manifesto held in the year 2001. Agile manifesto vouched for collaboration between team members as this would build a tacit knowledge among team members, advocate for succinct requirement documentation, room for change over the proposed project schedule and the involvement of customers on the project developed (Beck *et al.*, 2001).

Agile development methodologies attempt to provide numerous opportunities to evaluate the direction of a project through development. The project must be organised into iterations or cycles called Sprints where the DT demonstrates a shippable product increment (PI) to the customer for review before deployment. On each iteration, the DT validates that the product implemented meet customers' expectations before deployment. The DT engage in activities such as analysis, development and testing concerning validation of the product.

Various frameworks that support agility has been in practice ever since the fall of traditional methodology where iterations are done after the completion of the project (Dimitrijević, Jovanovic and Devedžić, 2015). Consequently, agile frameworks such as extreme programming (XP), Kanban, and Scrum have gained popularity. However, Scrum has shown to be the most prominent one amongst the three due to its ability to divide the project into small manageable modules (Diebold *et al.*, 2015; Sharma and Hasteer, 2017; Khabbazian *et al.*, 2018; Ralph, Sedano and Péraire, 2019). Therefore, our case study is going to rely on Scum methodology.

In practice, the Scrum team often selects small stories from the PB to implement. Small USs give the DT the confidence to choose them over epics during sprint planning because there are no unanticipated emergent details, and their efforts are easy to estimate. Therefore, the accuracy of the sprint planning lies in the heart of the US's complexity and inherent risk. Furthermore, decomposed stories have a higher probability of being completed on time during their execution than the larger stories.

2.2 Epics

Massive USs are sometimes referred to as epics (Dimitrijević, Jovanovic and Devedžić, 2015; Ali, Shaikh and Ali, 2016). Usually, this type of stories consists of two or more action verbs based on their analysed linguistic structures. For example, consider the following epic in Figure 2.1:

- The Administrator can reset the user password, update the user's details, and deactivate user accounts that are not functional within 3 months.

```
The --> --> DET --> det
Administrator --> --> NOUN --> nsubj
can --> --> AUX --> aux
reset --> --> VERB --> root
the --> --> DET --> det
user --> --> NOUN --> compound
password --> --> NOUN --> obj
, --> --> PUNCT --> punct
update --> --> VERB --> conj
the --> --> DET --> det
user --> --> NOUN --> nmod:poss
's --> --> PART --> case
details --> --> NOUN --> obj
, --> --> PUNCT --> punct
and --> --> CCONJ --> cc
deactivate --> --> VERB --> conj
user --> --> NOUN --> compound
accounts --> --> NOUN --> obj
that --> --> PRON --> nsubj
are --> --> AUX --> cop
not --> --> PART --> advmod
functional --> --> ADJ --> acl:relcl
within --> --> ADP --> case
3 --> --> NUM --> nummod
months --> --> NOUN --> obl
. --> --> PUNCT --> punct
```

Figure 2.1:Analysed linguistic structure of an epic requirement

From the analysed text, we extracted three action verbs through their POS tag VERB. The list of action verbs is, reset, update, and deactivate.

These stories are massive to handle on a single sprint, so they are moved on to the next sprint to avoid overfitting. Overfitting is when the US is too large and cannot be coupled with other USs very well due to its capacity. It is crucial to note that USs should not be too small or too big in terms of estimates. If the estimations of the user's story points are too small, say 0.3, there is a possibility of facing micromanagement. Moreover, a 60 points story stands a chance or high risk of ending up being a partially complete sprint (Ali, Shaikh and Ali, 2016). There are two methods of splitting USs into a manageable task: horizontal and vertical slice. A more recent study suggested that the utilization of predictive analysis can be deployed on managing USs (Dam *et al.*, 2018).

2.3 User stories

In agile methodology, requirements elicitation are stated as USs. USs are the representation of software requirements specification captured in a concise to avoid ambiguity. They are written in natural language. Over the last decade, USs have been ruled as the representation of user requirements in ASD. They are described as the semi-structured and concise representation of user requirements transcribed using NL (Kassab, 2015). These stories are transcribed concisely to enable the fast progression of software development while maximising business value. Being concise ascribe to the flexibility and adoption of this notation in a dynamic environment such as ASD where less documentation is required but rather focus on the implementation.

Albeit easy to use, there are few complications. Some stories are large enough to fit the sprint and this brings negative repercussions on the projects; partially complete tasks and schedule overrun. These large stories are referred to as epics. Therefore, decomposing them is essential for proper sprint planning. Decomposing is reducing stories into small manageable stories.

The widely embraced USs template by practitioners is given as follows: As < actor/user >, I want to < functionality/goal > so that < business value or reason > which is easy to comprehend and employ (Cohn, 2004).

- As a **user/role** (i.e., Customer, Administrator)
- I want functionality /**Goal** (i.e., Pay using ATM card)
- So that / what **reason** (i.e., I can receive discount next time I buy)

The last part of a US is the reason. This is an optional clause that describes why the implementation of the story is important to the user. As it is optional, not every US has it added nor needs it; however, it is a good practise to add it to avoid vague goal.

2.3.1 Quality criterion

As suggested by Wake (2003) properly written stories in Scrum must adhere to the INVEST criteria. INVEST is an acronym of (Independent, Negotiable, Valuable, Estimatable, Small, Testable).

- Independent: The story should be loosely coupled with one another or independent. Stories that parade a high degree of interdependency convolute estimating, prioritizing, and planning. When applying the independent criterion, the goal is not to eliminate all dependencies, but instead to write stories in a way that minimises dependencies. If the PB is fraught with lot of depended on stories, there are two common practices which can assist to resolve this issue. (1) combine similar story into one and (2) Split story into small manageable sizes (Cohn, 2004).
- Negotiable: A good story is negotiable. It is not an explicit contract in which the development team follow as the future requirements document; rather, placeholders for the conversations between customer and the development team. This means that a good story possesses

an adaptive feature that allow room for improvement; over time additional information can be added such as test ideas etc.

- Valuable: A good story must present a high return on investment (ROI) to the users and customers. They must be implemented in such a way that customers perceive them as important. However, developers may have input regarding the importance of stories, but agreements are reached after discussing them with the customers.
- Estimatable: It is imperative for developers to be able to estimate the amount of works required to complete a story. The quality of estimation varies with team's experience. If the team is well experienced with estimations, the more accurate are the estimates. This is influenced by the gained knowledge from prior stories with similar features. However, there are common barriers which could impede the estimation quality:
 1. Developer lacks domain knowledge.
 2. Developer lacks technical knowledge.
 3. The story is too big.
- Small: Good stories are small. In ASD, it is advisable that the stories are small because they are easily estimable. Having small stories in PB attribute towards the success of sprint as there are no anticipated emergent details which can trigger risks. A small story takes few days or hours to implement. This mean that if there are emergent risk associated with such story during its implementation, the team can resolve it on time.
- Testable: A good US should be written in such a way that its functionality can be validated by passing its test cases. In ASD there are two approaches to test if the developed feature corresponds to what it was intended, (1) behaviour driven development (BDD) and (2) Test driven development (TDD). However, the widely practised method is TDD. In TDD, there is what is called acceptance criteria (AC). AC is the spring of test cases, and they can also be described as the preconditions that must be accomplished to satisfy the PO on the functionalities that the team delivers after the Sprint. Recently, some of the software development companies practice the automation of testing through tools like Selenium etc.

If a customer is not familiar with testing something, this may show that the story isn't detailed enough, or that it doesn't reflect something valuable to them, or that the customer just needs help in testing.

2.4 Natural Language Processing

NL is the language that humans use in daily bases to communicate to one another, it normally routes from different sources such as social media, newspapers etc. These sources are also available in diverse

array of languages besides English. Therefore, having different natural languages translates to different standards concerning the construction of phrases or sentences.

NLP is subset of linguistics, information engineering and AI that concentrates on processing documented text written in variety of natural languages. Problems which are frequently tackled using NLP techniques are, information extraction, text generation, automatic text summarization, and automatic entity recognition. There are variety of tools available on the market that helps to accomplish those tasks. However, the tool that is currently dominating in NLP domain is Stanza. Stanza is the NLP library written in python language and supports variety of languages up to 66 (Qi *et al.*, 2020). In the subsequent subsection, we discuss the most common nlp techniques applied when forming Agile requirements artefacts.

2.4.1 Conference resolution

This NLP process that can also be understood as natural language understanding (NLU) process as it possesses the ability to understand who is been talked about in the given sentence or text processed. It substitutes the pronouns with their relative subjects in each text. The subject mentions are referred to as antecedent (Customer, Administrator) while pronouns are referred to as anophera (his, her, their). Consider the text below as an example to determine the conference resolution (Sukthanker *et al.*, 2020). The conference resolution results of the sentence are shown on Table 2.1.

- *“The bank administrator maintains customers information. But he cannot delete their transaction history.”*

Table 2.1:Conference resolution results

| Anophera | Antecedent |
|----------|------------------------|
| He | The bank administrator |
| Their | customers |

After the processing is done, the modified text is returned as:

- *“The bank administrator maintains customers information. But The bank administrator cannot delete customers transaction history.”*

2.4.2 Chunking

Definition: Separating a sentence into parts that have a discrete grammatical meaning. Examples of these meanings could be the noun phrase (‘the developers’) or verb group (‘have to develop’). Examples: The usage of chunking to identify noun and verb phrases by Mala and Uma (2006); identifying noun phrases and verb phrases using chunking by Arora *et.al.* (2015a); chunking as a pre-processing step for classification by Silberztein *et al.* (2018).

2.4.3 Lemmatization

Definition: The words in the search string are morphologically analysed to derive the basis of the term, known as the lemma. By deleting the inflectional endings of nouns and verbs, the lemma is identified. Using this strategy, we can find words that are similar or linked to the search keyword. For example, the lemma "go" is found in the lexemes "go", "goes", "going", "went", and "gone". The lemma of the search phrase is used to identify and demarcate the activity in the USs.

2.4.4 Named Entity Recognition (NER)

Definition: It the machine learning process of recognising the named entity mentions from a given text or sentence. It can categorise tags as PERSON, MONEY DATE, LOCATION, etc. Consider the sentence in Figure 2.2, the highlighted text is identified as the output of NER elements using stanza library. Table 2.2 below explains the keyword highlighted from the output obtained in Figure 2.2.

Examples: Consider the following code below that extracts NER features from a given text. The NER are highlighted by light blue, orange and green.

```
In [9]: import spacy_stanza
import stanza
from spacy import displacy

nlp = spacy_stanza.load_pipeline("en")
doc = nlp("Amazon has announced its new offices in Cape Town around 2019 September.")

displacy.render(doc, style='ent')
```

```
2021-11-13 16:16:43 INFO: Loading these models for language: en (English):
```

| Processor | Package |
|-----------|-----------|
| tokenize | combined |
| pos | combined |
| lemma | combined |
| depparse | combined |
| sentiment | sstplus |
| ner | ontonotes |

```
2021-11-13 16:16:44 INFO: Use device: cpu
2021-11-13 16:16:44 INFO: Loading: tokenize
2021-11-13 16:16:44 INFO: Loading: pos
2021-11-13 16:16:44 INFO: Loading: lemma
2021-11-13 16:16:44 INFO: Loading: depparse
2021-11-13 16:16:45 INFO: Loading: sentiment
2021-11-13 16:16:45 INFO: Loading: ner
2021-11-13 16:16:46 INFO: Done loading processors!
```

Amazon ORG has announced its new offices in Cape Town GPE around 2019 September DATE .

Figure 2.2:NER sentence results

Table 2.2:NER keywords meaning

| NER keyword | Meaning |
|-------------|---------|
|-------------|---------|

| | |
|------|--------------|
| ORG | Organisation |
| GPE | Geographical |
| DATE | Time |

2.4.5 Part-of-Speech (POS) tagging

Definition: It is the process of tagging words with their representative syntactic part of speech tags within a sentence. It categorises the tags as verbs, adjective, direct object or etc.

Table 2.3: Universal Part-of-Speech

| Tag | Meaning | Example |
|------|-------------|------------------------|
| CONJ | Conjunction | and, while, although |
| NOUN | Noun | London, John, |
| PRON | Pronoun | He, she |
| DET | Determiner | The, a |
| VERB | verb | Cook, eat, walk, sleep |
| ADV | adverb | Now, later, soon, |
| ADJ | adjective | Tall, old, lovely |

There are different libraries used to determine the POS tags for the given text. NLTK, Spacy, Stanza, Stanford CoreNLP, etc. For demonstration purpose we will utilise spacy-stanza. Consider the text below as the input text to be processed:

- *Our Turkey was eaten by the dog. There is considerable range of expertise demonstrated by the spam senders.*

```

Our --> PRON
Turkey --> PROP
was --> AUX
eaten --> VERB
by --> ADP
the --> DET
dog --> NOUN
. --> PUNCT
There --> PRON
is --> VERB
a --> DET
considerable --> ADJ
range --> NOUN
of --> ADP
expertise --> NOUN
demonstrated --> VERB
by --> ADP
the --> DET
spam --> NOUN
senders --> NOUN
. --> PUNCT

```

Figure 2.3:POS tags for sentence.

2.4.6 Sentence segmentation

Definition: In NLP, sentence segmentation is the process of splitting a sentence from a given text with multiple sentences.

2.4.7 Stemming

Definition: Reducing a word to a base form: the stem. However, as opposed to lemmatization, stemming operates without context; so, if, for example, ‘better’ would be both stemmed and lemmatized, stemming would yield no output while lemmatization would return ‘good’. A categorization of NLP within requirements elicitation and analysis.

Examples: Stemming as pre-processing for analysing use cases by (Bolloju, Schneider and Sugumaran, 2012); stemming words to create keywords of a requirement by (Ninaus *et al.*, 2014); stemming as pre-processing for predicate generation by (Veerappa and Harrison, 2013).

2.4.8 Syntactic parsing

Definition: Recognizing a sentence or text and assigning a syntactic structure to it. This task includes both dependency-based and constituency-based parse trees.

Examples: Constituent parse tree as pre-processing for the identification of candidate services in Bhat, Ye and Jacobsen (2014) creating a dependency-based parse tree to uncover dependencies between words in sentences by Biébow and Szulman (1993) using syntactic parsing as a way to evaluate well-formedness of a US by (Lucassen *et al.*, 2015).

Scrum grooming process

2.4.9 Grooming PB

The definition of PB is explained in chapter 1 section 1.1.3. It is a list of requirements from customers that needs to be implemented and present the working product. The contents of PB are called PBI and they are listed as epics, defects, updates, and features.

Refinement of PBI from PB is usually referred to as PB grooming. It is a technique utilized by PO in collaboration with development team to keep the PB clean and organized. However, involving the team leaders is optional. The involvement of the team leaders during grooming communicates a clear vision about the upcoming sprint to the team. Consequently, the development teams can anticipate the next sprint intensions (Cobb, 2015).

The PO and team members gather to distill the size, risk and assign priority values to USs based on their gained knowledge on previous backlogs. This is performed in sprint planning meeting where the next sprint is prepared.

New developments may emerge from this meeting due to customers' demands influenced by the demand on the market: adding new stories into PB, reassigning story points, and reducing the epics. This process affects the structure of PB. For instance, the decomposition of epics into small manageable stories cause the addition of new stories to the PB. New items must adhere to the principles or guidelines that govern USs properties, INVEST model. INVEST is an acronym that states how the good USs should be structured: Independent, Negotiable, Valuable, Estimable, Small, and Testable (Cohn, 2004). Moreover, there is a DEEP acronym used as a quality metric to identify good characteristics of PB. As outlined by Meyer (2014), DEEP stands for Detailed appropriately, Estimated, Emergent, Prioritized.

The most critical goal of grooming is to maintain the PB in good order such that it is prepared for the upcoming sprint. It is normally refined during sprint planning; the development teams and PO spans about 5 - 10 % of sprint doing grooming. Figure 2.4 shows activities involved in PB grooming; prioritisation, estimation using story point ,adding new PBI, deleting less important PBI, and refinement of large PBIs.

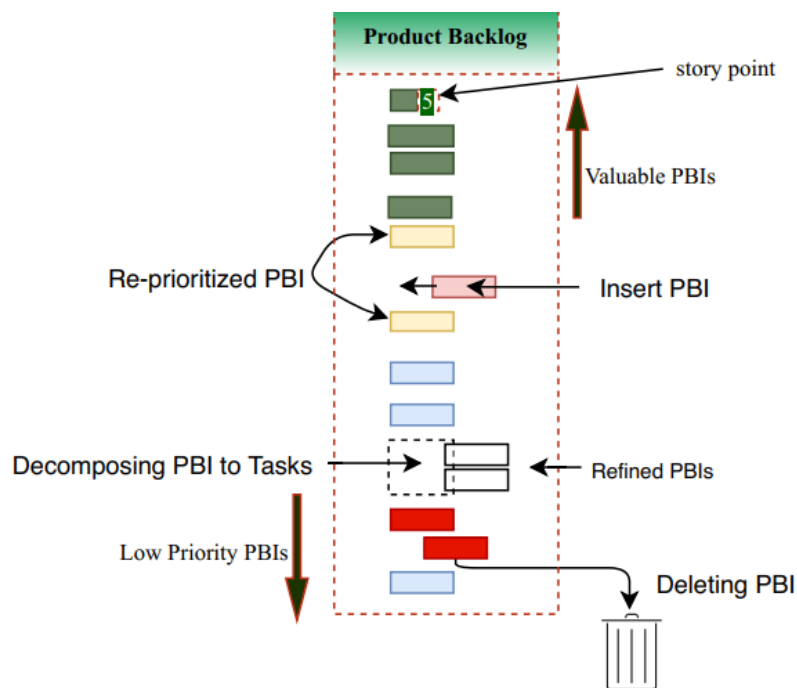


Figure 2.4: tasks involved in PB grooming

Reassigning story points, adding new stories, and the decomposition of epics contributes to the continual grooming of PBI in the PB. For instance, a low prioritized story can be moved into the high-value USs in the PB due to the dependency with higher US. In most cases, customers only concentrate on products that bring business value. Therefore, the current research studies suggest the decomposition of epics into USs to fit the sprint. It is worth noting that adding new features causes PB scope creep due to high incoming requirements.

We have discussed what grooming entails and how it is established. There are various activities involved in grooming: reprioritization, splitting USs, effort estimation, inserting new PBIs, etc. However, this thesis only concentrated on splitting epics into small manageable USs due to the scarcity of literature available in the academic world. We have also extended further to research about how USs are decomposed into tasks. Therefore, it was important to reasech about the state of art on this field.

2.4.9.1 Importance of grooming

The review and refinery of business prioritization and resize of the PB are done in parallel with the implementation of the sprint. If the product grooming is not done, the next sprint will lack an appropriate PB ready for the PO and the team to agree on what the next sprint need to include. Instead, they may have to spend a day or two doing an evaluation which could delay the start of the upcoming sprint.

2.5 Regular expressions

Regular expressions (regexps) are essential tools in computer science regarding information extraction, text analysis and etc. In information extraction, a popoular example could be the extraction of a pattern from the String. For instance, it detemines the authentic format of emails issued by the users. The table below illustrates the information about RE and their meanings. RE are highly used with NLP technique like POS Tagging.

Table 2.4:Regular expression meaning

| Character | Regular-expression meaning |
|-----------|--|
| . | Any character, including whitespace or numeric |
| ? | Zero or one of the preceding character |
| * | Zero or more of the preceding character |

| | |
|---|--|
| + | One or more of the preceding character |
| ^ | Negation or complement` |

2.6 Split the user story in a manageable task using traditional approaches

2.6.1 Horizontal and vertical slicing

According to Taibi *et al.* (2017), the widely practiced decomposing technique within the boundaries of Agile is US mapping. It describes the decomposition of large USs from the user's perspective; It provides the highest level of requirements abstraction. In story mapping, large stories are coarse-grained from epics to stories until their constituent's tasks. For instance, "create registration form" and "create login page for the system" are good examples of high-level requirements. In addition, they further explored how different agile methodologies such (XP, Scrum, Scrum with Kanban) engage in the decomposition process. Their research results revealed that the utilization of traditional processes is still inherited in the agile process during the splitting of stories. Moreover, the most proficient method between the discussed methods is Scrum with kanban followed by XP. The success of Scum with Kanban was due to the use of vertical slicing technique. Albeit its popularity among Agilist, story mapping is achieved by human expertise.

Vertical slicing is the technique of decomposing of an epic by touching aspects of every layer such as from User Interface (UI) to database. It encourages to showcase the delivery of product increments frequently to the end-users such that they provide feedback and incorporates updates within the subsequent iteration. In study contacted by Ratner and Harvey (2011), four teams were deployed to study and determine how efficient is horizontal and vertical slicing in US decomposition. Vertical tends to have positive traits than horizontal slicing in terms of risk and completions of project. The utilization of horizontal technique parades no functionality to the end-users rather partially completed tasks which leads to reiterate and delivers ineffective sprints.

There are other studies that worth noting, Lawrence proposed the strategy that decompose epics through the reprioritization and isolation of requirements. Here large story is chunked into small fragments and discard stories with lower importance or that has no importance. This technique was found to improve the isolation and decreases inter-dependencies between stories.

2.7 Split the story using NLP

The degree of automating requirements' decomposition is scarce in agile software engineering in both academic world and enterprise environment. However, the contemporary state of art only provides the roadmap of possibilities of using NLP to address the challenges faced by manual techniques. Although NLP is deemed to be effective approach to transform and model requirements in different granularities,

it is still fraught with complications. Requirements are sometimes ambiguous and inconsistent. Thus, there is a room for improvement in requirement elicitation in the form of NL.

The use of NLP techniques has modernized the refinery process of epics in ASD. NLP techniques are utilised to automate and augments human capabilities especially in the field of text analytics, language modelling and language translations. Consequently, having requirements predominately written in NL symbolize NLP as an effective candidate for US decomposition.

The preliminary attempt which paved the way on the decomposition of requirements using NLP was recently shed to light by (Müter *et al.*, 2019). They studied linguistic structure that characterize USs together with their corresponding sprint backlog items. To achieve this, they utilised the Stanford Part-of-Speech (POS) tagger to determine the structure of the task labels. POS tagger is especially used in NLP to extract language structure such as verbs, adjectives, nouns etc. Their results revealed some useful insights that can be employed to form linguistic structure of tasks. Table 2.5 shows their analysed results.

Table 2.5: Frequent linguistic structure from (Müter *et al.*, 2019)

| Structure | Frequency | % | Example |
|------------------------------|-----------|------|---|
| VB, NN(S), NN | 130 | 8.17 | create tender-settings component |
| VB, NN(S), NN, NN(S) | 67 | 4.18 | Create Message DB tables |
| NN, NN(S), IN, NN | 25 | 1.57 | Admin licences breadcrumbs |
| VB, NN(S), IN, NN | 21 | 1.32 | Add filters for KO |
| VB, NN, NN(S), NN(S), NN | 20 | 1.26 | Implement TenderPlan actions business logic |
| VB, JJ, NN(S), NN | 18 | 1.13 | Create disqualified offers card |
| VB, NN | 18 | 1.67 | Create |
| | 27 | | TenderProcessDefinitionLevelRule |
| VB, NN(S), IN, NN, NN | 15 | 0.94 | Bind rules per section item |
| VB, NN, NN, IN, NN, NN(S) | 13 | 0.82 | Create SQL script for AcceptedById items |

| | | | |
|-----------|----|------|----------------|
| NN, NN(S) | 10 | 0.62 | Update actions |
|-----------|----|------|----------------|

They went further and define task as the given equation 2:1.

$$\text{task} = \text{verb, follow, \{follow\}}. \quad 2:1$$

Where:

follow = noun | conjunction | adjective | "to" | cardinal number.

(Gunes and Aydemir, 2020) developed a goal-oriented NLP powered tool that generates and modify USs. To accomplish their objective, they presented the fundamental heuristics that combines the pieces of information distilled from USs by applying NLP techniques and stored in a Neo4j graph to view the relations between analyzed artefacts.

2.8 State of art: generating Agile requirements artefacts

There is a plethora of research papers that presented the automatic generation of Agile artefacts using NLP techniques. Applying NLP is not the new topic in the generation of Agile software artefacts. The structure of agile requirements makes NLP to be viable solution to generate them. This was due to the new NLP libraries which provides high accuracy as compared to the state of the art. Requirements are predominately written concisely using NLP to focus more on production. This section highlights the success of NLP in generation agile artefacts namely, USs and test cases. The rationality to select these two artefacts was that they are mostly used in every agile project developed, especially in Scrum context.

(Azzazi, 2017) proposed a framework that transforms USs into use cases by exerting NLP techniques. To reach their goal, they studied the heuristics that can generate the use case from the US’s text by studying the linguistic features of US and how it relates to the linguistic structure of the use case. They divided the US’s information into two categories: (1) extraction of Noun from US of which the Noun is appended into a list referred to as “Actor List” and (2) extraction of the verb from the US and stored the acquired information into a list called “Use case “. This was feasible solution due to the relationship that the US shares with use case is <<extends>>, so this means the existence of use case is strongly dependent on the presence of US.

(Robeer *et al.*, 2016) amalgamated NLP heuristics to form an algorithm that automatically generates conceptual models from the USs. Their paper reported that they have acquired high precision and recall

greater than 80% on both metrics. A more similar paper to Robeer *et al.* (2016) was recently proposed by (Tugce and Aydemir, 2020). (Tugce and Aydemir, 2020) have presented goal-oriented models using the NLP pipeline which aided in automating the creation and visualisation of the generated models from USs. The results obtained from their paper were proclaimed to be so accurate that they were compared to models generated by human experts.

2.8.1 Generating test cases using nlp

Software testing is an integral part of the software process which verifies that the feature developed conforms to the standards it was meant to serve by detecting errors and defects at an early stage to avoid software failure like in (Garfinkel, 2005). By observing its positive traits, it can also be used as a metric to measure the quality assurance (QA) of the developed software. Testing a developed feature triggers the generation of test cases. According to (Ansari, 2017), test case generation comes in three phases: coding, design and specification. The specification phase deals with deriving test cases from functional requirements. Notwithstanding its importance, it consumes a lot of time to generate test cases especially in the context of extensive complex projects. Therefore, researchers had to come up with new approaches to automate the generation of test cases of which most of them harness the power of NLP techniques.

(Verma, 2013) proposed the construction of test cases from software requirements specification (SRS) documentation expressed as NL with the help of NLP techniques. To accomplish their goal, they apply POS tags and syntactic relation parser which returns the relation between words in a graphical manner. The graph provision for better text analysis that leads to test case generation. In 2017, Ansari (2017) proposed a similar approach to the Verma (2013) by excavating test cases from SRS documentation using NLP. The noticeable difference was that the algorithm proposed by Ansari (2017) was built on keywords context from the functional requirement of the SRS documentation while Verma didn't specify.

A more recent study by Wang *et al.* (2020) presented the automated user acceptance test cases generation from use case specification by utilising NLP approach. Their algorithm proclaimed to provide unlimited abstraction when it comes to writing the use cases unlike other approaches. The systematic literature conducted by Raharjana, Siahaan and Fatichah (2021) provides a comprehensive work carried out in research concerning the applicability of NLP in USs. The study reported on creation of traceability matrix using NLP, generation of test cases from SRS, generating use cases from USs etc.

There is also other noticeable work which contributed towards the generation of test cases besides using the NLP techniques. The test case generation from unified modelling language diagrams using a genetic algorithm. The algorithm is proclaimed to provide early defects detection and lessens the time consumed

while extracting test cases as compared to manually efforts. NLP was used to transform USs into use cases.

There also classification machine learning approach that helped to identify if the story needs to be decomposed or not. The dataset used was numerical values with two classes, 1 and 0. The 1 indicates the need to decompose a story while 0 indicates no need to decompose the story.

2.8.2 Applying deep neural network to generate trace links

The use of deep neural networks (DNN) is slowly overcoming NLP in NL problems. This is due to the DNN being able to process data in a short time and return highly accurate results. Some of the flavours of neural networks possess memory to remember important features which can be exerted to excavate useful information such as understanding sentence semantics etc. The variety of DNN which can offer those features are the convolutional recurrent neural networks (RNN) models namely; long short-term memory (LSTM), bidirectional LSTM (Bi-LSTM), gated recurrent unit (GRU) and Bi-GRU. (Guo, Cheng and Cleland-Huang, 2017) presented a semantically enhanced DNN architecture (Bi-GRU) to determine the trace links between software artefacts. The results reported on their papers has surpassed the state of arts. Determining trace links in agile is imperative for provision of defects and requirements management.

2.8.3 Summary of applying NLP in requirement engineering

Using NLP is deemed to stand as the pillar of automatically generating agile artefacts, especially in requirements elicitation, requirements testing, and requirements management. There is profound evidence that can attest to this statement based on the discussed theories applied in preceding sections. From the literature, one can infer that NLP provides substantial techniques that can obviate manual work carried by business analysts concerning the capturing of requirements from customers, generating USs and test cases.

The use of DNN seems to provide accurate results which surpass the state of art in relation to requirements traceability. However, the major challenge with DNN is that it requires massive amount of data to train and its memory hungry. Triggered by this observation, this thesis has adopted the use of NLP techniques to generate USs and extended the functionality of the tool proposed by (Pereira, 2018). In contrast with Pereira (2018) our tool introduces two features namely (1) decomposition of USs to tasks and (2) assign the decomposed tasks to the developers.

2.9 Task assignment

Task assignment is a paradigm of assigning tasks to workers to accomplish a certain goal. The assignment process is mostly governed by criterion such as delivery time and resources spend. The task

assignment problem is regarded as an NP-complete problem (Salman, Ahmad and Al-Madani, 2002). Recently, the problems associated with task assignment are predominately resolved using Crowdsourcing and Hungarian algorithm especially in software development.

2.9.1 Crowdsourcing for task assignment

Crowdsourcing is the task assignment paradigm where tasks are allocated to dispersed workers across the globe. The paradigm aims to improve collaboration between developers globally rather than small group isolated developers (Begel, Bosch and Storey, 2013). This technique is currently studied in recent literature under the topics "spatial crowdsourcing and online task assignment". Crowdsourcing was adopted in software industries due to the potential to deliver rapid products at a low cost.

In crowdsourcing, developers are assigned microtasks to complete and they receive incentives as rewards. Remote workers are under the spectrum defined by the requester and participants, or developers subscribe to the to-do list by the requester (i.e., the requester must be bonded by location with the worker). Sometimes tasks are allocated to workers by physically collecting them from the requester and receiving money once completed.

However, some of the proposed crowdsourcing methods possess characteristics that can affect product quality adversely. There is no assurance that the participant developer subscribed to solve the task can implement the expected content as their expertise levels are sometimes unknown (Ho and Vaughan, 2011). Furthermore, since developers come from different network communities; therefore, this diversity sometimes translates to workers not working together.

2.9.1.1 *Applying crowdsourcing in unknown competence levels of the developers*

Since traditional crowdsourcing was fraught with various challenges, including the unknown skill set, some of the researchers took it up for a challenge to learn the skillset that the developers possess by monitoring the developers' performance and time developers take to complete the task. (Ho and Vaughan, 2011) explored the problem of assigning miscellaneous tasks to workers with various, unknown skill sets in crowdsourcing. Their proposed algorithm included two constraints: a fixed set of tasks and budgeted for each task with several iterations the worker must complete the task. They implemented a solution on the premise that their objective was to allocate tasks to workers such that they achieve maximum profit. To achieve this, the unknown worker's competence level had to be known. This was discovered later by monitoring the performance of each worker as they accomplish the task to estimate their skills. Although their aim proclaimed to maximise profit, assigning tasks to participants with unknown competence levels poses a threat to the project's schedule and quality assurance.

An adaptive crowdsourcing framework called SMARTCROWD is presented (Basu *et al.*, 2015). This framework can address task optimisation through knowledge-intensive crowdsourcing (KI-C) which accounts for human aspects such as skillset, proposed income per requirements to be developed and the presence of workers inside the optimisation process. KI-C is a recent form of the crowd, which concentrates on knowledge development rather than traditional crowdsourcing whose optimisation concentrates on quality and cost. Basu *et al.*, tooling presented an adaptive feature that can be efficient for APM. For instance, delegating tasks to the senior developer in the case the current developer who was involved in the implementation of certain tasks is absent. Observing the positive traits that this framework exerts, can diminish the development time and budget due to multiple developers who undertake a single task.

A real-time spatial crowdsourcing task assignment where only developers from the same neighbourhood as the requester are eligible to participate in the task assignment was proposed (Tran *et al.*, 2018). Algorithms under this domain often concentrate on maximising the number of tasks assigned to developers under a limited budget across the entire campaign. Being influenced by the complexity of classical crowdsourcing paradigms, they further proposed an online heuristic that exploits the spatial and temporal knowledge learnt over time.

2.9.1.2 *Online crowdsourcing*

The recent state of the art shows that there is a trend of online crown sourcing when it comes to (Miao *et al.*, 2020) designed a probabilistic online tasks assignment suitable for mobile crowdsourcing. The technique is distinct from traditional crowdsourcing in the following perspective, tasks and workers appear in the platform dynamically and workers are only restrained to perfume spatial tasks with the limited number of tasks they can execute.

Because crowdsourcing some of the workers are unreliable due to the small amount of money they receive, crowdsourcing decided to devise a strategy to ensure the reliability of workers. The common practice amongst crowdsourcers was to assign a single task to different workers and compare the best solution provided by combining the answers in some way as majority voting. To diminish the cost of assigning a single task to different workers, Karger (2011) proposed an iterative learning algorithm that possesses the ability to distinguish appropriate workers to execute a certain task and to infer correct answers from workers' answers. Figure 2.5 shows the crowdsourcing process followed for task assignment. For the process to be successful, it starts from step 1 and end at step 6.

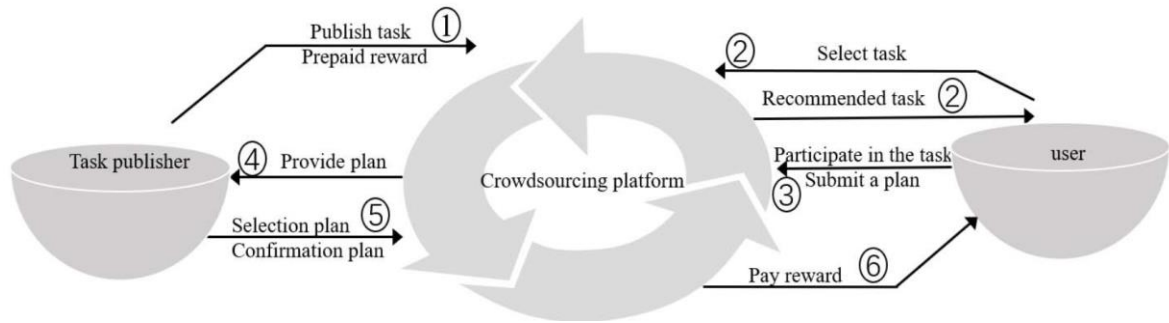


Figure 2.5: Crowdsourcing process (Shi *et al.*, 2020)

Since crowdsourcing is fraught with several complications concerning quality assurance and time spent by novice workers to deliver tasks, (Shi *et al.*, 2020) saw an opportunity to impede this behaviour by proposing a new task recommendation model based on the Hungarian algorithm. This model can enhance the efficiency of crowdsourcing by lessening the time spent on tasks by novice developers as they will be assigned tasks that are equivalent to their expertise.

2.9.2 Task assignment based on Hungarian

A novel based task assignment algorithm that harness the power of the Hungarian algorithm was proposed (Yu, 2019). Hungarian was utilised to resolve optimum matching when the bipartite graph structure is determined. The structure was dynamically attained by altering the bipartite graph structure through the collaborative candidate group replacement strategy. Other researchers addressed the Travelling Salesman Problem (TSP) with Hungarian due to reaching optimum solution (Mondal, Hossain and Saha, 2013).

To address the problem of allocating competent developers to appropriate tasks as an unbalanced personnel assignment problem, Wang *et al.* (2017) proposed an algorithm which improved traditional Hungarian algorithm by applying three strategies; (a) assign tasks to developers with an exceptional skill set to accomplish a task, (b) create a cluster developer based on their optimal ranking, and (c) group developers based on for the optimal group assignment. An improved version of the Hungarian algorithm was proposed by (Mills-tetty and Stentz, 2007).

A reinforced Hungarian algorithm (RHA) for task assignment in global software development was proposed (Wu *et al.*, 2017). RHA consist of three key phases. First, RHA changes $n \times m$ cost matrix by adding $(2n - m)$ virtual development tasks. Second, RHA executes the traditional HA on the two $n \times m$ matrix to get optimal assignment results. Finally, RHA removes the virtual development sites to get the optimal assignment results.

2.10 Summary

This chapter introduced ASD and its terminologies. Agile methodologies have captured researchers' attention in the past decades due to its ability to accommodate change in requirements along the project lifetime without affecting the project schedule and cost. We described and discussed requirements artefacts which are involved in Agile particularly user stories and epics. US are the representation of software requirements specification captured in a concise to avoid ambiguity. The most complex USs to work with are called epics due to their size. The rules that govern the properly written US were discussed. Later showed necessary NLP techniques which are used to extract essential information to generate agile artefacts. The literature showed that POS tagger is the most used NLP data extraction technique. POS tagger is especially used in NLP to extract language structure such as verbs, adjectives, nouns etc. In ASD, POS tagger extract information such as action verbs which are used to form part of US, use cases, test cases etc.

Different task assignment models were discussed applicable in agile environment were discussed. The literature revealed that problems associated with task assignment are predominately resolved using Crowdsourcing and Hungarian algorithm especially in software development. We concluded the chapter by analysing the importance of applying AI in project management.

2.11 Summary of literature

| Author/Date | Aim/goal | Concept Theoretical model | Paradigm / Method | Context / Setting / Sample | Findings | Future Research |
|----------------------------|---|----------------------------------|------------------------------|---------------------------------------|---|---|
| (Raharjana et al., 2021) | to capture the current state-of-the-art of NLP research on user stories. | forward and backward snowballing | systematic literature review | Nlp in ASD | <ul style="list-style-type: none"> The generation of user stories from free text has not yet been much explored Contextual knowledge is needed when processing user stories Generating models/artifacts from a user story is widely performed by researchers. | We hope that the ASD would also thrive in NLP and user story research. Research in broader aspects, such as management and requirement |
| Taibi <i>et al.</i> (2017) | To investigate the process through which user stories are refined into tasks | | NLP | Study the backlog item of 1,593 items | <ul style="list-style-type: none"> widely practiced decomposing technique within the boundaries of Agile is user story mapping | <ul style="list-style-type: none"> The guidelines are likely to need some amplification. their impact on software development needs to be evaluated in vivo |
| (Ratner & Harvey, 2011) | To determine how efficient is horizontal and vertical slicing in user story decomposition | | Survey | four teams | <ul style="list-style-type: none"> Vertical tends to have positive traits than horizontal slicing in terms of risk and completions of project. horizontal technique parades no functionality to the end-users rather partially completed tasks which leads to reiterate and delivers ineffective sprints. | Automate the process |
| (Pereira, 2018) | To generate user stories from SRS | | NLP | ASD | <ul style="list-style-type: none"> concise texts with the description of the software focused on the user perspective have the greatest result | <ul style="list-style-type: none"> enhancing the sentence splitting and the separation of a verb plus complement pairs saving the information in a database before returning the list of user stories |

| | | | | | | |
|---------------------|--|------------|---|-----------------------|---|---|
| (Ansari, 2017) | generate test cases from the functional requirement given in conjunctive statement format | | NLP | | Due To maximum efforts and time consumed in developing test cases is being saved | <ul style="list-style-type: none"> • automatically construct test cases from the functional requirement given in any form • predict that what test cases would prove to be more important in the future |
| (Basu et al., 2015) | To address task optimisation through knowledge-intensive crowdsourcing | SMARTCROWD | | Crowd sourcing | <ul style="list-style-type: none"> •to integrating the human factor further into the task assignment process •To determine the peripheral methods on which KI-C optimisation is based | <ul style="list-style-type: none"> • imperial study to validate the |
| (Miao et al., 2020) | proposed a real-time spatial crowdsourcing task assignment where only developers from the same neighbourhood as the requester are eligible to participate in the task assignment | | Spatial crowd sourcing | Online crowd sourcing | <ul style="list-style-type: none"> • workers are only restrained to perform spatial tasks with the limited number of tasks they can execute | <ul style="list-style-type: none"> • Use approximation algorithm to lessen time complexity • change the worker assignment scheme in Algorithm 1 such that tasks with higher priorities are assigned to workers with higher probabilities. |
| (Son et al., 2021) | propose an approach based on multi-objective combinatorial optimization to do this automatically | | Online crowd sourcing and Hungarian algorithm | | <ul style="list-style-type: none"> • improve the efficiency of crowdsourcing by lessening the time spent on tasks by novice developers | <ul style="list-style-type: none"> • to refine the model to consider new goals and constraints in many situations. |

2.12 Conclusion

Agile methodologies are the future of software development with the integration of AI. Adopting AI brings lucrative results to the executives as it promotes quality development of products. On the other hand, Agile delivers value to customers in a short iterative manner and does not compromise project schedule and costs. The integration of AI and Agile is in demand to develop the tools that are truly intelligent to support project managers in the project planning activities.

Improving project management activity such as estimates facilitate more effective control of time and budget.

Based on the attained information from the literature, it is evident that to develop a scalable ubiquitous APM tool that supports traceability between requirements and PB management, AI is the right candidate. The gradual increase in requirements does not place a substantial thread to the system performance but increases its estimation accuracy while using AI. It is also worth noting that, the quantity and quality of data contributes towards the higher precision of trained AI solutions, especially in the field of deep learning.

Most of APM tools focus on PB management and sprint planning. However, according to the author's knowledge, there is the scarcity of tools that incorporate agent-based technology in managing the PB with the inherent USs risks. Therefore, this thesis aimed to bridge the gap in the literature and implement the smart tool that will support the outlined objectives in 1.5 using AI. Less effort is devoted to decomposition of USs. The acquired literature leads to the development of a methodology that is described in chapter three.

3. CHAPTER THREE.

RESEARCH METHODOLOGY

3.1 Introduction

Based on the analysed advantages and disadvantages of techniques found in the literature APM tools performance can be enhanced using AI techniques. (Dam *et al.*, 2019) provided profound evidence that APM tools efficiency is in the hands of automated processes developed using AI. The premise for this thesis is to apply AI techniques to achieve the following objectives below:

1. To groom PB with NLP
2. To assign tasks to developers

3.1.1 Aim

Enable agile teams to spend more time delivering right solutions with reduced sprint planning time and effort.

3.1.2 Input

- PO's responsibilities
 - Provide unstructured text file which comprised of requirements to be processed.
- Developers:
 - provides proposed time frame for each task to be developed.

3.1.3 Output

- User stories
- Tasks
- Feasible tasks to fit sprint backlog

3.2 Methodology

NL and Hungarian algorithm were integrated to support in the automation of two activities involved in Scrum process (1) the grooming of the PB, and (2) task assignment for sprint planning. NLP was mainly utilised to address the following objectives: the decomposition of epic to USs, and USs to tasks. Hungarian algorithm attempted sprint planning problem by assigning task to developers. These techniques were interconnected to guarantee interoperability amongst each other. The integration of these techniques presented a smart agile project management (SAPMT) tool that advocates in decision making during the sprint planning phase in Scrum methodology until the release of the potential shippable product increment. The Figure 3.1 illustrate the enhanced Scrum process proposed.

3.3 Enhanced Scrum process

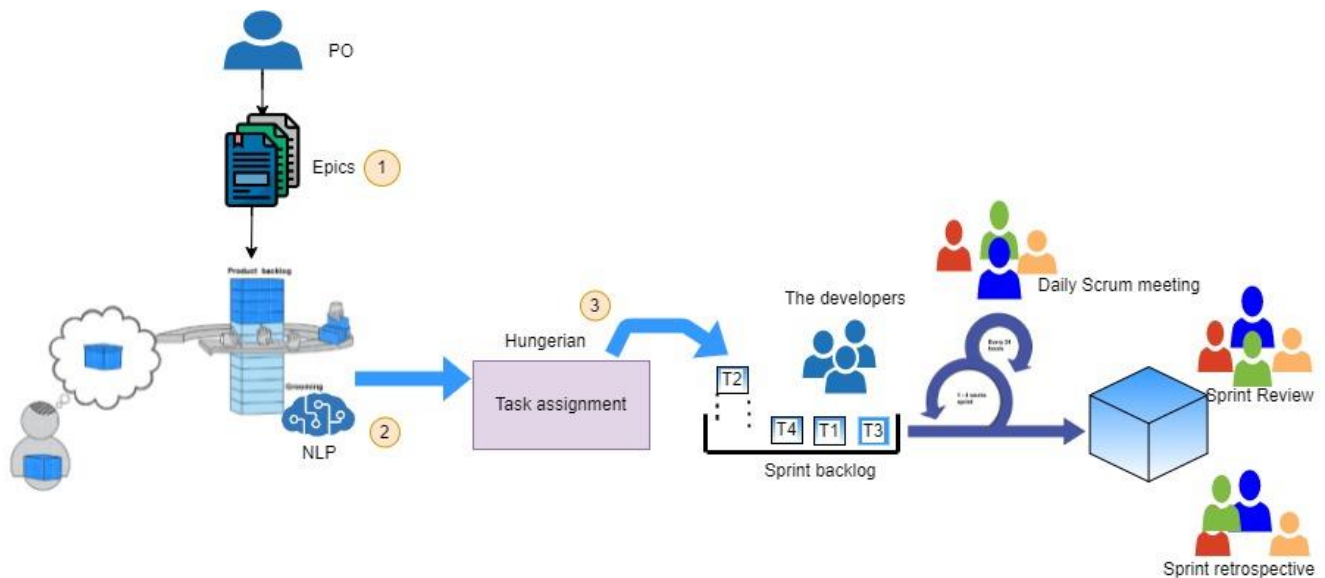


Figure 3.1: Proposed Scrum process

1. The PO collects epic requirements from the customers and save them into PB.
2. A pool of requirements is inserted into PB where refinery of large PBIs is decomposed using NLP (stanza) techniques proposed in chapter 4. After these requirements are decomposed into USs, they are normally assigned priority values using either MoSCoW or AHP. Most important PBI are moved to the top level in the PB called the definition of ready (DoR) for the next iteration. However, our work will only concentrate on the decomposition of epics using NLP.
3. After the epics are decomposed, now we apply task assignment model to assign the decomposed task to developers.

3.4 User story template

To form a US using a text generative model, it is important to define the model this thesis followed for validation purposes. There is a limited number of templates proposed in the literature for US format. The template popularised by (Cohn, 2004) state that the US should follow the template, AS a <user role> I want to <goal> so that <benefit>. The user role references the who part of the story while goal focusses on the what functionality does the story delivers and why focusses on the benefit part of the story (i.e., it answers what value will the story bring If implemented or show why it is essential to have such story in sprint). In most cases when automating the US generation from unstructured text using NLP, the last part of the US (benefit) is omitted (Lin *et al.*, 2014; Gunes and Aydemir, 2020). Since the last part of the US template is optional, this thesis has omitted it. Thus, the template that this thesis use is as follows:

Template = As <user role> I want to be able to <goal>. This template has been used before to generate the USs from requirements specifications using NLP (Pereira, 2018).

3.5 Dataset

This Thesis have formulated its own dataset to validate the feasibility of applying the SAPMT in Agile processes (grooming and sprint planning). However, this thesis tried to bring the project to real life situations when constructing the data by using one of the prominent case studies in software engineering industry. The dataset used was relying on assumptions as these is not an empirical study, we therefore used ATM and Ecommerce in our case studies. ATM and Ecommerce have been studied for last two decades now, these systems have diverse requirements and there is always room for improvement concerning the proliferation of technology. During the elicitation of both case studies, the requirements were written in an active voice for the system.

Since the research falls under text generative models, we had to perform data transformation before using our dataset to enhance the performance of our tool. Data transformation was performed using python library named regular expression (regex); It helps on the elimination of stop words and unwanted text from the dataset that reduces the accuracy of the tool. The rationality to choose this library was that it offers freedom to developers to express their creativity without limiting them and its fast.

3.6 Attributes of the research

3.6.1 Grooming PB with NLP

3.6.1.1 *Decomposing user stories into manageable stories.*

The decomposition of USs was performed prior to sprint planning due to the gathering of the necessary feedback from the stakeholders and the system's customers. The PBIs in the definition of ready state were identified and examined. The stories which comprised of more than one action verbs on their analysed linguistic structure were referred to as epics. An epic is the US that constitute large functionality that cannot fit into a single sprint. Therefore, they need to be on their simplest granularity, tasks before inserted in the SB. To demonstrate how an epic was coarse-grained into manageable stories, consider an example in chapter 4 in section 4.4.2.2. Spacy-stanza was used as the tool to process all the information needed to formulate the US from epics.

3.6.2 Hungarian algorithm

Hungarian algorithm is classical combinational optimisation task assignment model which dates back as 1955 proposed by Harold Kuhn to find the minimum total cost of job assignment to each worker. To find the minimal total cost, the problem is addressed as the square matrix of the costs of workers executing tasks. Figure 3.2 Illustrated how the algorithm attain the minimal cost. The algorithm solves the problem in a polynomial computational time complexity $O(n^3)$ for any $n \times n$ assignment problem. Figure 3.2

shows flowchart of Hungarian algorithm. The algorithm attain minimum cost can be mathematically described by equation 3:1.

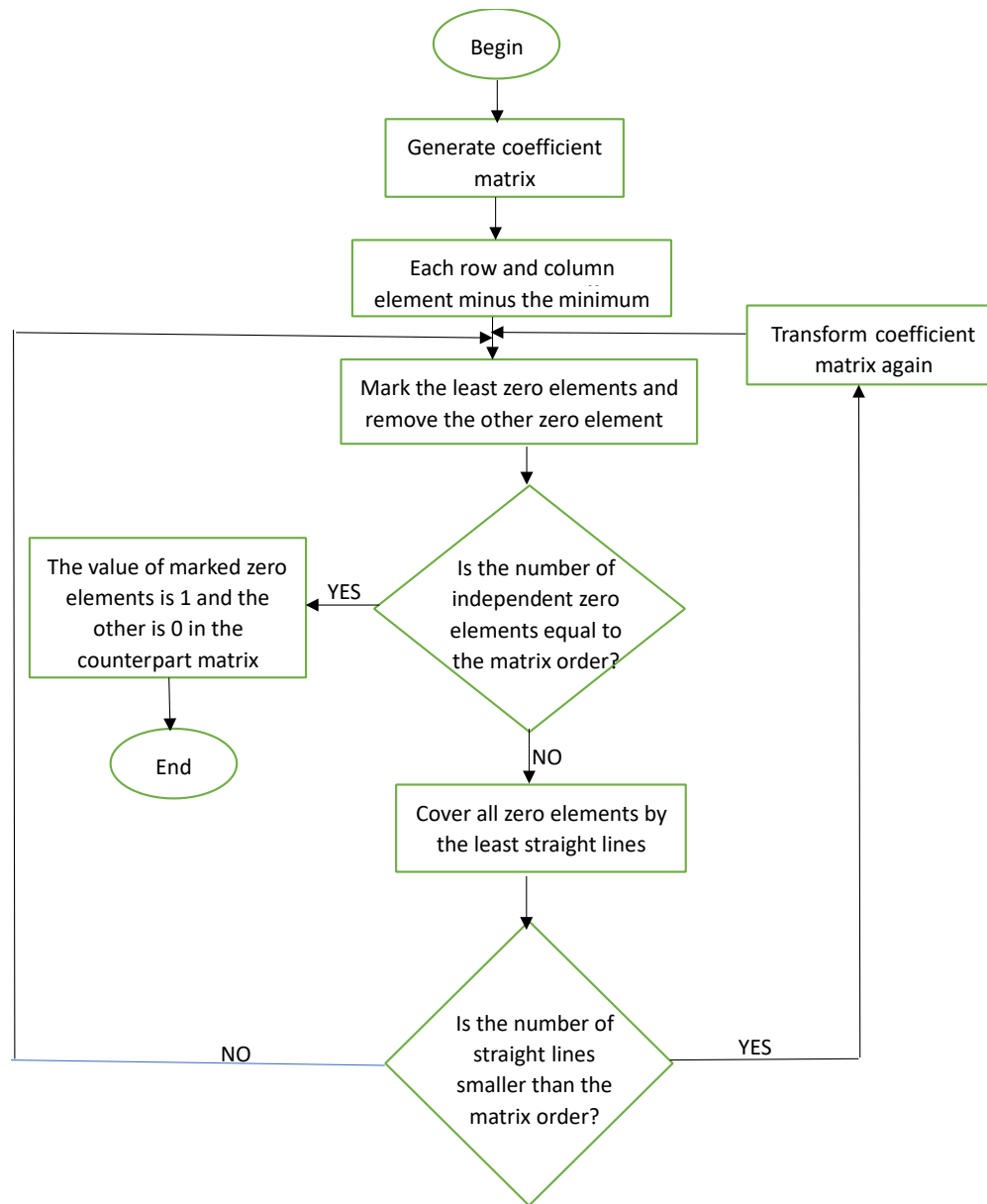


Figure 3.2:Hungerian algorithm flowchart

Although Hungarian is a classical solution for task assignment, it is still applied to current models as the evaluation tool that measures their accuracy and performance of newly developed task assignment models.

$$\text{minimize } f_p(x) = \sum_{i=1}^n \sum_{j=1}^n c_{ij}x_{ij},$$

3:1

$$\begin{aligned}
 & \text{subject to } \sum_{j=1}^n x_{ij} = 1, i = 1,2,3 \dots, n, \\
 & \sum_{i=1}^n x_{ij} = 1, i = 1,2,3 \dots, n, \\
 & x_{ij} > 0, \quad i, j = 1,2,3 \dots, n
 \end{aligned}$$

3.7 Summary

This chapter presented the core objectives of this thesis followed by an overview of automated Scrum process. The two objectives of designing this framework were identified and outlined as follows: the decomposition of epics and task assignment. Fulfilling these objectives will answer to the limitations of existing APM tools mentioned in problem statement. Both Hungarian algorithm and NLP techniques were integrated to form enhanced Scrum process. The integration of these approaches was seamlessly integrated and helped us to propose the enhanced Scrum process outlined in section 3.3. Among these subsystems, NLP model is responsible for generating USs from unstructured text file which comprised of epics and later transform this USs into tasks. The Hungarian was utilised to resolve task assignment. We concluded the chapter by discussing benefits of using Hungarian algorithm for task assignment model.

4. CHAPTER FOUR.

SYSTEM ARCHITECTURE AND IMPLEMENTATION

4.1 Introduction

In reference to the boundary conditions that are associated to the complexity of the prevailing APM tools as identified in the problem statement, this chapter is about the description of the design, system architecture and implementation of the tool that is accountable of the mentioned requirements in problem statement.

The architecture was designed in such a way that it accommodates scalability. Object oriented programming paradigm was adopted to provide flexibility and code reuse, while trained AI models were used to provide scalable solution and high accuracy. Furthermore, a web application was designed to provide heterogenous accessibility of the developed tool. That is, it can be accessible to any device that possess Internet or web browser. Etc., phone, laptop, and tablet.

The tool was developed using micro-python framework, Flask, MySQL, and Tailwind CSS framework. Python was selected because it serves as the multipurpose programming language with large community support, and it also provides extensive documentation and tutorials.

In the subsequent sections, we present the design approach, high-level system architecture, inputs expected for the system to function, drill down of system implementation, and expected output. Then, in section 4.11, we summaries the chapter.

4.2 Design approach

In this thesis, we're proposing called SAPMT. The preliminary approach to achieve the implementation of this tool initially starts by designing a high-level architecture and model the system using activity diagrams.

For the system to function, the PO issues unstructured text file which comprised of desired project's documentation to the developed system as input. The input is processed using the Restful technologies from python Flask framework and regular expression for NL data cleaning. The system is a web-based application which entails the sublayers which are proficient to filter and process attributes which acts as inputs data for different pipelines on the system.

4.3 Design goals

The goal was to design and implement a SAPMT for requirements engineering especially in the context of Scrum methodology that can automate the generation of Agile USs and tasks from epics. Later assign the decomposed tasks to available developers using Hungarian algorithm without taking away Scrum's agility processes. We hypothesised that the generated artefacts could help project managers (PMs) and PO in sense that it can lessen the time spend on requirements elicitation. We tried to align our work with the Agile framework called Scrum.

4.4 System architecture

In this thesis, we have adopted some of the concepts from blackboard architecture due to its flexibility to integrated several algorithms and data representation into a coherent and flexible computational framework (Hayes-Roth, 1985). The Figure 4.1 illustrates the architecture of the proposed tool. The architecture is composed of system components, mainly dependent on NLP engine and web services and Hungarian model. This architecture deviates from a decision support system for sprint planning proposed by Khabbazian *et al.* (2018) by one element. Although our proposed architecture adapted their task assignment model, we enhance the architecture by adding new feature, NLP engine which is responsible for the decomposition of epics.

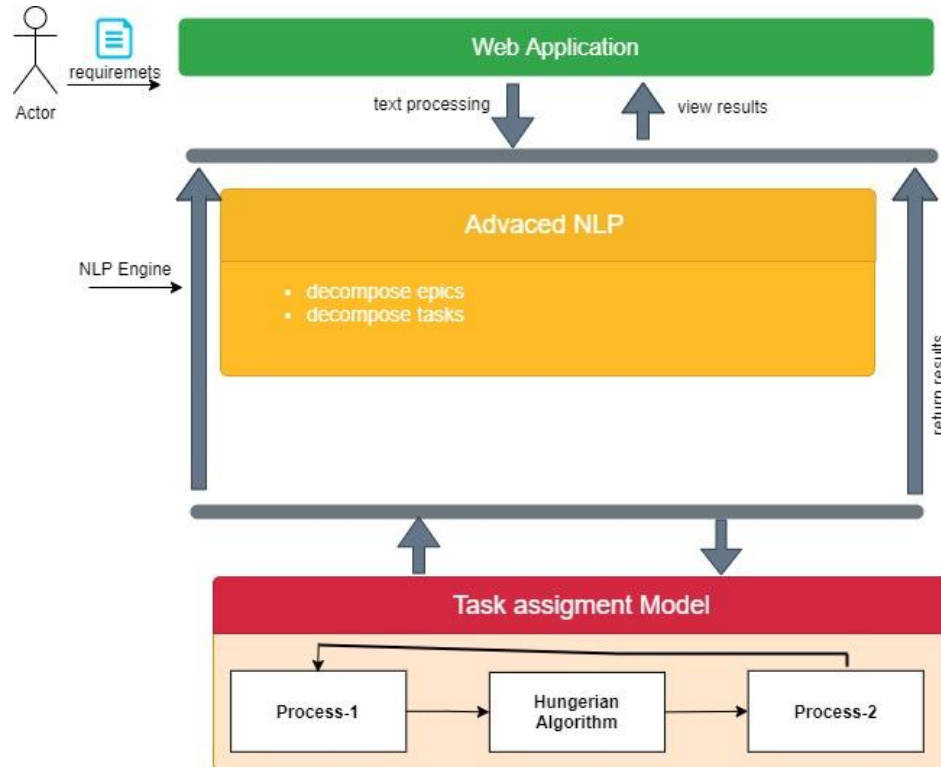


Figure 4.1: Proposed high-level architecture

4.4.1 components of architecture.

- Web application: presents results to the end user using RESTful web services. The web services are also used to send data to NLP pipelines where it's processed to get an output.
- NLP Engine
 1. Advanced NLP: return the generated Agile artefacts (USs and tasks).
- Hungarian algorithm: assign tasks to competent developers.

4.4.2 NLP engine 3

The following diagram in Figure 4.2 illustrated high-level design of decomposing USs and tasks. The processes are divided into two pipelines to enhance the tool's performance: first pipeline and second pipeline. The first pipeline carries the most common NLP functions which will be described in section 4.4.2.1.

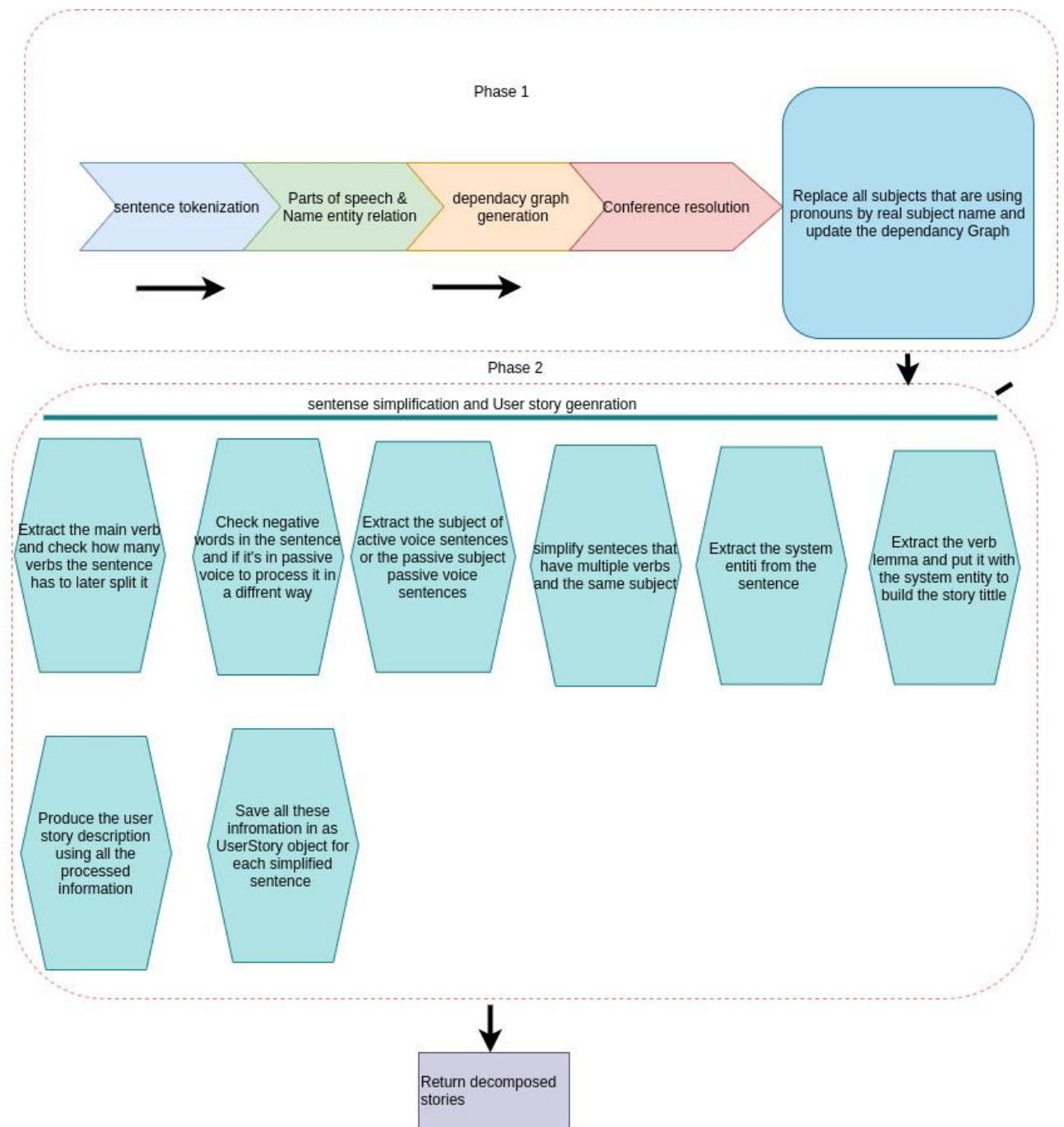


Figure 4.2: Outline the design process of decomposing epics

Adapted from (Pereira, 2018)

4.4.2.1 First pipeline implementation

There are of six processes executed in this stage: sentence segmentation, POS, NER, dependency graph generation, conference resolution and the replacement of the subject mentions with their pronouns (Pereira, 2018). This thesis has adopted the use of new python NLP libraries called Stanza and Spacy-Stanza. The first four tasks were resolved by utilizing spacy-stanza pipelines and annotations which returned the POS tag of words, lemma and dependency graph of text provided as an output. As mentioned in chapter 2, the dependency graph returns syntactic relationship between words in a sentence. The graph can consist of POS tags, root, etc. To visualize the dependencies, we imported displacy library from spacy¹.

The subsequent step resolved conference resolution between the sentences by utilizing Stanza CoreNLPCClient interface annotators. This process simply replaces the pronouns by their correlated subject names or noun present in the Mentions. If similar subjects refer to the same pro/noun on the text given, the algorithm returns none, and continues with the output of the current dependency graph where the graph's metadata acts as the input to the second pipeline. For demonstration purposes, we have illustrated the process of attaining the first output pipeline by using the text below.

Sentence 1:

- *“The bank Administrator views customer profile. But he cannot delete transactions history”.*

After the text went through all processes from the first pipeline, the output comes as modified text below:

Transformed sentence:

- *The bank Administrator views customer profile. But the bank administrator cannot delete transactions history.*

It is worth noting that the algorithm identified “The bank administrator” and “he” as the subjects in both sentences. However, the pronoun “he” refers to the same subject as “the bank administrator”, therefore the algorithm suggested to replace the pronouns its respective subjects. Therefore, this triggers the dependency graph to be updated and return the modified text. Figure 4.3 and **Error! Reference source not found.** shows the dependency graph after performing conference resolution. shows the programmatic output of the dependencies excavated from the sentence 1 while 4.4 shows the output of dependencies in a graphical form.

¹ <https://spacy.io/>

```

The DET det
Bank PROPN compound
Administrator NOUN nsubj
views VERB root
customer NOUN compound
profile NOUN obj
. PUNCT punct
But CCONJ cc
The DET det
Bank PROPN compound
Administrator NOUN nsubj
cannot AUX aux
delete VERB root
transactions NOUN compound
history NOUN obj
. PUNCT punct

```

Figure 4.3: programmatic dependency output

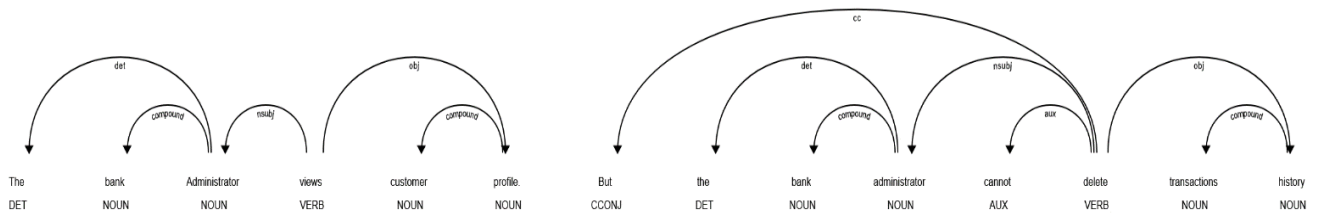


Figure 4.4: Dependency graph visualized by displacy

4.4.2.2 Second pipeline implementation 4

The second pipeline receives metadata from the first pipeline's output and execute the most fundamental NL tasks to decompose the epics into USs. Having the input as dependency graph from the first pipeline, it permits us to perform text analysis and extract essential information to generate USs information. For instance, to extract the user/actor of the US from dependency graph, we have implemented a function that facilitates extraction of the subject representatives of nouns and returns them as a list. This was accomplished by extracting a word with POS tags NOUN, PROPN and dependency token either nsubj, or nsubjpass, compound. The code illustrates how to extract the user/actor for the US generation.

```

def subject_extraction ():
    for sent in doc. sentences: # perform sentence segmentation
        for word in sent. words: #
            if 'nsubj' in word. deprel and word.pos == 'PRON, NOUN': #

```

```
    subject_words.append(word.text)
return subject_words
```

The first line from the code performs the sentence segmentation while the second and third lines perform tokenization. The fourth line finds all the words that are subjects with active voice (nsubj) by using the dependency attribute `deprel`.

The items from this list are retrieved later to form part of the US's information, `user /actor`. For instance, the template that this thesis followed is given as:

Template: "As "+ `<user/actor>` + "I want to be able to "+ `<phrase>`.

To generate the US, the algorithm starts by counting number of verbs as indicated in activity diagram in Figure 4.7. The processing is given in two categories, single verb and multiple verbs. For each category, there are different steps to follow until the USs and tasks are generated.

4.4.2.3 Algorithm for single verb sentences 4

The process initiates by counting the number of verbs present in each sentence. Most of the algorithm steps followed is activity diagram in Figure 4.7 which are self-explanatory. After that, we determined the subject from the sentence using the dependency graph by finding the keyword `key nsubj`. The `nsubj` denotes that the sentence is in active voice and it's a nominal subject, while `nsubjpass` denotes that the sentence it's in passive mode. However, this thesis focusses mainly on text written in active mode. We therefore discard all text written in passive mode.

For demonstration purpose we will continue with the first sentence from the first pipeline.

- *The bank Administrator views customer's profile.*

Following the algorithm proposed by (Pereira, 2018), iterate through the entire dependency graph and count the number of verbs in each sentence by finding their POS tag with keyword `VERB`. It is worth noting that this research only implemented the USs that are in active mode. The activity diagram in Figure 4.7 illustrates the algorithm that is utilised to attain USs and tasks. From the above text, 'views' was found as verb. Find the position of the identified verb and start the partial sentence generation from the index of the verb until end of the sentence. Save this information in a string variable called partial phrase. The figure 4.5 shows the code snippet that this thesis has developed to identify the index of the verb and the generate partial sentence.

```

for token in doc_sents[i]:
    i = 0
    if token.pos_ == 'VERB':
        new_phrase = []

        phrase = [word.text for word in doc_sents[i]]

        new_phrase.append(phrase[token.i:doc_sents[i].end])

```

Figure 4.5: Code snippet for partial sentence generation

The first line of the code finds tokens found in a sentence. The `doc_sents[i]` in this case indicates the specific sentence we are focusing on with single verb. Line 3 identifies the verb from the sentence by using POS tag VERB. After finding the position of the verb in a sentence, get the index of the verb by using keyword `token.i`. The next task was to determine the end of the sentence. As indicated from line 6, the end of sentence was found by using the keyword `end` with the sentence. After finding the position of the verb and the end of the sentence, construct a partial sentence by starting the text generation from verb's index to the end of the sentence. Store this information in a string called partial phrase. Line 6 of the code performed exactly partial text generation and gives the output below.

output:

Partial phrase = views customer's profile.

The subsequent step followed was to eliminate the punctuation at the end of sentence. This thesis has used the power of regular expression (regex) for text processing to remove the punctuation from partial phrase. After that, replace the verb from partial phrase with its verb lemma. Figure 4.6 shows a code snippet that illustrates how to replace verb with its lemma.

```

new_ = new_.replace(token.text, token.lemma_)
sentence = ''.join([str(e) for e in new_]) #''.join(map(s

```

Figure 4.6: Replacing verb with its lemma

The ultimate step left is now to collect the pieces of information that attribute towards the formation of US (subject and partial phrase]. Use this information to form US using template in section 1.

From the template we get the following US when using sentence 1.

Template = [subject/actor] I want to be able to [partial phrase]

Subject: The bank Administrator

Partial phrase: view customer's profile.

It is also imperative for the US formed to possess an object. This was extracted from partial phrase's dependency graph by identify word with dependency keyword `obj`. If the object exists, utilise the US's template to fill in the corresponding missing information to form US. The results of US for above text were found as:

User story: As the bank Administrator I want to be able to view customer's profile.

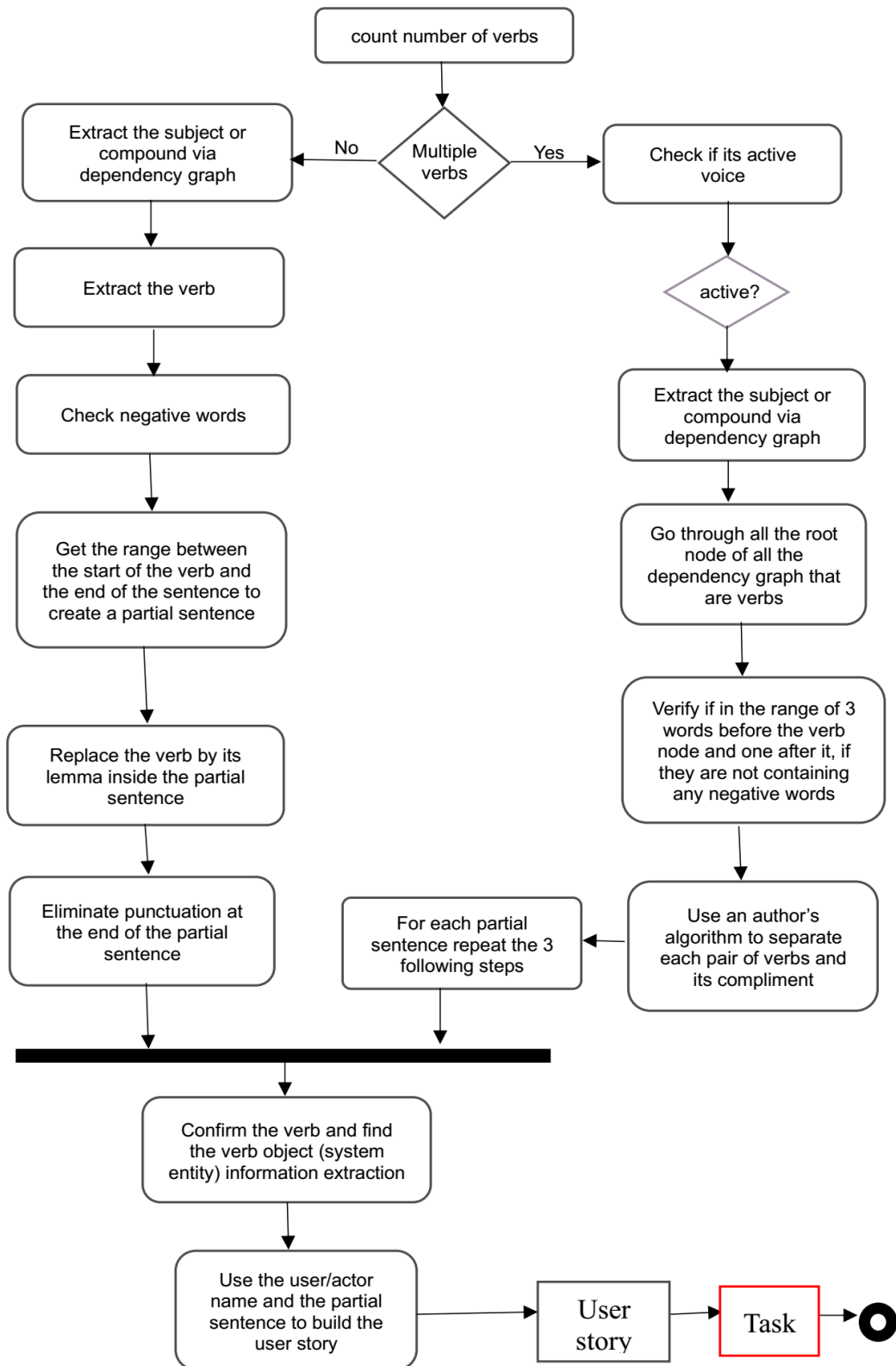


Figure 4.7: Activity diagram decomposes user story and task

4.4.2.4 Algorithm for multiple verbs.

In case the sentence is comprised of multiple verbs that are in active voice, use dependency graph to determine subject or compound from the sentence given. Furthermore, utilise the dependency graph's metadata to extract the presence of root nodes that are verbs from the entire sentence. Save the index of each verb found from the input text. Then, initialise the splitting process to form partial phrases. Spitting process was summarised by the following steps below:

1. Get the root verbs and their index positions from the entire sentence. Save this information in a list
2. Count three words before reaching the next root verb in the sentence and append this text to the root verb.
3. Form partial sentences with the output of step 2.
4. Verify the presence of an object (obj) from the partial sentences by searching through the dependency graph. We have illustrated this process by using code snippet in Figure 4.8.

```
doc2 = nlp(template)
for token in doc2:
    if 'obj' in token.dep_ :

        #stories.append(template)
        tasks.append(partial_sentence)
```

Figure 4.8: Determine the presence of object in generated phrase

5. Refine all unnecessary conjunctions words that complicates the US to avoid ambiguity.
6. From the partial sentence formed, replace the verb with its lemma.
7. With the information from step 6, retrieve the subject fill in the US's template with its corresponding text. Remember template = As [subject], I want to be able to [partial phrase].
8. Use chunking to extract the tasks from obtained USs generated.
9. Repeat all the steps until the entire text is processed.

For illustration purposes, we have used the following.

- The bank customer can withdraw money from ATM without card. The bank customer can also deposit money on the ATM, change PIN on ATM and transfer money from current account to saving account using the ATM. If money is deposited or withdrawn from account, customer receives SMS notification.

Looking at the text provided, the subject of the sentence never changed, thus, conference resolution just continues using the bank customer as the subject through the entire text. Therefore, the output of the dependency graph that is fed to second pipeline for above sentence is given in Figure 4.9.

```
The --> DET --> det
bank --> NOUN --> compound
customer --> NOUN --> nsubj
can --> AUX --> aux
```

```

withdraw --> VERB --> root
money --> NOUN --> obj
from --> ADP --> case
the --> DET --> det
ATM --> NOUN --> obl
without --> ADP --> case
card --> NOUN --> obl
. --> PUNCT --> punct
The --> DET --> det
bank --> NOUN --> compound
Customer --> NOUN --> nsubj
can --> AUX --> aux
also --> ADV --> advmod
deposit --> VERB --> root
money --> NOUN --> obj
on --> ADP --> case
the --> DET --> det
ATM --> NOUN --> obl
, --> PUNCT --> punct
change --> VERB --> conj
PIN --> NOUN --> obj
on --> ADP --> case
ATM --> NOUN --> obl
and --> CCONJ --> cc
transfer --> VERB --> conj
funds --> NOUN --> obj
from --> ADP --> case
the --> DET --> det
current --> ADJ --> amod
account --> NOUN --> obl
to --> ADP --> case
savings --> NOUN --> compound
account --> NOUN --> nmod
using --> VERB --> acl
ATM --> NOUN --> obj
. --> PUNCT --> punct
If --> SCONJ --> mark
money --> NOUN --> nsubj:pass
is --> AUX --> aux:pass
deposited --> VERB --> advcl
or --> CCONJ --> cc
withdrawn --> VERB --> conj
from --> ADP --> case
account --> NOUN --> obl
, --> PUNCT --> punct
customer --> NOUN --> nsubj
receive --> VERB --> root
SMS --> NOUN --> compound
notification --> NOUN --> obj

```

Figure 4.9: The output of the dependency graph

The first sentence contains the single verb which is, withdraws. The verb was replaced with its lemma, withdraw. Following the algorithm in section 4.4.2.4., we got the output of the first sentence as:

US: As the bank Customer, I want to be able to withdraw money from ATM without card.

Task: withdraw money from the ATM without card.

The next sentence contains multiple verbs, deposit, change, transfer, deposited, withdrawn and lastly receive. We followed the steps outlined in section 4.4.2.4 to attain USs for different verbs. The output of the USs and tasks was given as below:

US: As the bank Customer, I want to be able to deposit money on the ATM.

Task: deposit money on the ATM

US: As the bank customer, I want to change Pin on the ATM.

Task: change PIN on ATM

US: As the bank Customer, I want to be able to transfer money from current account to savings account using ATM

Task: transfer money from current account to savings account using ATM.

User story: As the bank Customer, I want to be able to receive SMS notification.

4.4.3 Extracting tasks from user stories

After the generation of USs is complete, we extracted useful insights from the decomposed USs to form tasks associated with those decomposed stories. To decompose USs to tasks, this thesis builds on the guidelines provided by an empirical study on how to formulate tasks from a given USs by applying NLP techniques (Müter *et al.*, 2019). We have also distilled grammatical patterns that generate the task from the given US by using chunking technique through the aid of Spacy-stanza annotations and pipelines. To be more specific, verb phrase detection was the most effective chunking technique we employed. A verb phrase is a syntactic phrase which consist of at least one action verb. This verb can be trailed by other chunks, such as object phrases, noun phrase etc.

The tasks were extracted by analysing partial phrases from the US information. This thesis determined the rules that govern the determination of tasks from their US linguistic structure linguistic task stricture.

- Initial word should be verb a with dependency tag 'root'

Therefore, the pattern of finding the tasks from US was given as:

Pattern = 'r(<VERB>? <OBJ>*<NN>+)

For demonstration purpose, consider the formulated US at section Algorithm for single verb.

The output of the task was found as:

Task: view customer's profile

4.5 Tasks assignment to the developers.

This section shows how the designer of this thesis has assigned the obtained tasks from the USs to the developers using Hungarian model. While using Hungarian, the number of tasks should be equal to the number of developers. We therefore count the number of tasks extracted from the USs and equate them

to the number of developers. Table 4.1 below shows the tasks that are available from SB waiting for task assignment model.

Table 4.1:Task extracted from ATM text

| ID | Tasks |
|----|--|
| 1. | withdraw money from ATM without card |
| 2. | deposit money on the ATM |
| 3. | change PIN on ATM |
| 4. | transfer money from current account to savings account using ATM |

Table 4.2:Available developers

| ID | Developers |
|----|------------|
| 1. | Thabo |
| 2. | George |
| 3. | Albert |
| 4. | Clyde |

Developers compete for task assignment by issuing their proposed delivery time to complete each task. This information is received secretly by the PO who model this information in a cost matrix format which is suitable for Hungarian algorithm to process it. The secrecy provides the transparency as developers would not base their opinions on their colleague’s decisions. From Table 4.1 and Table 4.2, the number of tasks equate to the number of developers, thus, each developer will implement a single task. Both tables were modelled as the cost matrix with the proposed time frame below. The aim was to find optimum sprint with less development cost while maximising value.

Table 4.3:TO-DO Table (iteration 1) Hungarian

| | | | | |
|-------|--------|--------|--------|--------|
| TO-DO | Task 0 | Task 1 | Task 2 | Task 3 |
|-------|--------|--------|--------|--------|

| | | | | |
|--------|---|---|---|---|
| Thabo | 3 | 6 | 4 | 2 |
| George | 5 | 1 | 7 | 5 |
| Albert | 2 | 4 | 3 | 6 |
| Clyde | 4 | 2 | 4 | 5 |

Table 4.4:After applying Hungarian algorithm

| TO-DO | Task 0 | Task 1 | Task 2 | Task 3 |
|--------|--------|--------|--------|--------|
| Thabo | 3 | 6 | 4 | 2 |
| George | 5 | 1 | 7 | 5 |
| Albert | 2 | 4 | 3 | 6 |
| Clyde | 4 | 2 | 4 | 5 |

Following the activity diagram in section 3.6.2, we obtained the results of the task assignment model as show in Table 4.5. The numbers highlighted in blue colour in Table 4.4 are results proposed by Hungarian algorithm.

Table 4.5:Iteration 1

| Iterations | Iteration 1 | |
|------------|-------------|----|
| | Task | TT |
| Thabo | Task 3 | 2 |
| George | Task 1 | 1 |
| Albert | Task 0 | 2 |
| Clyde | Task 2 | 4 |
| | Total TT | 9 |

We have used pseudo names for tasks obtained from Table 4.1. For example, Task 1 is mapped with withdraw money from ATM without card. However, the real tasks will be displayed during task assignment while using the system.

4.6 Implementation details

4.6.1 Web application

After considering different possibilities to present the tool, a web-based platform was selected as the potential solution. The rationality to select the web-based platform was the ability to provide heterogeneous accessibility to the developed tool. To attain this, we have adopted the Tailwind CSS framework (Tailwind, 2019) and presented our tool as a dashboard application that facilitates the communication between the development team to enhance their decision making. Figure 4 illustrates the implemented tool. The tool recently supports four project management activities, mentioned on the objectives.

To guarantee seamless accessibility to the developed tool, we designed the application using responsive web design (RWD) principles. RWD states that way the users interacts with the developed system must be same regardless of the device used to access the system (H. Gillbert Miller, 2011). The web application designed using these principles adopt the flexible layout by harnessing the power of cascades styles sheets (CSS3) media queries.

It is crucial that the tool supports PO activities and the development team activities. The user interface currently supports the following PO activities: create, update, and delete task from the PB and to approves estimated requirements' priorities made by AI model. Create option is utilised to generate new tasks, while update is used to edit the parameters of tasks and lastly, delete option is used to remove selected element from the Prioritised PB.

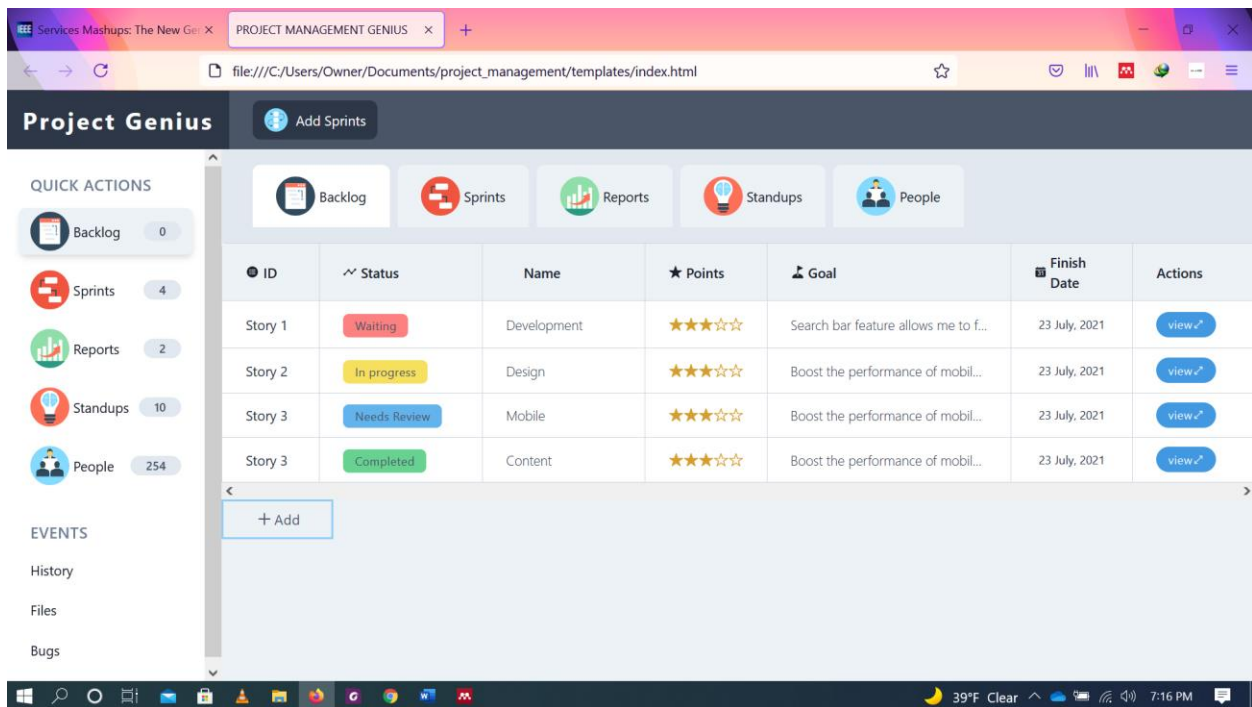


Figure 4.10: The user interface of the developed tool

4.6.2 Database

The database module is utilised to save information in a persistent and consistent way and has no distinct functionalities. The web application is interconnected to the database to perform the read and write functionalities. The database is a significant module of the SAPMT as it is where all the data issued by the users is stored.

4.6.2.1 Model

The SAMPT utilises a relational database as its storage system. There are currently numerous other alternatives, but relational database was preferred due to their extensive usage, and they permit the practical implementation of the models. It is not the scope of this thesis to explore alternative methods to store information.

4.6.2.2 Database management system

There is diverse array of database management system (DBMS) platforms that implement a relational model. Despite having different features, all those DMBS have common base; provide data abstraction in a tabular manner which is easy to use and understand. So, the rationality to select the DMBS was based on opensource, reliable technical support and popularity. To develop SAPMT, we utilised MySQL which is an open-source database which has the set of comprehensive advanced features, management tools and technical support to achieve the highest levels of MySQL scalability, security reliability and uptime. MySQL is the most popular opensource DBMS which is mostly integrated with web applications

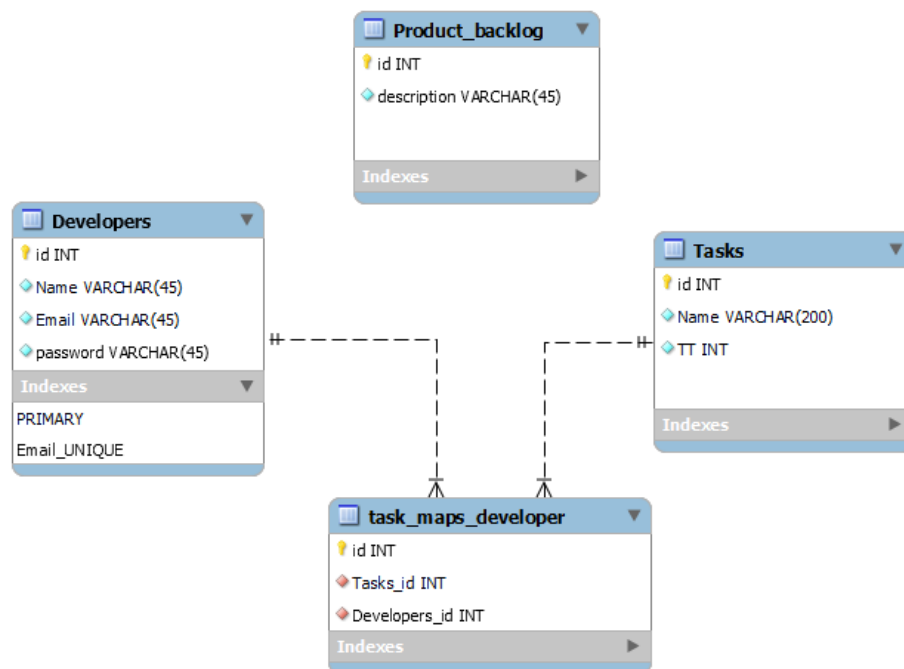


Figure 4.11:Database schema of developed tool

4.7 Libraries used

To achieve the goal of this research, we have used different kinds of NLP libraries, Spacy-stanza, Stanza, Stanford CoreNLP. But the main library which offered diverse number of functionalities was Stanza. Stanza is the python NLP library which was renamed after its predecessor Stanford CoreNLP (written in java). During the writing of this thesis, the library currently supports diverse array of language up to 66 official languages in the world (Qi *et al.*, 2020). It is one of the best efficient NL libraries as compared to its competitors spacy and NLTK. The tool supports the functions like sentence segmentation, POS, Lemmatization, Name Entity Recognition (NER), sentence sentiments analysis etc. Additionally, it sources java packages to enhance its functionality: to resolve conference resolution on text, it accesses Stanford Server toolkit via Stanza CoreNLP interface which is written using native python. It also provides flexibility to allow developers to customise their annotators and pipelines. Lately Stanza was integrated with Spacy to provide visualisation of dependencies by using displacy.

4.8 Inputs and outputs

For the system to function, the input was given as a textual data in a form of a file. The file was be comprised of epic USs.

4.8.1 Product Owners inputs

The framework as it appears in the Figure 4.1:Proposed high-level architecture, it starts with the PO who is accountable for providing the detailed project documentation captured during. The documentation consists Agile epic requirements stated by the customer. This input was processed at different pipelines demonstrated on system architecture. The PO also participates in the entering information provided by the DT regarding task assignment.

4.8.2 Developers' inputs

The registered developers provide their credentials to access system and view tasks assigned to them. These credentials are passed to authentication subsystem where they are going to be validated to log on the system. The tasks are assigned to individual development team members are resolved by Hungarian model. The certain developer will be given administrative rights to approve or change the proposed sprint velocity by the PO. The tasks from definition of ready are selected by the sprint planner implemented by Hungarian algorithm to provide less risky sprint with ROI.

4.9 Task's selection model

The PO can start the Sprint planning as soon as there are present tasks on the SB such that tasks are assigned to developers by applying Hungarian algorithm. The task assignment model is divided into three subprocesses which are discussed on the subsequent sections.

4.9.1 Process 1

At this stage, the tool will check the presence of developers to equate them to equal number of tasks present on the SB. After equating process, results are sent to Hungarian algorithm as square matrix.

4.9.2 Hungarian algorithm

The Hungarian algorithm will use the TO-DO-Table as a square matrix to determine the shortest time cost and forward the results to the subsequent stage of the process (Khabbazian, 2018).

4.9.3 Process 2.

The output of the Hungarian algorithm will be processed to allocate the tasks for the developers. Under normal circumstances, the system will complete the allocation procedure for that task by copying the Hungarian algorithm result “Developer, Task, Time” to the Assigning-Table, submitting the iteration result in the Iterations-Table, and removing the Task from the TO-DO-Table. This is the progressive process, so the system will find available developers and return to work again from the first level of the processing (Process - 1) until the entire tasks have been allocated to developers.

4.10 Output

The output will be comprised of two tables: (1) Iteration tables and (2) Task assignment table. Iteration tables will consist of attributes such as developers' names, tasks they are bound to implement together with proposed total time (TT) which can be spent to implement such task. The second table is assignment table which will show all tasks that each developer has contributed towards its implementation. To encounter for sprint duration, the developers must work on the tasks assigned to them on iteration table.

4.11 Summary

We have presented the NLP and Hungarian based decision support system architecture to enhance the project managers and development team duties in the following activities, decomposition of epics and task assignment. The system consists of sprint planner which outputs the proposed Sprint in the form of iterations.

We harnessed the ability of Blackbox architecture to integrate two different technologies, NLP, and the Hungarian algorithm. This resulted into the design of the proposed high-level architecture described in section 4.4. NLP was responsible for the decomposition of epics into user stories and task while Hungarian algorithms were used for task assignment. Furthermore, we presented an outline of the design process of decomposing epics; the algorithm was divided into two sub sections (1) single verb and multiple verbs. This was done to enhance algorithm's performance. Both algorithms were discussed into more details, the complexity of multi verb's algorithm is more complicated than single verbs. The algorithm for multiple verbs is more complicated than single verbs. We further demonstrated how to extract the necessary information to decompose the USs.

We further describe the technology used to design and implement the tool, in aspect of database and user interface design. MySQL was selected as the best solution for data storage while Tailwind CSS was used to design the user interface. We stated expected inputs for the systems to function.

5. CHAPTER FIVE

EVALUATION OF THE DEVELOPED TOOL (SAMPT)

5.1 Introduction

In the previous chapter, we engaged in the description of system architecture and drill-down of how we have implemented the proposed tool that can help Agile teams efficiently use timeboxed sprints by applying minimal effort. These can enable Agile teams to spend more time delivering the right solutions with reduced sprint planning time and effort. Therefore, it is essential to evaluate the quality of the developed tool by executing case studies to conclude on the tool's performance.

To evaluate our solution, we started by discussing the methodology this thesis followed to validate the tool. Subsequently, give an example on how to attain the results from a given epic, assign the decomposed task to users using Hungarian algorithm, discuss results obtained and ultimately summarise the chapter.

5.2 Methodology

This chapter embarks on the validation of the developed tool and how it can be evaluated. To assess the validity of the SAPMT, we conducted a case study on ATM project and Ecommerce project. ATM is a bank machine which offer functionalities like withdrawals, deposit of money, change of PIN to the customers. Ecommerce is a web-based shopping platform that sells products to customers online. customers can purchase the products they like.

We build on the hypothesis that the implemented tool should obtain similar results to the methodology we have adapted. Therefore, this thesis has used IBM's payroll requirement text to validate SAPMT.

5.2.1 Example of evaluation

Since we build on the hypothesis that the tool will attain identical results to the methodology adapted in (Pereira, 2018), this thesis adapted the payroll system requirement from IBM to validate the efficiency and accuracy of the tool.

Input: Unstructured text from IBM

- *“The Payroll Administrator maintains employee information. The Payroll Administrator is responsible for adding new employees, deleting employees, and changing all employee's information such as tittle, address, and payment classification (hourly, salaried, commissioned) as well as run administrative reports.”*

Following the algorithm in **first pipeline**, the system starts the processing from sentence 1 where it identified that the first sentence contains single verb, Therefore, single story and task were generated as follows:

US => *As the Payroll administrator I want to be able to maintain employee information.*

Task => *maintain employee information.*

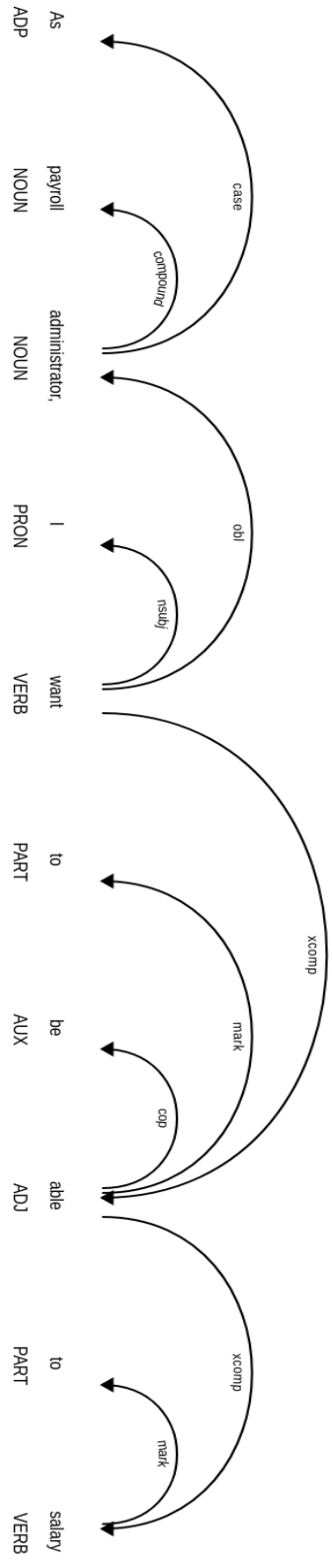


Figure 5.1: Graphical dependency output

After the text generation processes, USs and tasks were copied to the pandas. The table below shows all obtained USs from the IBM text.

5.3 Extracted user stories and tasks

This section shows results obtained after applying NLP for US and task generation. The tasks from text were sent to the database as To-Do list. Each task possesses a unique identity (id) to avoid replication. For instance, the task “*maintain employee information*” hold position 1 from Table 4.1:Task extracted from ATM text, therefore it was assigned an id of 1. All the ids are auto incremental by 1 and all subsequent tasks followed that sequence.

The results can be viewed on the Table 5.1. Table 5.1 shows the generated user stories and tasks from IBM's payroll system text.

Table 5.1:Generated user stories and tasks from IBM's payroll system text

| User story description | Task |
|---|--|
| As Payroll Administrator I want to be able to maintain employee information | maintain employee information |
| As Payroll Administrator I want to be able to add new employees | add new employees |
| As Payroll Administrator I want to be able to delete employees | delete employees |
| As Payroll Administrator I want to be able change all employee's information such as tittle, address, and payment classification (hourly, | change all employee's information such as tittle, address, and payment classification (hourly, |
| As Payroll Administrator I want to be able to run administrative reports | run administrative reports |

5.4 Task assignment process

5.4.1 Process 1

The PO decides when to initiate task assignment process. The PO receives all tasks from the decomposition process as pandas' data frame where all extracted tasks are sent to the database such that they can be viewed from the PB of the developed tool. Since Hungarian algorithm (HA) functions efficiently when assigned a square matrix (the number of tasks must be equal the number of developers available for implementation of tasks), the system will pick the total number of tasks equal to the developers. However, there are circumstances where number of tasks are less than the number of developers. Suppose there are 3 tasks with their unique identities (id) (id = 1, maintain employees'

information, id = 2, add new employees, id = 3, delete employees) for simplicity purposes, we gave them pseudo names as follows, task 1, task 2, and task 3:

- In case the number of tasks is less than the number of developers, the system present additional virtual task with zeros so that the matrix is squared

5.4.2 Applying Hungarian algorithm

The system consists of five developers who compete for each task. Each developer gives their proposed delivery time for each task and their answers are received by the PO who puts them in a cost matrix format such that Hungarian algorithm selects the best candidate to perform certain job. Table 5.2 shows the proposed delivery time recorded as cost matrix. Hungarian algorithm selects the optimal assignment for each iteration. The job of Hungarian is to retain the sprint or iteration at minimum development time with maximised quality.

5.4.3 Process 2

After Hungarian results are processed, each developer now knows what s/he will be focusing on concerning the next iteration. The PO will be authorised to view the tasks assigned to each developer while the developers will only view assigned tasks to them. It is significant to eliminate assigned tasks from the PB as this will evade repeating tasks that are already assigned to developers. Therefore, SAPMT saved all assigned tasks to assignment table in database with attributes, task id, developer, and Total time (TT). TT refers to the time each developer proposes to complete that task.

5.5 Sprint Iterations

This section illustrates the detailed description of task assignment model. After the completion of iterations, the PO will have access to view the resulted sprints assigned by the Hungarian algorithm. To accomplish this, the iteration must have registered developers who are willing to compete for task assignment by issuing their proposed time to complete a task. Since sprint planning is progressive process, it divided into subsections called iterations. This will give the DT to focus more on stories which are aligned to the defined goal. Table 5.2 shows the proposed time by different DT.

5.5.1 Iteration 1

5.5.1.1 Available developers

- London
- Mphaufe
- Tau
- Lehlohonolo
- Shale

Table 5.2: Square matrix for task assignment

| | | | | | |
|-------|--------|--------|--------|--------|--------|
| TO-DO | Task 0 | Task 1 | Task 2 | Task 3 | Task 4 |
|-------|--------|--------|--------|--------|--------|

| | | | | | |
|-------------|---|----|---|---|---|
| London | 5 | 13 | 4 | 8 | 2 |
| Tau | 4 | 1 | 7 | 2 | 6 |
| Mphaufele | 6 | 9 | 8 | 5 | 9 |
| John | 7 | 3 | 6 | 4 | 4 |
| Lehlohonolo | 8 | 4 | 4 | 7 | 5 |

Table 5.3:Hungarian results for iteration 1

| Iterations | Iteration 1 | |
|-------------|-------------|----|
| | Task | TT |
| London | Task 4 | 2 |
| Tau | Task 3 | 2 |
| Mphaufele | Task 0 | 6 |
| John | Task 1 | 3 |
| Lehlohonolo | Task 2 | 4 |
| | Total TT | 17 |

After Hungarian assign task to developers, results are populated to the assignment table with the following attributes (Developer's name, Task, Total Time (TT)).

The web application retrieves the obtained results:

- Each developer can view assigned tasks on their private dashboard after successfully login. To attain results, we have used the object relational mapper (ORM) which queries database in a high-level format.
- If the task is implemented, they can change the status of the task by selecting either busy, idle, or completed. This will help the PO and stakeholders to gauge the progress made regarding the project.

5.6 Case studies

5.6.1 Case study 1: ATM

- “The bank customer can withdraw money from the ATM without card. The Bank customer can also deposit money on the ATM, change PIN on the ATM and transfer funds from the current account to savings account on the ATM. The customer should be able to receive SMS notifications when money is withdrawn from the account.”

From the ATM case study, there are 5 USs and task expected:

Expect stories are from the following phrases:

- *Withdraw money on ATM*
- *Deposit money on ATM*
- *Change PIN on ATM*
- *Transfer funds*
- *Receive SMS notifications.*

To generate the stories, this thesis automatically identified indexes that has potential to generate the story from the given text. Table 5.4 shows the generated USs together with their corresponding tasks after applying our algorithm. First, the algorithm gives out indexes that will be used to form ranges that has potential to generate a story. For instance, suppose the index’s output was given as a list with values [12,22,24,34]. To generate the first story, the range could be given as 12:22, while the second story could be from 22:24 and the rest will consecutively follow the same pattern. However, note that the last number on the list does not have a neighbouring number to produce a range. Therefore, to solve this challenge, we have used the pythonic way of determining the solution. Since 34 is at the end of the list, python programming gives the last element of array as value -1. Therefore, the last index could be given as 34:-1.

Table 5.6 below shows the results of above input text where the second and third rows shows generated stories with their corresponding tasks. The results of applying the range can be observed at figure 6.6 during production appendix 4.

Table 5.4:Decomposed epics results from ATM text

| Index range | Generated user stories | Tasks |
|-------------|---|---|
| 4:11 | As the bank customer I want to be able to withdraw money from ATM without card | Withdraw money from ATM without card |
| 16:22 | As the bank customer I want to be able to deposit money on the ATM | Deposit money on the ATM |
| 22:27 | As the bank customer I want to be able to change PIN on the ATM and | Change PIN on the ATM and |
| 27:35 | As the bank customer I want to be able to transfer funds from the current account to savings account on the ATM | Transfer funds from the current account to savings account on the ATM |
| 35:42 | As the bank customer I want to be able to use the ATM if money is | to use the ATM if money is |
| 49: -1 | As the bank customer I want to be able to receive SMS notification | Receive SMS notification |

5.6.1.1 Task assignment

Since the output of Agile epics decomposition influences the sprint planning, for instance if the generated text does not the fit qualify criteria of the story especially the rule that specifies that the US should read like a sentence with no grammar mistakes there will be no tasks to process. Therefore, subsequent step was to apply task assignment model. We have used same developers from section 5.5.1.1 throughout the entire evaluation.

5.6.1.2 Iteration

5.6.1.3 Available tasks on the sprint backlog:

1. Withdraw money from the ATM
2. Deposit money on the ATM
3. Change PIN on the ATM and
4. Transfer funds from the current account to savings account on the ATM.
5. to use the ATM if money is
6. Receive SMS notification

5.6.1.4 Available developers

- London

- Tau
- Mphaufele
- John
- Lehlohonolo

Table 5.5 Hungarian matrix input 2

| TO-DO | Task 5 | Task 6 | Task 7 | Task 8 | Task 9 |
|-------------|--------|--------|--------|--------|--------|
| London | 2 | 4 | 3 | 1 | 1 |
| Tau | 5 | 7 | 6 | 3 | 4 |
| Mphaufele | 6 | 8 | 7 | 4 | 9 |
| John | 4 | 4 | 2 | 1 | 3 |
| Lehlohonolo | 5 | 7 | 9 | 8 | 6 |

The highlighted text in blue are the selected tasks after applying the Hungarian algorithm. The proposed minimum time was found to be 19. The results are indicated on Table 5.6: Iteration-Table (iteration 2) below. Table 5.6 shows the results of iteration

Table 5.6: Iteration-Table (iteration 2)

| Iterations | Iteration 2 | |
|-------------|-------------|----|
| | Task | TT |
| London | Task 9 | 1 |
| Tau | Task 8 | 3 |
| Mphaufele | Task 6 | 8 |
| John | Task 7 | 2 |
| Lehlohonolo | Task 5 | 5 |
| | Total TT | 19 |

5.6.2 Case study 2: Ecommerce

For this case study, suppose there is a company in Cape Town which is looking for the development of their Ecommerce websites and they specifies their requirements as:

Input text:

“The customers should be able to view products sold online. If the customer decides to purchase the products online, add products to the bucket where s/he can continue with the purchasing process as the

guest or create account for shipping purposes. They must be able to pay with visa cards or cash on delivery (COD). The system should validate expired cards to avoid scammers.”

Looking at the text provides above, there are 6 stories that could be extracted:

Expected stories are from the following phrases:

- View products sold online
- Purchase the product online
- Add products to the bucket
- Create account for shipping purpose
- Pay with visa card or cash on delivery (COD).
- Validate cards to avoid scammers

Table 5.7:Extracted user stories and tasks from Ecommerce text

| Index range | Generated user stories | Tasks |
|-------------|--|---|
| 6:8 | As products I want to be able to view products. | View products |
| 14:16 | As products I want to be able to purchase the products online | Purchase products online |
| 26: | As products I want to be able to add products to the bucket where s he can | Add products to the bucket where s he can |
| 41:50 | As products I want to be able to create account for shipping purposes they must be able to | create account for shipping purposes they must be able to |
| 65:69 | As products I want to be able to validate expired cards to ‘] | validate expired cards to ‘] |

5.6.2.1 Available tasks on the sprint backlog

- View products
- Purchase products online
- Add products to the bucket where she can
- create account for shipping purposes they must be able to

- validate expired cards to ']

5.6.2.2 Available developers

Since the number of tasks in sprint backlog are equal to the number of developers, we just continue with the assignment. Therefore, we use the same developers from section 5.6.1.4.

Table 5.8: The Hungarian matrix input 3

| TO-DO | Task 10 | Task 11 | Task 12 | Task 13 | Task 14 |
|-------------|---------|---------|---------|---------|---------|
| London | 3 | 4 | 4 | 5 | 2 |
| Tau | 4 | 6 | 3 | 5 | 7 |
| Mphaufele | 2 | 3 | 1 | 1 | 5 |
| John | 6 | 7 | 4 | 8 | 4 |
| Lehlohonolo | 1 | 4 | 9 | 7 | 6 |

Table 5.9: Iteration-Table (iteration 3)

| Iterations | Iteration 3 | |
|-------------|-----------------|-----------|
| | Task | TT |
| London | Task 11 | 4 |
| Tau | Task 12 | 3 |
| Mphaufele | Task 13 | 1 |
| John | Task 14 | 4 |
| Lehlohonolo | Task 10 | 1 |
| | Total TT | 13 |

5.6.3 Case study 3

Input text:

The user of the application can track his/her performance when running or riding his/her bike via the GPS. His/her performance can be saved to his/her account and shared with other friends from his/her social networks. The user cannot delete any entries once they are saved to the account. The user can create a report with all the activities by date range, or by type (running or biking).

The expected USs generated from this text of the second case study will be:

- Track his/her performance
- Save performance

- Share performance
- Cannot delete any entries
- Create a report

Table 5.10:generated user stories and tasks from agile samurai textbook

| Index range | Generated user stories | Tasks |
|-------------|---|---|
| 6:8 | As his I want to be able to track his her performance when' | Track his her performance when |
| 14:16 | As his I want to be able to ride his her bike via the GPS His her performance can be | ride his her bike via the GPS His her performance can be |
| 26: | As his I want to be able to delete any entries once they are | delete any entries once they are |
| 41:50 | As products I want to be able to have ability to | Have ability to |
| 65:69 | 'As his I want to be able to create a report with all the activities by date range or by type ('] | create a report with all the activities by date range or by type (' |

5.6.3.1 Available tasks on the sprint backlog

- Track his her performance
- ride his her bike via the GPS His her performance can be Add product to the bucket where she can
- delete any entries once they are
- have ability to
- create a report with all the activities by date range or by type ('

5.6.3.2 Available developers

We apply the same rules defined in section 5.6.2.2 to determine the available developers.

Table 5.11:Hungerian input 4

| TO-DO | Task 15 | Task 16 | Task 17 | Task 18 | Task 19 |
|---------|---------|---------|---------|---------|---------|
| London | 7 | 9 | 4 | 5 | 3 |
| Tau | 10 | 6 | 7 | 9 | 6 |
| Mphaufe | 5 | 3 | 4 | 2 | 5 |
| John | 6 | 8 | 5 | 7 | 4 |

| | | | | | |
|-------------|---|---|---|---|---|
| Lehlohonolo | 1 | 4 | 5 | 3 | 6 |
|-------------|---|---|---|---|---|

Table 5.12: Iteration-Table (iteration 4)

| Iterations | Iteration 4 | |
|-------------|-----------------|-----------|
| | Task | TT |
| London | Task 19 | 3 |
| Tau | Task 16 | 6 |
| Mphaufele | Task 18 | 2 |
| John | Task 17 | 5 |
| Lehlohonolo | Task 15 | 1 |
| | Total TT | 17 |

5.7 Output of iterations and task assignments

Table 5.13 consist of all iterations after applying the Hungarian algorithm. The proposed iterations optimised to deliver high quality with loss cost.

Table 5.13: Iterations-table

| Iterations | Iteration 1 | | Iteration 2 | | Iteration 3 | | Iteration 4 | |
|-------------|-----------------|-----------|-----------------|-----------|-----------------|-----------|-----------------|-----------|
| | Task | TT | Task | TT | Task | TT | Task | TT |
| London | Task 4 | 2 | Task 9 | 1 | Task 11 | 4 | Task 19 | 3 |
| Tau | Task 3 | 2 | Task 8 | 3 | Task 12 | 3 | Task 16 | 6 |
| Mphaufele | Task 0 | 6 | Task 6 | 8 | Task 13 | 1 | Task 18 | 2 |
| John | Task 1 | 3 | Task 7 | 2 | Task 14 | 4 | Task 17 | 5 |
| Lehlohonolo | Task 2 | 4 | Task 5 | 5 | Task 10 | 1 | Task 15 | 1 |
| | Total TT | 17 | Total TT | 19 | Total TT | 13 | Total TT | 17 |

Table

5.14: Task assignment

| Developer's name | Task name | Total TT |
|------------------|-----------|----------|
| London | Task 4 | 2 |
| London | Task 9 | 1 |
| London | Task 11 | 4 |
| London | Task 19 | 3 |

| | | |
|-------------|---------|---|
| Tau | Task 3 | 2 |
| Tau | Task 8 | 3 |
| Tau | Task 12 | 3 |
| Tau | Task 16 | 6 |
| Mphaufele | Task 0 | 6 |
| Mphaufele | Task 6 | 8 |
| Mphaufele | Task 13 | 1 |
| Mphaufele | Task 18 | 2 |
| John | Task 1 | 3 |
| John | Task 7 | 2 |
| John | Task 14 | 4 |
| John | Task 17 | 5 |
| Lehlohonolo | Task 2 | 4 |
| Lehlohonolo | Task 5 | 5 |
| Lehlohonolo | Task 10 | 1 |
| Lehlohonolo | Task 15 | 1 |

5.8 Results

This section concentrates on the measure of how accurate and efficient the SAPMT in is (1) generating Agile artefacts (USs and tasks) from unstructured text using NLP techniques and assign the attained tasks from USs to available developers using Hungarian algorithm. This thesis harnessed power of the classical machine learning called confusion matrix metric in classification problems with known answers. Table 5.15 provides detailed descriptions about what was referred as a true positive, false positive, true negative, and false negative. Since the creating of US affects the presence of tasks, there was no need to access the feasibility of creating tasks from the generated stories as tasks strongly depends on the existence of USs.

Table 5.15: Classification of sentences that will correctly create user story (Pereira, 2018)

| Sentences/partial phrase | Should create User Story and Task | Shouldn't create User story and Task |
|-----------------------------|---------------------------------------|--|
| Creates a user story | True Positive (TP) Correct results | False positive (FP) Incorrect results |

| | | |
|------------------------------------|---|--|
| Doesn't create a user story | False Negative (FN) Incorrect results | True Negative (TN) Correct results |
|------------------------------------|---|--|

Where:

- TP = The sentence or partial phrase processed by the SAPMT that generates a US, task and which really should create these artefacts.
- FP = The sentence or partial phrase that after processed by the SAPMT generate an Agile US and task but shouldn't do it.
- TN = The sentence or partial processed by the US that doesn't generates Agile US and task, and this is the expected behaviour.
- FN = When the sentence or partial phrase are processed by the SAPMT generates a story, but it shouldn't.

Then, above metrics are utilised to formulate the accuracy equation 5.1 to product

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad 5:1$$

Another imperative metrics that were computed are the Precision (P) and Recall (R) which attributed in the formation of F measure in equation 5:5. The precision rate is the percentage of selected items that are correct, and it's computed by equation 5:2.

$$Precision (P) = \frac{TP}{TP + FP} \quad 5:2$$

The Recall is the percentage of correct items that are selected, and it's computed by Equation 5:3

$$Recall (R) = \frac{TP}{TP + FN} \quad 5:3$$

$$F = \frac{1}{\alpha P + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \quad 5:4$$

Where:

$$\beta = 1$$

$$\alpha = 0.5$$

These constants (β and α) help to reduce Equation 5:4 to Equation 5:5

$$F1 \text{ measure} = \frac{2PR}{P + R} \quad 5:5$$

Table 5.16:Case study results

| Case study | experiment | TP | FP | TN | FN | Accuracy rate (%) | Precision | Recall | F1 Measure |
|--------------------|------------|----|----|----|----|-------------------|-----------|--------|------------|
| IBM payroll system | 1 | 5 | 0 | 1 | 1 | 85.6 | 100 | 83.3 | 90.7 |
| ATM | 2 | 5 | 0 | 1 | 0 | 100 | 100 | 83.3 | 91 |
| Ecommerce | 3 | 5 | 0 | 1 | 0 | 100 | 100 | 83.3 | 91 |
| Agile samurai book | 4 | 3 | 0 | 2 | 2 | 71.4 | 100 | 60 | 75 |

The preceding Table 5.16 presented the results of the experiments performed on four use cases. It illustrates the performance of each case study when using the case study (IBM Payroll system, ATM, Ecommerce and agile samurai(book) respectively. The values of metrics A, P, R and F1 measure were determined. For all metrics used, the higher the value obtained better the results. The results of case study 5.6.3 have an F1 score of 75% reflecting is a huge gap as compared to case other studies. This was affected by the Lack of clarity and concise information attributed to the complication of processing it correctly.

Table 5.14 below shows the general performance of the tool aggregated an average on the tested use cases.

Table 5.17:Avarage performance of the tool

| Metrics | Percentage % |
|----------------|---------------------|
| Accuracy | 89.25 |
| Precision | 100 |
| Recall | 77.25 |
| F1 Measure | 87% |

In comparison with the results obtained by Pereira, (2018), the results of experiment 1 were almost identical our results except the last US generated. The results slightly differ from our tool due to words like salaried and commissioned which were found to possess POS tag with VERB. This means that those words have potential to generate the USs. However, this raises false alarm and cause ambiguous results as indicated on appendix. This resulted into the Accuracy of 85.6, Precision of 100, Recall of 83.3 percent and lastly F1 measure of 90.7%. Moreover, one of the noticeable differences is the results of the

conference resolution. On case study 2, our tool presented the antecedent as products while (Pereira, 2018) was his/ her.

5.9 Discussions

This section discusses the tool's performance, threats to validity, limitations and challenges faced while conducting this study. The proposed tool in this study improves the generation of USs and tasks given the file which comprised of epic USs by automatically generating small manageable stories and tasks. Observing the results in Table 5.16, the tool has efficiently achieved its goals. The research questions, to what extent can NLP be utilised on USs and task generation provided the epics written in NL and how efficient can the automated solution replace the manual solutions were answered. The accuracy and precision degree of the SAPMT has shown an outstanding result that illustrated how NLP can be infused in US decomposition based on the evaluations made. The results obtained shows there is a promising similarity that could be exerted if the epics were decomposed by an expert. Based on these observations, we can argue that our tool can be useful to start-up companies which lacks experienced personnel in requirements decomposition and task assignment.

The best results were obtained when using case study 1,2,3 yielding F1 Score performance of 91%. Furthermore, the tool proposed in section 4.4 which implement the proposed approach has the potential to assist PO and project manager during the sprint planning.

5.9.1 SAMPT performance

On our first exploration, we observed poor performance on NLP while performing conference resolution process concerning the accuracy and processing time. There was a long delay as the algorithm tried to produce the expected results. This caused the laptop to overhead as we waited for the results. To address the delay issue, we enhance the performance of the NLP core server by increasing the RAM parameter on code from 4GB to 6GB and the performance was better.

Upon the arrival of the results from conference resolution process, the results were occasionally not accurate. For example, in experiment 2, the customer was supposed to be replaced with was supposed to be the subject of the US throughout the entire text, however "product" was used instead.

5.9.2 Lack of dataset

There is lack of publicly available data sources that can be used by machine leaning, artificial intelligence and NLP techniques concerning the automated generation of Agile artefacts. The researchers are bound with the rules that compel them to conceal the information from companies they. this complicates the NLP algorithms to analyse and come with the standardize excavation of agile artefacts solution.

5.10 Threads to validity

This section discusses the most relevant threats to validity for our evaluation. Text used to evaluate the tool was generated by the designer. Therefore, there are selective biases in data. This means that the evaluation results obtained in this study do not portray the generic results for other case studies. The validity of this study contests the veracity of constructing the USs and tasks given the file comprised of agile epics written in NL. This study followed aspects of veracity discussed by Runeson and Höst (2009) namely, construct validity, reliability.

The construct validity focuses on the relation between theory and observation. This paradigm concerns whether the measurements studied characterise what the researcher envisioned to examine. Furthermore, the aspects of construction validity were considered during this project. This thesis concluded that not only the accuracy would provide the measurement of the approach's performance but also precision, recall and F1 score.

Reliability concerns the ability to extend this study to which data and analysis are reliant on the researcher's interests. In reference with Runeson and Höst (2009), reliability states that the results of the replicated study should reflect the same results as the original publication. However, there might be complications that could pose threats to reliability. The main reliability threat in this thesis is the technique used to excavate USs and tasks from the SRS consisting of Agile epics. Following section 4.4.2.4, there are linguistic features which are relevant concerning the formulation of USs and tasks but couldn't be extracted. Furthermore, since the SRS consist of Agile epics which are written in NL, these specifications can be ambiguous due to spelling misstates. If the keyword in the specification is misspelt, the precision, recall and F1 metrics are affected. Moreover, if the requirements are written in a different language besides English, the results may differ.

5.11 Conclusion

The results presented in this chapter propose that succinctly written software documentations with the description of the software focused on the user perspective leads to higher accuracy as compared with long texts. The metrics results demonstrated texts with concise structure produces higher precision, recall and F1 Measure. Hungarian is a simple task assignment problem which optimise cost and it is still used in agile environment.

6. CHAPTER SIX.

CONCLUSION AND RECCOMENDATION

6.1 Introduction

This thesis has presented a SAPMT that aids in decision support during PB grooming and sprint planning in Scrum. We proposed the integration of NLP techniques with Hungarian algorithm to attain cost effective tool that maximise the quality of the developed project. Based on the results obtained, this thesis conclude that NLP is adequate technique to automate Agile software artefacts' generation.

6.2 summary of findings

The use of Scrum in ASD has gained popularity among the researchers. 81% of Agile practitioners choose it as the development framework. The generation or construction of agile artefacts is recently performed by NLP. NLP presents viable techniques which automate the text generation processes to minimise total time and development cost. Although NLP is widely used in the generation of Agile artefacts, it produces results that has not reach certain level maturity which can replace human intelligence. This is due to (1) lack of opensource dataset used to train NLP models and structure used to construct requirements. The accuracy of NLP lies in a succinctly written SRS.

Hungarian algorithm is recently used to enhance the performance of task assignment algorithms.

Although it is Classical, its applicable to address task assignments in ASD.

6.3 Conclusion

To conclude the results of the study we want to address the research questions and the aim set in the beginning of the research. The aim was to design and implement an intelligent tool for IoT application requirements specification into stories for the Scrum team.

To address the specified research questions stated in section 1.6, we conducted evaluation experiments regarding How NLP can be used to decompose agile epics to manageable US and tasks. NLP provides rich text analysis techniques which help to generate user stories and tasks. To decompose epics, there are two aspects of US that needs to be extracted: (1) the who and (2) what aspects of the US. POS tagger is NLP technique which is used to extract action verb in a sentence to provide what aspect of US. For complex sentences with more than two action verbs, POS tagger is intertwined with dependency graph to distinguish verbs with dependency tag, root verb. All individual roots verbs are regarded as parts

which form what aspect of US. Discard all other verbs. To extract the who aspect of US, subject mention's results are used in conjunction with conference resolution technique. The output of the two methods provides the answer as pronoun or noun. Activity diagram in section shows a detailed NLP techniques on how epics are decomposed into USs and tasks. Since the algorithm obtained 78% of recall, NLP is the trusted technique to decompose epic stories. Through further experiments and analysis of sentence structures that formulates Agile epics, this thesis infers that the linguistic feature of epics explicitly has two or more action verbs. Action verbs are easily identified by POS tagger. Although the results of automating the decomposition of epics using Nlp provides 90% of accuracy, they have not reached certain level of maturity that can be compared to human experts.

To answer the research question what is an effective approach used to assign tasks to developers in an agile environment?, this thesis has extracted some insights from the literature review. There are two mostly used methods used task assignment models, Crowd sourcing and Hungarian algorithm. Although Hungarian algorithm is a classical task assignment algorithm, its usage is deemed as a powerful technique that is be adapted on ASD.

6.4 Future research and Recommendations

From table 5.14, we can infer that our tool produces 100% Precision. Despite these satisfying results our tool has generated, there is still room for improvement; some of the generated stories possess minor grammatical errors. It would be great to advance the tool with word sense ambiguity. If the words are misspelled, the system should be able to correct them. This would improve the accuracy of the tool.

Our tool's assessment has shown that there a stringent need to enhance the performance of the tool in (1) conference resolution stage in terms of accuracy and time spent to execute the results, add feature for generating USs with passive voice and (3) the complexity of automating text generation was intricate when large text is processed which attributes to complex automation of USs in multiple verbs and lastly introduce software agent with capabilities to capture requirements during requirements elicitation.

In reference to conference resolution, the usage of Stanza with Stanford CoreNLP server could be swapped with NeuralCoref and observe the results. To enhance the tools performance, that indexes that generate USs should be automated thus improving scalability.

Bibliography

- Ahmed, M. *et al.* (2019) 'Estimation of Risks in Scrum Using Agile Software Development', *Springer International Publishing AG* [Preprint].
- Ali, M., Shaikh, Z. and Ali, E. (2016) 'Estimation of Project Size Using User Stories', *International Conference on Recent Advances in Computer Systems (RACS)*, (Racs 2015), pp. 54–60.
- Ansari, P.A. (2017) 'Constructing Test Cases Using Natural Language Processing', *3rd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB17)* [Preprint], (3).
- Azzazi, A. (2017) 'A Framework using NLP to automatically convert User-Stories into Use Cases in Software Projects', *International Journal of Computer Science and Network Security*, 17(5), pp. 71–76.
- Basu, S. *et al.* (2015) 'Task assignment optimization in knowledge-intensive crowdsourcing', *The VLDB Journal*, pp. 467–491
- Beck, K. *et al.* (2001) 'The agile manifesto'.
- Begel, A., Bosch, J. and Storey, M.-A. (2013) 'Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder', *IEEE software*, 30(1), pp. 52–66.
- Bhat, M., Ye, C. and Jacobsen, H.-A. (2014) 'Orchestrating soa using requirement specifications and domain ontologies', in *International Conference on Service-Oriented Computing*, pp. 403–410.
- Biébow, B. and Szulman, S. (1993) 'Acquisition and validation: from text to semantic network', in *International Conference on Knowledge Engineering and Knowledge Management*, pp. 427–446.
- Bolloju, N., Schneider, C. and Sugumaran, V. (2012) 'A knowledge-based system for improving the consistency between object models and use case narratives', *Expert Systems with Applications*, 39(10), pp. 9398–9410.
- Choetkiertikul, M. *et al.* (2016) *A deep learning model for estimating story points*. Available at: www.openhub.net, (Accessed: 9 July 2019).
- Cobb, C.G. (2015) *The project manager's guide to mastering agile: principles and practices for an adaptive approach*. John Wiley & Sons. (Accessed: 14 June 2019).
- Cohn, M. (2004) *User stories applied: For agile software development*. Addison-Wesley Professional.
- Dam, H.K. *et al.* (2018) 'Towards effective AI-powered agile project management', *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE- NIER)*, pp. 41–44.
- Dam, H.K. *et al.* (2019) 'Towards effective AI-powered agile project management', in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 41–44.
- Diebold, P. *et al.* (2015) 'What Do Practitioners Vary in Using Scrum?', in *Springer International Publishing Switzerland*. Springer, Cham, pp. 40–51.
- Dimitrijević, S., Jovanovic, J. and Devedžić, V. (2015) 'A comparative study of software tools for user

story management', *Information and Software Technology*, 57(1), pp. 352–368.

García, T.R., Cancelas, N.G. and Soler-Flores, F. (2014) 'The Artificial Neural Networks to Obtain Port Planning Parameters', *Procedia - Social and Behavioral Sciences*, 162, pp. 168–177.

García, T.R., Cancelas, N.G. and Soler-Flores, F. (2015) 'Setting the port planning parameters in container terminals through bayesian networks', *Promet - Traffic - Traffico*, 27(5), pp. 395–403.

Garfinkel, S. (2005) 'History's worst software bugs', *Wired News*, Nov [Preprint].

Gunes, T. and Aydemir, F.B. (2020) 'Automated Goal Model Extraction from User Stories Using NLP', *Proceedings of the IEEE International Conference on Requirements Engineering*, 2020-Augus, pp. 382–387.

Guo, J., Cheng, J. and Cleland-Huang, J. (2017) 'Semantically Enhanced Software Traceability Using Deep Learning Techniques', *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017*, 39, pp. 3–14.

H. Gillbert Miller (2011) 'The spark of innovation begins with collaboration', *Inside the digital Ecosystem*, 11(1), pp. 13–19.

Hayes-Roth, B. (1985) 'A blackboard architecture for control', *Artificial intelligence*, 26(3), pp. 251–321.

Ho, C. and Vaughan, J.W. (2011) 'Online Task Assignment in Crowdsourcing Markets', pp. 45–51.

Karger, D.R. (2011) 'Iterative Learning for Reliable Crowdsourcing Systems', pp. 0–9.

Kassab, M. (2015) 'The changing landscape of requirements engineering practices over the past decade', in *2015 IEEE fifth international workshop on empirical requirements engineering (EmpiRE)*, pp. 1–8.

Khabbazian, A. *et al.* (2018) 'A decision support system for goods distribution planning in urban areas', *SoutheastCon 2018*, 57(1), pp. 1–9.

Khabbazian, A. (2018) 'A decision support system for goods distribution planning in urban areas', *SoutheastCon 2018*, pp. 1–9.

Klotins, E., Unterkalmsteiner, M. and Gorschek, T. (2016) *Software Engineering in Start-up companies: an Exploratory Study of 88 Start-ups*, Submitted to *EMSE*. Empirical Software Engineering.

Kniberg, H. (2015) *Scrum and XP from the Trenches*. Available at: <http://tscherning.mono.net/upl/10004/110224ScrumAndXpFromTheTrenches.pdf> (Accessed: 7 July 2019).

Lin, J. *et al.* (2014) 'Using goal net to model user stories in agile software development', *2014 IEEE/ACIS 15th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014 - Proceedings*, pp. 1–6.

Lin, Y. *et al.* (2015) 'Multi-Agent System for intelligent Scrum project management', *Integrated Computer-Aided Engineering*, 22(3), pp. 281–296.

Liu, J.W. *et al.* (2019) 'The role of Sprint planning and feedback in game development projects: Implications for game quality', *Journal of Systems and Software*, 154, pp. 79–91.

Lucassen, G. *et al.* (2015) 'Forging high-quality user stories: towards a discipline for agile requirements', in *2015 IEEE 23rd international requirements engineering conference (RE)*, pp. 126–135.

- Meyer, B. (2014) 'Agile principles', *Agile!*, pp. 49–78.
- Miao, X.I.N. *et al.* (2020) 'Quality-aware Online Task Assignment in Mobile', 16(3).
- MIHALACHE, A. (2017) 'Project Management Tools for Agile Teams', *Informatica Economica*, 21(4/2017), pp. 85–93.
- Mills-tetty, G.A. and Stentz, A. (2007) 'The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs', (July).
- Mohagheghi, P. and Jorgensen, M. (2017) 'What contributes to the success of IT projects? Success factors, challenges and lessons learned from an empirical study of software projects in the norwegian public sector', *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, pp. 371–373.
- Mondal, R.N., Hossain, M.R. and Saha, S.K. (2013) 'An Approach for Solving Traveling Salesman Problem', 3(2), pp. 15–26.
- Müter, L. *et al.* (2019) 'Refinement of User Stories into Backlog Items: Linguistic Structure and Action Verbs', in: Springer, pp. 109–116.
- Nayak, A. and Dutta, K. (2018) 'Impacts of machine learning and artificial intelligence on mankind', *Proceedings of 2017 International Conference on Intelligent Computing and Control, I2C2 2017*, 2018-Janua, pp. 1–3.
- Ninaus, G. *et al.* (2014) 'INTELLIREQ: intelligent techniques for software requirements engineering', in *ECAI 2014*. IOS Press, pp. 1161–1166.
- Pereira, A.C. (2018) *using NLP to generate user stories from software specification in natural language*. UNIVERSIDADE FEDERAL DO PARANÁ.
- Perkusich, M. *et al.* (2017) 'Assisting the continuous improvement of Scrum projects using metrics and Bayesian networks', in *Journal of Software: Evolution and Process*.
- Qi, P. *et al.* (2020) 'Stanza: A Python natural language processing toolkit for many human languages', *arXiv preprint arXiv:2003.07082* [Preprint].
- Raharjana, I.K., Siahaan, D. and Fatichah, C. (2021) 'User Stories and Natural Language Processing : A Systematic Literature Review', *IEEE Access*, 9, pp. 53811–53826.
- Ralph, P., Sedano, T. and Péraire, C. (2019) *The Product Backlog*. Available at: <https://www.researchgate.net/publication/330823863> (Accessed: 14 June 2019).
- Ramirez-noriega, A. *et al.* (2016) 'Using Bayesian Networks to Obtain the Task ' s Parameters for Schedule Planning in Scrum', *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, (1), pp. 167–174.
- Ratner, I.M. and Harvey, J. (2011) 'Vertical slicing: Smaller is better', in *Proceedings - 2011 Agile Conference, Agile 2011*, pp. 240–245.
- Robeer, M. *et al.* (2016) 'Automated Extraction of Conceptual Models from User Stories via NLP', *IEEE International Requirement Engineering Conference* [Preprint], (24).
- Runeson, P. and Höst, M. (2009) 'Guidelines for conducting and reporting case study research in software engineering', *Empirical software engineering*, 14(2), pp. 131–164.

- Salman, A., Ahmad, I. and Al-Madani, S. (2002) 'Particle swarm optimization for task assignment problem', *Microprocessors and Microsystems*, 26(8), pp. 363–371.
- Sharma, S. and Hasteer, N. (2017) 'A comprehensive study on state of Scrum development', *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*, pp. 867–872.
- Shi, Z. *et al.* (2020) 'New Task Oriented Recommendation method Based on Hungarian algorithm in Crowdsourcing Platform', pp. 134–144.
- Silberztein, M. *et al.* (2018) *Natural Language Processing and Information Systems: 23rd International Conference on Applications of Natural Language to Information Systems, NLDB 2018, Paris, France, June 13-15, 2018, Proceedings*. Springer.
- Sukthanker, R. *et al.* (2020) 'Anaphora and coreference resolution: A review', *Information Fusion*, 59, pp. 139–162.
- Taherdoost, H. and Keshavarzsaleh, A. (2015) 'A Theoretical Review on IT Project Success / Failure Factors and Evaluating the Associated Risks', *4th International Conference on Telecommunications and Informatics, Sliema, Malta*, (August), pp. 80–88.
- Taibi, D. *et al.* (2017) 'Comparing Requirements Decomposition Within the Scrum, Scrum with Kanban, XP, and Banana Development Processes', in Baumeister, H., Lichter, H., and Riebisch, M. (eds) *Agile Processes in Software Engineering and Extreme Programming*. Cham: Springer International Publishing, pp. 68–83.
- Tailwind, C.S.S. (2019) 'A utility-first CSS framework for rapidly building custom designs'.
- Tran, L. *et al.* (2018) 'A real-time framework for task assignment in hyperlocal spatial crowdsourcing', *ACM Transactions on Intelligent Systems and Technology*, 9(3), pp. 1–26.
- Tugce, G. and Aydemir, F.B. (2020) 'Automated Goal Model Extraction from User Stories Using NLP', *International Requirements Engineering Conference (RE)*, (28), pp. 382–387.
- Veerappa, V. and Harrison, R. (2013) 'Assessing the maturity of requirements through argumentation: A good enough approach', in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 670–675.
- Verma, R.P. (2013) 'Generation of Test Cases from Software Requirements Using Natural Language Processing', *6th International Conference on Emerging Trends in Engineering and Technology*, (6), pp. 141–148.
- Vinet, L. and Zhedanov, A. (2011) *A 'missing' family of classical orthogonal polynomials*, *Journal of Physics A: Mathematical and Theoretical*. Addison-Wesley Professional.
- Wake, B. (2003) 'INVEST in good stories, and SMART tasks', *Retrieved December, 13*, p. 2011.
- Wang, C. *et al.* (2017) 'How to Reduce Software Development Cost with Personnel Assignment Optimization', *ACM International Conference Proceeding Series*, pp. 270–279.
- Wang, C. *et al.* (2020) 'Automatic Generation of Acceptance Test Cases from Use Case Specifications: an NLP-based Approach', *IEEE Transactions on Software Engineering*, pp. 1–1.
- Wang, H. (1997) 'Intelligent agent-assisted decision support systems: Integration of knowledge discovery,

knowledge analysis, and group decision support', *Expert Systems with Applications*, 12(3), pp. 323–335.

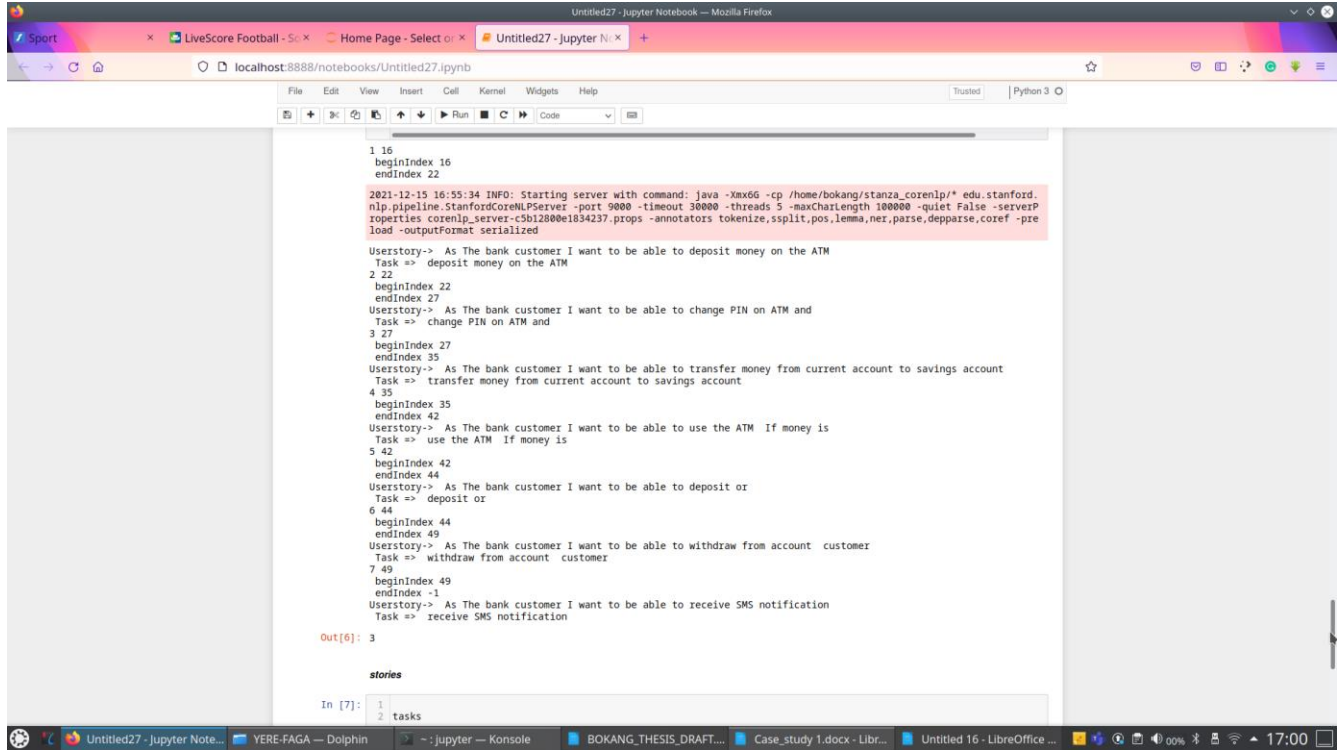
Wu, M. *et al.* (2017) 'A Reinforced Hungarian Algorithm for Task Allocation in Global Software Development'.

Yu, D. (2019) 'Crowdsourcing Software Task Assignment Method for Collaborative Development', *IEEE Access*, 7, pp. 35743–35754.

7. APPENDICES

7.1 APENDIX A

Generating ATM results.



```
1 16
beginIndex 16
endIndex 22
2021-12-15 16:55:34 INFO: Starting server with command: java -Xmx6G -cp /home/bokang/stanza_corenlp/* edu.stanford.nlp.pipeline.StanfordCoreNLP.Server -port 9000 -timeout 30000 -threads 5 -maxCharlength 100000 -quiet False -serverProperties corenlp_server-csb1280@e1834237.props -annotators tokenize,ssplit,pos,lemma,nz,parse,depparse,coref -preload -outputFormat serialized
Userstory-> As The bank customer I want to be able to deposit money on the ATM
Task => deposit money on the ATM
2 22
beginIndex 22
endIndex 27
Userstory-> As The bank customer I want to be able to change PIN on ATM and
Task => change PIN on ATM and
3 27
beginIndex 27
endIndex 35
Userstory-> As The bank customer I want to be able to transfer money from current account to savings account
Task => transfer money from current account to savings account
4 35
beginIndex 35
endIndex 42
Userstory-> As The bank customer I want to be able to use the ATM If money is
Task => use the ATM If money is
5 42
beginIndex 42
endIndex 44
Userstory-> As The bank customer I want to be able to deposit or
Task => deposit or
6 44
beginIndex 44
endIndex 49
Userstory-> As The bank customer I want to be able to withdraw from account customer
Task => withdraw from account customer
7 49
beginIndex 49
endIndex -1
Userstory-> As The bank customer I want to be able to receive SMS notification
Task => receive SMS notification

Out[6]: 3

stories

In [7]: 1
        2 tasks
```

Table 7.1: Saved stories and tasks

```
mysql> alter table requirements modify UserStory varchar(200) NOT NULL Tasks varchar (200) NOT NULL, UNIQUE KEY(UserStory);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Tasks varchar (200) NOT NULL, UNI
UE KEY(UserStory)' at line 1
mysql> alter table requirements modify UserStory varchar(200) NOT NULL ;
Query OK, 0 rows affected (0.06 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table requirements modify UserStory varchar(200) NOT NULL UNIQUE KEY (UserStory);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(UserStory)' at line 1
mysql> alter table requirements modify UserStory varchar(200) NOT NULL UNIQUE KEY (UserStory);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(UserStory)' at line 1
mysql> alter table requirements modify UserStory varchar(200) NOT NULL UNIQUE KEY ('UserStory');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(UserStory)' at line 1
mysql> alter table requirements modify UserStory varchar(200) NOT NULL UNIQUE KEY unique_story(UserStory);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'unique_story(UserStory)' at line 1
mysql> select *from requirements
->
+----+-----+-----+
| id | UserStory                                     | Tasks                                     |
+----+-----+-----+
| 16 | As products I want to be able to view products | view products                           |
| 17 | As products I want to be able to purchase the products online | purchase the products online           |
| 18 | As products I want to be able to add products to the bucket where s he can | add products to the bucket where s he can |
| 19 | As products I want to be able to create account for shipping purposes They must be able to | create account for shipping purposes They must be able to |
| 20 | As products I want to be able to validate expired cards to | validate expired cards to               |
+----+-----+-----+
5 rows in set (0.01 sec)

mysql> select *from requirements;
+----+-----+-----+
| id | UserStory                                     | Tasks                                     |
+----+-----+-----+
| 16 | As products I want to be able to view products | view products                           |
| 17 | As products I want to be able to purchase the products online | purchase the products online           |
| 18 | As products I want to be able to add products to the bucket where s he can | add products to the bucket where s he can |
| 19 | As products I want to be able to create account for shipping purposes They must be able to | create account for shipping purposes They must be able to |
| 20 | As products I want to be able to validate expired cards to | validate expired cards to               |
| 21 | As his I want to be able to track his her performance when | track his her performance when         |
| 22 | As his I want to be able to ride his her bike via the GPS His her performance can be | ride his her bike via the GPS His her performance can be |
| 23 | As his I want to be able to delete any entries once they are | delete any entries once they are       |
| 24 | As his I want to be able to have the ability to | have the ability to                    |
| 25 | As his I want to be able to create a report with all the activities by date range or by type ( | create a report with all the activities by date range or by type ( |
+----+-----+-----+
10 rows in set (0.01 sec)

mysql>
```

```

Userstory-> As products I want to be able to create account for shipping purposes They must be able to
Task => create account for shipping purposes They must be able to
7 50
beginIndex 50
endIndex 65
Userstory-> As products I want to be able to pay with visa cards or cash on delivery ( COD ) The system should
Task => pay with visa cards or cash on delivery ( COD ) The system should
8 65
beginIndex 65
endIndex 69
Userstory-> As products I want to be able to validate expired cards to
Task => validate expired cards to
9 69
beginIndex 69
endIndex -1
Userstory-> As products I want to be able to avoid
Task => avoid

stories

In [28]: 1
         2 tasks
Out[28]: ['view products',
         'purchase the products online ',
         'add products to the bucket where s he can',
         'create account for shipping purposes They must be able to',
         'validate expired cards to']

In [29]: 1 stories
Out[29]: ['As products I want to be able to view products',
         'As products I want to be able to purchase the products online ',
         'As products I want to be able to add products to the bucket where s he can',
         'As products I want to be able to create account for shipping purposes They must be able to',
         'As products I want to be able to validate expired cards to']

In [5]: 1 import pandas as pd
         2
         3 df = pd.DataFrame(
         4     {'UserStory': stories,
         5     'Tasks': tasks,
         6     }, columns=['UserStory', 'Tasks'])
         7

In [ ]: 1
In [ ]: 1 import pyMySQL

```

Untitled27 - Jupyter Notebook — Mozilla Firefox

localhost:8888/notebooks/Untitled27.ipynb

File Edit View Insert Cell Kernel Widgets Help

Python 3

```

Out[6]: 3

stories

In [7]: 1
        2 tasks

Out[7]: ['deposit money on the ATM ',
         'change PIN on ATM and',
         'transfer money from current account to savings account',
         'use the ATM if money is',
         'receive SMS notification']

In [8]: 1 stories

Out[8]: ['As The bank customer I want to be able to deposit money on the ATM ',
         'As The bank customer I want to be able to change PIN on ATM and',
         'As The bank customer I want to be able to transfer money from current account to savings account',
         'As The bank customer I want to be able to use the ATM if money is',
         'As The bank customer I want to be able to receive SMS notification']

In [9]: 1 import pandas as pd
        2
        3 requirements = pd.DataFrame(
        4     {'user Stories': stories,
        5     'Tasks': tasks
        6     })
        7
        8 requirements
        9

Out[9]:

```

| | user Stories | Tasks |
|---|--|---|
| 0 | As The bank customer I want to be able to depo... | deposit money on the ATM |
| 1 | As The bank customer I want to be able to chan... | change PIN on ATM and |
| 2 | As The bank customer I want to be able to tran... | transfer money from current account to savings... |
| 3 | As The bank customer I want to be able to use ... | use the ATM if money is |
| 4 | As The bank customer I want to be able to receo... | receive SMS notification |

In []: 1

In []: 1

Untitled27 - Jupyter Note... YERE-FAGA — Dolphin ~: Jupyter — Konsole BOKANG_THESIS_DRAFT... Case_study 1.docx - Libr... Untitled 16 - LibreOffice ... 00% 17:

Untitled27 - Jupyter Notebook — Mozilla Firefox

localhost:8888/notebooks/Untitled27.ipynb#

File Edit View Insert Cell Kernel Widgets Help

Python 3

```

properties corenlp_server-33538dab93d41b9.props -annotators tokenize,ssplit,pos,lemma,nlp,parse,depparse,coref -pre
load -outputFormat serialized

Userstory-> As his I want to be able to track his her performance when
Task => track his her performance when
1 14
beginIndex 14
endIndex 16
Userstory-> As his I want to be able to run or
Task => run or
2 16
beginIndex 16
endIndex 31
Userstory-> As his I want to be able to ride his her bike via the GPS His her performance can be
Task => ride his her bike via the GPS His her performance can be
3 31
beginIndex 31
endIndex 38
Userstory-> As his I want to be able to save to his her account and
Task => save to his her account and
4 38
beginIndex 38
endIndex 52
Userstory-> As his I want to be able to share with other friends from his her social networks The user cannot
Task => share with other friends from his her social networks The user cannot
5 52
beginIndex 52
endIndex 59
Userstory-> As his I want to be able to delete any entries once they are
Task => delete any entries once they are
6 59
beginIndex 59
endIndex 66
Userstory-> As his I want to be able to save to the account The user
Task => save to the account The user
7 66
beginIndex 66
endIndex 78
Userstory-> As his I want to be able to have the ability to
Task => have the ability to
8 78
beginIndex 78
endIndex 85
Userstory-> As his I want to be able to create a report with all the activities by date range or by type (
Task => create a report with all the activities by date range or by type (
9 85
beginIndex 85
endIndex -1
Userstory-> As his I want to be able to run or biking )
Task => run or biking )

```

Untitled27 - Jupyter Not... Dolphin Konsole Spyder (Python 3.8) Nomfundo Moh - P... LibreOffice Writer 00% 02:40

Untitled27 - Jupyter Notebook - Mozilla Firefox

localhost:8888/notebooks/Untitled27.ipynb#

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
```

```
Task => save to the account The user
7 66
beginIndex 66
endIndex 70
Userstory-> As his I want to be able to have the ability to
Task => have the ability to
8 70
beginIndex 70
endIndex 85
Userstory-> As his I want to be able to create a report with all the activities by date range or by type (
Task => create a report with all the activities by date range or by type (
9 85
beginIndex 85
endIndex -1
Userstory-> As his I want to be able to run or biking )
Task => run or biking )

stories

In [48]: 1
2 tasks

Out[48]: ['track his her performance when',
'ride his her bike via the GPS His her performance can be',
'delete any entries once they are',
'have the ability to',
'create a report with all the activities by date range or by type (']

In [49]: 1 stories

Out[49]: ['As his I want to be able to track his her performance when',
'As his I want to be able to ride his her bike via the GPS His her performance can be',
'As his I want to be able to delete any entries once they are',
'As his I want to be able to have the ability to',
'As his I want to be able to create a report with all the activities by date range or by type (']

In [30]: 1 import pandas as pd
2
3 df = pd.DataFrame(
4     {'UserStory': stories,
5      'Tasks': tasks,
6     }, columns=['UserStory', 'Tasks'])
7

In [ ]: 1

In [ ]: 1 import pymysql
```

02:41

APENDIX B

