Assisted Requirements Selection by Clustering Using Analytical Hierarchical Process

By

Shehzadi Nazeeha Saleem

00000363530

Supervisor

Dr. Wasi Haider Butt

Department of Computer and Software Engineering

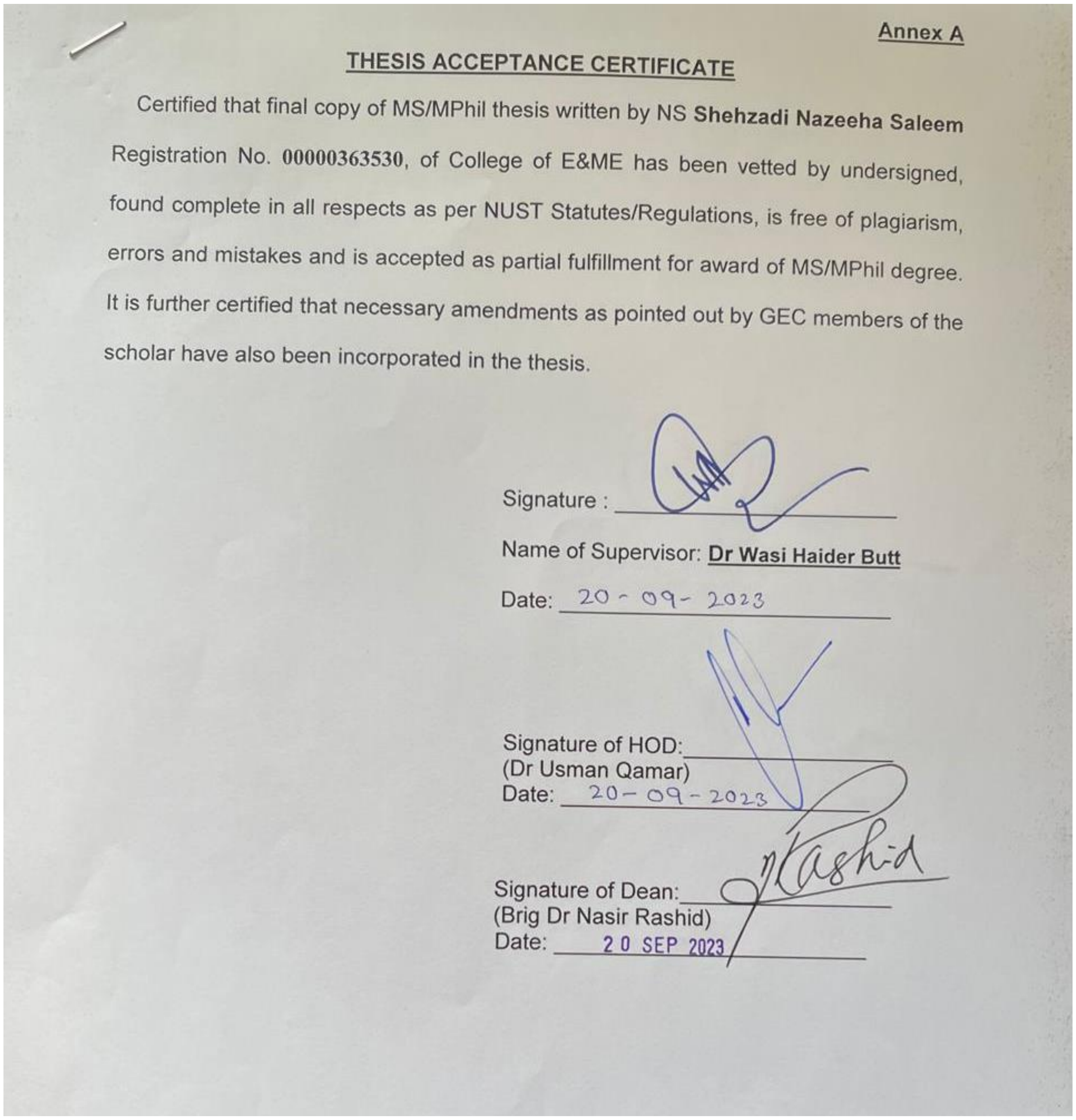
College of Electrical and Mechanical Engineering

National University of Science and Technology (NUST)

Islamabad, Pakistan

September 2023

Thesis Acceptance Certificate



Annex A

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by NS **Shehzadi Nazeeha Saleem** Registration No. **00000363530**, of College of E&ME has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the thesis.

Signature : [illegible]

Name of Supervisor: **Dr Wasi Haider Butt**

Date: 20-09-2023

Signature of HOD: [illegible]
(Dr Usman Qamar)
Date: 20-09-2023

Signature of Dean: [illegible]
(Brig Dr Nasir Rashid)
Date: 20 SEP 2023

Dedicated to my beloved parents and cherished siblings, for your unwavering support and endless love.

Acknowledgements

First and foremost, I offer my deepest gratitude to Allah, the All-Knowing and All-Wise, for granting me the strength, perseverance, and opportunities to complete this thesis. Verily, all success comes from Him, and I pray that He continues to guide me in all my endeavors.

I would like to extend my sincere appreciation to my supervisor, Dr. Wasi haider Butt, for his unwavering support, mentorship, and valuable insights. Your guidance has illuminated my path and enriched my understanding in ways beyond measure.

To my beloved family, whose unwavering love, encouragement, and prayers have been a constant source of strength, I am profoundly thankful. Your belief in me has been the foundation upon which I have built my aspirations.

I also extend my gratitude to my friends who have walked beside me, offering their encouragement and support. Your camaraderie has lightened the burdens and multiplied the joys of this journey.

Abstract

The success of a project depends on the efficient prioritisation of its software requirements. The application of clustering and related data mining techniques for requirements prioritisation within the context of software engineering is still unexplored and frequently overshadowed by established procedures. This study begins a thorough investigation of clustering's untapped potential as a cutting-edge method to enhance requirements prioritisation and enhance project outcomes. To improve the organisation of complicated requirements and determine their relative importance, the study offers the novel idea of combining clustering techniques with the Analytic Hierarchy Process (AHP). Two meticulously constructed quantitative datasets, each containing 20 and 100 software meticulously form the core of this research. Notably, the development of an AHP dataset represents a fresh contribution and serves as a standard by which clustering methods can be unbiasedly assessed. Five main clustering algorithms emerge as the investigation progresses: K-means, Hierarchical, Partition Around Medoids (PAM), Gaussian Mixture Models (GMM), and BIRCH. Each of these methods offers a wide range of analytical techniques for examining the datasets. The Dunn Index, Silhouette Index, and Calinski Harabaz Index are used to statistically measure the quality and cohesion of the created clusters to assess the effectiveness of these approaches. The MoSCoW approach is then used to order the identified criteria into clusters, guaranteeing that crucial requirements are met while allowing for flexibility for less important features. This dual strategy combines strategic prioritisation with quantitative analysis, allowing for an unbiased evaluation of clustering results and simplifying resource allocation based on requirement priority. Overall, this research pioneers the innovative integration of advanced data analysis methodologies into project management and emphasises the viability of clustering techniques for requirement prioritisation in the software domain, with a focus on the ground-breaking combination of AHP and clustering as a transformative approach to prioritise requirements.

Key Words: *Requirements Prioritisation, Software Product Planning, Decision Support, MoSCoW, AHP, Clustering Algorithms, K-Menas, GMM, BIRCH, PAM, Hierarchical, Clusters Evaluation*

Table of Contents

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Software engineering is not only about programming, rather it stands on multiple pillars. It includes all supporting documentation, design tenets, or concepts needed to make these programs work as intended. One of the design principles that enables software that is being considered for development to work as planned is software requirements prioritisation (SRP)[1].

Requirements prioritisation is a branch of requirements engineering which aids in choosing requirements depending on the interests of stakeholders. It is a procedure used in software engineering that deals with giving individual requirements a priority to determine the sequence in which they should be implemented. A requirement engineering decision process is used to decide which features or requirements will be developed in the upcoming release while taking into account technical, resource, risk, and budget constraints [2]. It is a critical phase in the software development process which involves choosing the order in which requirements should be addressed. This procedure helps in controlling the urgency and importance of software requirements while taking stakeholder, cost, quality, resource, and time considerations into account. Definitions for the prioritisation of software needs have been offered by numerous academics. One definition of software requirements prioritisation is the procedure that chooses the sequence in which the needs will be implemented [3]. According to Karlsson and Ryan, it is the process of selecting the best set of requirements from several conflicting and competing expectations gathered from various stakeholders participating in a software development project [4].

Numerous requirements prioritisation strategies have been suggested as a solution to this problem. These

methods seek to shorten the time and expense of software development projects by assisting developers in understanding which requirements are most crucial and urgent. Each technique has drawbacks and both overt and covert assumptions about the project context in which requirement prioritisation takes place [5]. When evaluating a requirement prioritisation approach experimentally, whether for usefulness, utility, application, or effectiveness, these assumptions must be considered.

In the past requirements prioritisation had been done manually but now that technology is evolving like never before, researchers are working on automatic requirements prioritisation. This will shed some load off stakeholders' shoulders. Data driven method logies have been incorporated in requirement engineering and they are generating some amazing results. Still, like any other field of research, there is room for improvement in this area as well.

Clustering algorithms are one of the methods for prioritising software requirements. Clustering is a mechanism using which similar observations, data points, or feature vectors can be grouped together based on shared traits [6]. To group and categorise requirements based on similarity or relatedness; clustering algorithms are used in the prioritisation process. This makes it possible to efficiently prioritise requirements based on the traits of each cluster and to uncover patterns and linkages among them. By grouping requirements into meaningful clusters that can subsequently be prioritised more effectively, clustering algorithms can help in managing the complexity of prioritising many requirements.

In this study we will be using two well-known prioritisation methods namely MoSCoW (Must, Should, Could, Would) and AHP (Analytical Hierarchical Process) with clustering techniques like K-means, Partition Around Medoids, Hierarchical clustering and Gaussian Mixture Models. For the clustering purpose we will be using two of the main prioritisation factors i.e., 'Effort' and 'Satisfaction' which are provided by major stakeholders for each individual requirement. This will help the stakeholders to take better decisions and eventually develop successful systems.

1.1 Motivation

Software Requirement Specification in general and Software Requirement Prioritisation in particular plays the pivotal role in the success or failure of a project. As per the Standish Group, every year almost 80% of software projects fail to meet their definitions of success based on time, cost, and scope criteria [7]. As requirements are frequently published and infrequently updated, it is mostly caused by shifting requirements. This argues that software initiatives fail because they are unable to successfully adapt to changing requirements or make room for new ones. This emphasises how crucial next release management is, as well as the necessity of making informed decisions regarding the functionality of a software product's release. A well-selected release will minimize problems with shifting requirements in future releases.

1.2 Problem Statement

The accurate and timely prioritization of requirements plays a pivotal role in the success of software projects. The main goal of this study is to determine whether combining clustering techniques with the Analytic Hierarchy Process (AHP) can lead to better classification of software requirements, which in turn will affect and improve the overall trajectory of project outcomes.

1.3 Aims and Objectives

The following are the study's primary goals:

- To assess the efficacy of integrating the Analytic Hierarchy Process (AHP) with clustering techniques in enhancing the evaluation and prioritization of software requirements.
- To explore the effectiveness of clustering techniques in enhancing the prioritization of software requirements.
- To partially automate software requirements prioritization activity.

1.4 Research Questions

This thesis will try to find out the answer to the following questions.

- **RQ1:** Is a semi-automated approach to SRP processes possible with the incorporation of clustering techniques?
- **RQ2:** Does the fusion of AHP and clustering generate better results?

1.5 Structure of Thesis

The structure of this work is as follows:

Chapter 2 covers the importance of requirements prioritization and discusses types of requirements prioritisation techniques. It further discusses clustering and its different types.

Chapter 4 gives a review of the relevant literature and the important work produced by scholars in recent years for the ranking of needs.

Chapter 5 includes an explanation of the suggested process.

Chapter 6 explains all the experimental findings in great depth and includes all necessary graphs and tables.

Chapter 7 finishes the thesis and outlines the direction this study will take going forward.

CHAPTER 2

BACKGROUND

2.1 SOFTWARE REQUIREMENTS PRIORITISATION

The process of building or sustaining software systems in a systematic way is called the Software Development Life Cycle (SDLC). The Software Development Life Cycle, a structured process, enables the production of high-quality, low-cost software as quickly as possible. The goal of SDLC is to create top-notch software that satisfies and exceeds all client requirements and expectations. A thorough plan with stages, or phases, each with its own procedure and results, is developed and specified by the SDLC. Following the SDLC expedites development while boosting production efficiency and lowers project risks and costs.

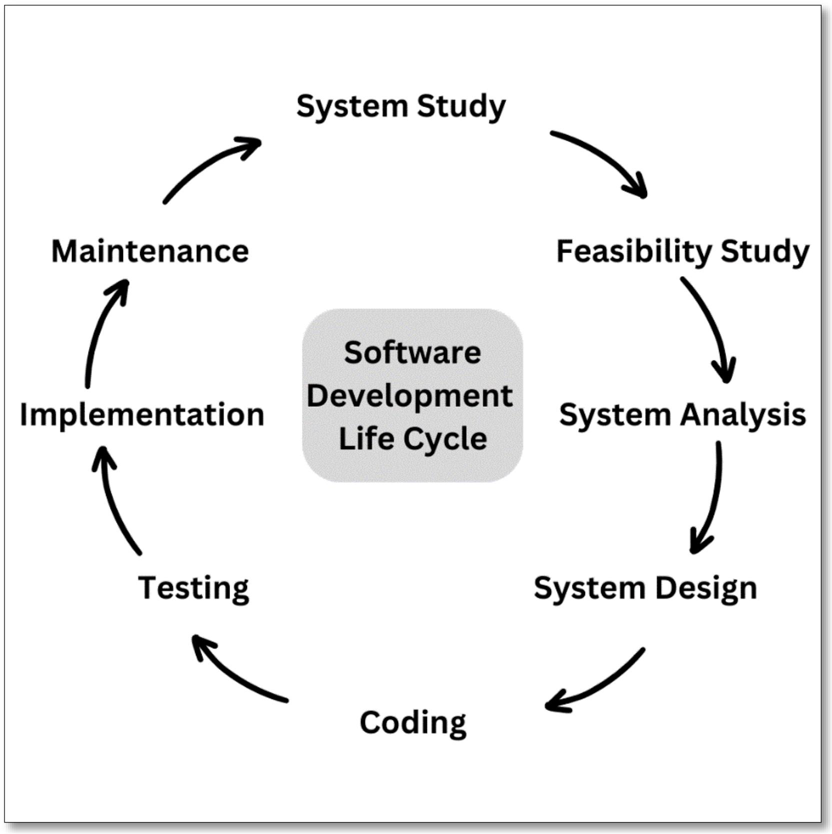


Figure 1 *Software Development Life Cycle*

Any software development process is divided into a number of logical steps, which allows a software development company to efficiently plan out its efforts in order to build a software product with the needed capabilities within a specified time frame and budget. The phases of requirement gathering, business analysis, system design, implementation, and quality assurance testing are completed in all software projects. A generic Software Development Life Cycle with numerous phases was proposed by A. Mishra and D. Dubey [8].

System and feasibility study are crucial stages in Software Development Life Cycle (SDLC) that involves senior team members, stakeholders, and industry experts. Planning quality assurance requirements and identifying project-related risks are both involved. To acquire information about the client's needs, the end user, and the product objectives, a meeting with the client is scheduled. A core understanding of the product is essential before creating a product. The analysis is completed with auditing the feasibility of growth and setting up a signal for further discussion.

The SRS (Software Requirement Specification) document is created, which developers must follow and review for future reference. The next stage is System Analysis, where software requirements are represented and documented, gaining acceptance from project stakeholders. This is accomplished by the creation of the SRS document, which contains all product requirements to be constructed and developed throughout the project life cycle.

System Design is the next phase, bringing together the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two stages, including customer input and requirement gathering.

Coding is the actual development stage, where programming is built and implemented using coding guidelines and programming tools. Testing is conducted against the requirements, including unit, integration, system, and acceptance testing. Once certified, the software is deployed, either as is or with suggested enhancements.

Maintenance begins after deployment, and once the client starts using the developed systems, real issues arise, and requirements need to be solved. This process is known as maintenance, where care is taken for the developed product.

2.1.1 Requirements Prioritisation

Requirements prioritisation is a key component of requirements engineering, which is a stage that is extremely significant in the software development life cycle. The process for gathering requirements include requirements elicitation, requirements analysis, requirements negotiation, requirements documentation, and requirements validation according to the traditional Kotonya and Sommerville requirements engineering approach [9].

Requirements elicitation, often known as requirements collecting or acquisition, is the initial phase of requirements engineering. The system users, clients, and developers—who are referred to as system stakeholders—are asked for their input regarding systems' requirements. Finding the appropriate set of requirements and setting system boundaries are the key goals of this approach. However, by working with the stakeholders, it is possible to specify what requirements can be created. Negotiations with stakeholders are conducted to determine which requirements to execute because it is vital to analyze precise requirements at this stage. The chosen requirements are then recorded and tracked. Then the requirements are checked to see if they are comprehensive and consistent before implementation in the next stage, known as validation.

Testing is also required to determine whether the goals of the implemented requirements have been met. Additionally, because of high competition in the market, businesses concentrate on requirements prioritisation. This enables them to promptly deploy the system's most crucial features to its clients [10].

How well a software program can satisfy user and customer needs determines its quality [11]. Therefore, gathering requirements and determining the correct requirements prior to the release of the appropriate requirements with good functionality is the key to a successful product. Requirements prioritisation is a crucial component of decision-making. This prioritisation aids in separating the key requirements from the less significant ones. The benefits of requirements prioritisation include estimating customer satisfaction, reducing rework and plan instability, assigning relative significance to each demand,

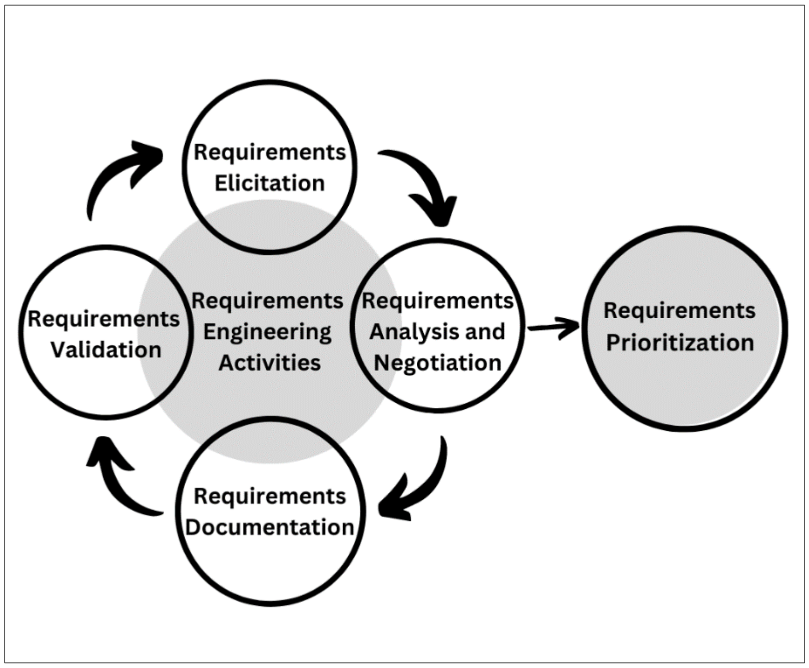

Requirements Elicitation
Requirements Analysis and Negotiation
Requirements Prioritization
Requirements Documentation
Requirements Validation
Requirements Engineering Activities

Figure 2 *Requirements Engineering Activities*

which results in requirements with high value and cheap costs, among other benefits. These activities highlight the significance of setting priorities and selecting the needs that should be considered while developing a product [12]. To produce high-quality software products, managing software requirements is a critical component of the requirements engineering process [13]. On the other hand, requirements prioritisation is acknowledged as a crucial but difficult job in software engineering. Requirements engineering flaws are cited as the main reason software projects go wrong. Later, the prioritised final requirements may serve as the foundation for product and marketing plans and even serve as a driving force for the project plan. The difficulty lies in choosing the "right" requirements from a pool of potential candidates to satisfy the many main interests, technological limitations, and preferences of the important stakeholders while also maximising the product's overall commercial value. The ability to identify requirements flaws such inaccurate, confusing, and poorly evaluated requirements is another advantage of prioritising requirements. They are examined from a new angle, taking requirements into account as part of the review. As a result, requirements start off being general and then get more specific as knowledge about the product increases.

Requirements Prioritisation approaches involve subject-matter specialists, frequent communication with stakeholders, and a close relationship to other criteria. This makes the task of suggesting the best strategy more challenging and increases the need for improvements to these strategies. Karlsson [14]suggests that a prioritising session might include the following three steps in order:

- **The Preparation Stage:** The person organises the requirements at this stage in accordance with the guiding concept of the prioritising strategy that may be employed. Additionally, a team is selected for the session, along with a team leader, and they are given all important information.
- **The Execution Stage:** Based on the knowledge gathered from the previous stage, the decision-makers determine the real prioritising for the criteria in this step.
- **The Presentation Stage:** When the outcomes of the execution are made available to the parties involved.

2.1.2 Requirements Prioritisation Techniques

There are numerous methods for prioritising requirements, some of which are better suited to small numbers of requirements while others are better suited to extremely complex projects with numerous considerations.

Prioritisation techniques let decision-makers evaluate requirements and assign numbers or symbols that accurately reflect their priority [15]. Using these methods, requirements are prioritised using multiple aspects.

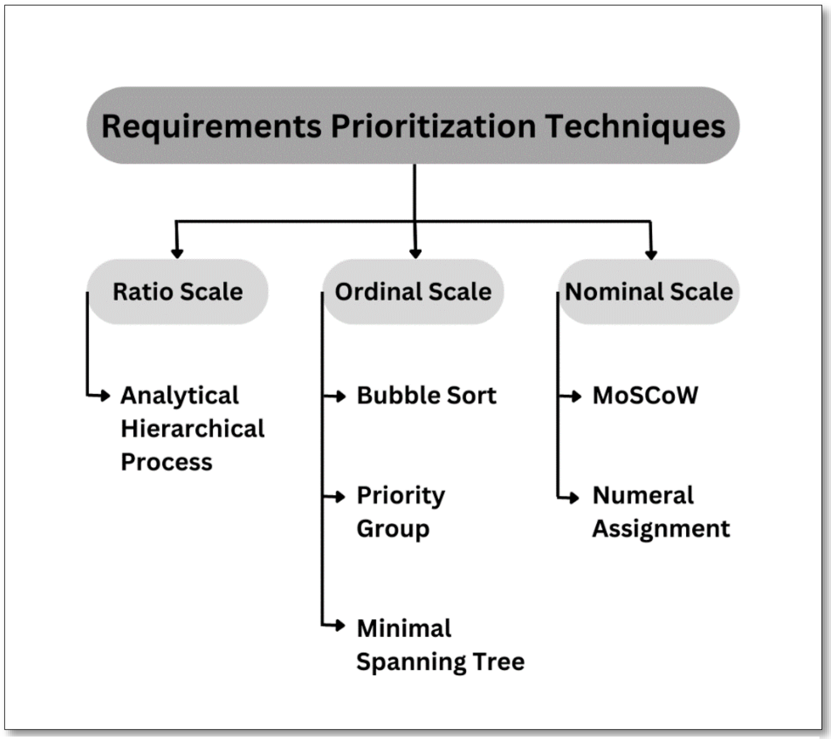


Figure 3 *Requirements Prioritisation Techniques*

Requirement prioritisation techniques are divided into three major groups namely Ratio scale, Ordinal scale, and Nominal scale. These groups have further subcategories as shown in the figure below.

In this study we will be using only two techniques Analytical Hierarchical Process and MoSCoW and hence will be discussing these two as well.

2.1.2.1 Ratio Scale

Techniques for ratio scale prioritisation produce ranked lists of requirements. Results from ratio scale approaches can show how different things are relative to one another.

2.1.2.1.1 Analytical Hierarchical Process

According to Saaty [16] the most common method for requirements prioritisation is called the analytical hierarchy process. Analytic Hierarchy Process (AHP) is a systematic decision-making technique [17]. It was developed for complex decision-making so that the decision-maker could set priorities and get to the best option possible [18]. Thomas L. Saaty created AHP in the 1970s, and since then it has found widespread use in a variety of industries, including business, engineering, project management, and decision analysis. AHP offers a systematic technique to compare and order different criteria or factors, which is especially helpful when there are several to consider.

To arrive at a prioritised ranking of alternatives based on a set of criteria, AHP involves several processes and mathematical calculations. Here is a thorough breakdown of each stage of the AHP procedure:

- **Problem Definition:** The decision problem is defined precisely, together with the choices that need to be assessed. Other criteria or variables that will be taken into consideration while deciding are listed and different ranking factors for a project are considered, as examples of criteria.
- **Pairwise Comparisons:** Alternatives are compared in pairs for each criterion to determine their relative weight. This is accomplished by employing a scale that displays the relative weight or preference given to each criterion. Saaty's 1 to 9 scale is the most often used scale, with 1 denoting equal importance, 3 denoting moderate importance, and 9 denoting exceptionally significant importance. Participants assign values from the scale to each criterion and compare them to each other. For each criterion, a comparison matrix is made using these data.
- **Consistency Check:** Calculations are used to determine whether the pairwise comparison judgements are consistent. Biased outcomes can originate from inconsistent judgements. To make sure the judgements are rational and not in conflict, a consistency ratio is computed. The judgements are regarded as consistent if the consistency ratio is within a reasonable range.
- **Calculation of Weighted Values:** The weighted values for each criterion are calculated based on the pairwise comparison matrices. Each criterion's relative weight in respect to the others is shown by these weights. The pairwise comparison judgements are synthesised to determine the weights.
- **Matrix Multiplication:** A matrix is made to show how each alternative was rated in relation to each criterion. To create a new matrix of weighted scores, this matrix is multiplied by the matrix of criterion weights.

- **Aggregation:** To get a result, weighted scores are added for each possibility. This aggregate rating represents how well or how desirable each choice is in relation to the selected criteria.
- **Sensitivity Analysis:** A sensitivity study is conducted to determine how changes in the assessments will affect the final ranking. This increases the results' reliability.
- **Final Ranking:** Based on their combined scores, alternatives are ordered. The option with the highest overall score is regarded as the best option.

AHP offers a methodical, transparent technique to evaluate difficult choices and rank them according to several factors. It aids in preventing the biases and contradictions that might develop during subjective decision-making processes. AHP requires mathematical computations, however software tools are available to help, making it possible for people without a background in mathematics to use it.

2.1.2.2 Nominal Scale

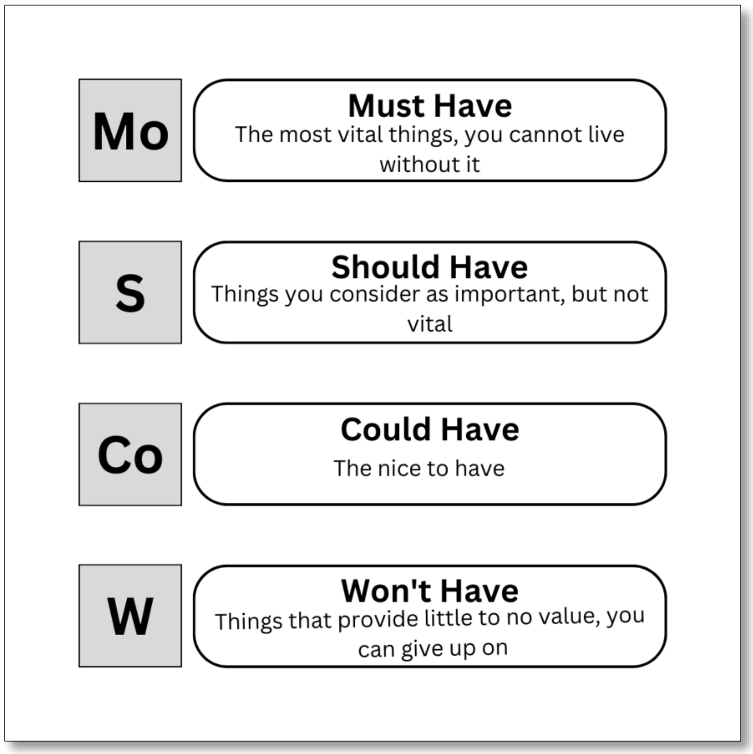
Mechanisms for nominal scale prioritisation produce an array of classes into which objects can be subdivided. Accordingly, requirements are categorised based on their significance. As a result, the priority of all requirements that fall under the same category is the same [19]. Only the Numeral Assignment Technique and the MoSCoW Technique are included in this kind.

2.1.2.2.1 MoSCoW

A common strategy for prioritising requirements in many projects, including software development, product management, and business analysis, is the MoSCoW technique. As a matter of fact, it is one of the easiest techniques [15]. The Dynamic Software Development Method (DSDM) provides the foundation for the MoSCoW method [20]. It offers a methodical manner to classify and rank requirements according to their significance and impact. The abbreviation "MoSCoW" stands for "Must have, Should have, Could have, and Won't have," the initial letter of each category of priorities.

Tudor and Walter [21] gave the following MoSCoW Technique model.

- **Mo - Must Have**: Must-have needs are deemed non-negotiable since they are essential to the project's success. They stand for the fundamental capabilities or features that are necessary for the project to achieve its main goals. The project wouldn't be complete or functional without these prerequisites.


Mo
Must Have
The most vital things, you cannot live without it
S
Should Have
Things you consider as important, but not vital
Co
Could Have
The nice to have
W
Won't Have
Things that provide little to no value, you can give up on

- **S - Should Have:** Although necessary for the project's success, should-have needs are not as crucial as must-have requirements. They significantly enhance the project's value and increase its overall efficacy. Even though the project may still run without them, it would fall short of its full potential and fail to satisfy the expectations of all stakeholders.
- **Co - Could Have:** Could-have needs are preferable because they offer more benefits or features. They stand for improvements or extras that would raise the standard of the project as a whole or enhance the user experience. The decision to include could-have needs depends on the resources and priorities that are available, but they are not essential to the project's fundamental operation.
- **W – Won't Have:** Won't have criteria are purposefully left out of the project's current stage. They are attributes or capabilities that cannot be implemented, are not in line with the project's immediate objectives, or are postponed for later development. Recognising won't-have requirements aids in establishing reasonable expectations and averts scope creep.

Making judgements on which requirements to concentrate on and assign resources to can be done using the MoSCoW technique, which offers a clear framework for doing so. It aids in managing stakeholder expectations, directs development efforts, and makes sure that the most important project components are properly addressed. Teams can streamline their efforts and produce significant results by classifying requirements into must-have, should-have, could-have, and won't-have categories.

2.1.3 Requirements Prioritisation Factors

Prioritising requirements is an essential part of software development since it enables teams to deploy resources wisely and satisfy stakeholders. To make sure that the most crucial features and functionalities are covered, several considerations need to be considered while prioritising requirements. Factors are considered when ranking requirements. Cost, time, importance, risk, and others are some regularly considered factors [22]. Some of the major prioritisation factors are given in the figure below.

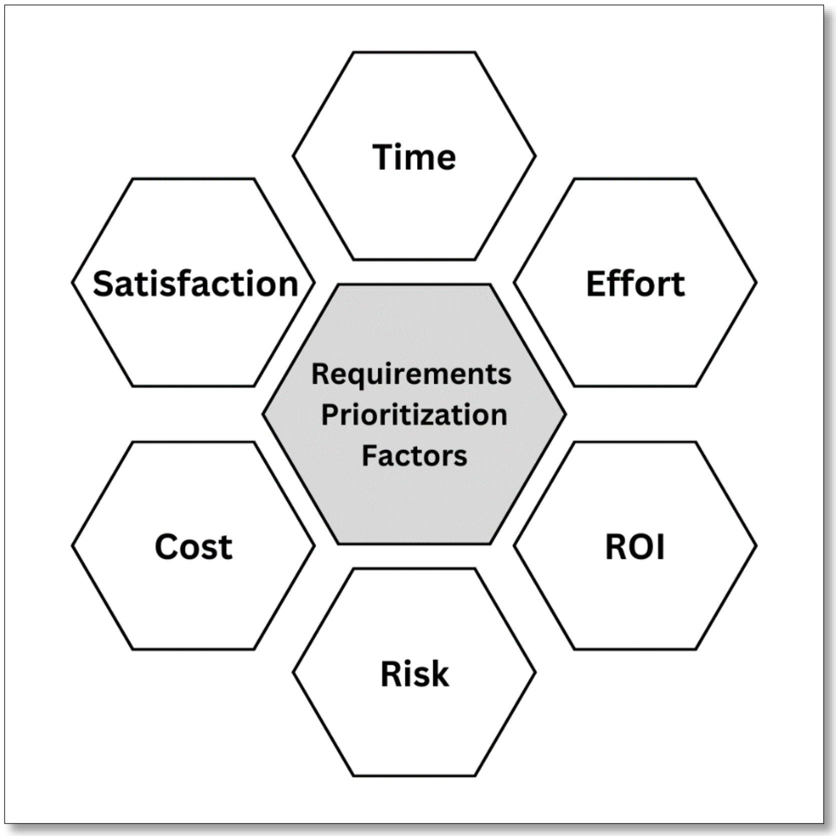


Figure 4 *Requirements Prioritisation Factors*

In this research we will be using two of these factors i.e., Satisfaction and Effort.

2.1.3.1 Satisfaction

The satisfaction factor for requirements prioritisation focuses on evaluating the level of satisfaction and fulfilment that requirements offer to stakeholders. Here is further information:

The following factors must be considered for requirements prioritisation to be successful:

- **Stakeholders Involvement:** It entails determining the degree to which stakeholders, including investors, regulatory agencies, and end users, are involved in the requirement. Their wants and preferences are considered, which results in the prioritisation of attributes that closely match their expectations.
- **User Experience:** The improvement of user experience is essential to stakeholder satisfaction. Requirements that directly enhance the user experience, making the programme more accessible, user-friendly, and in line with user expectations, are given more priority.
- **Market Trends:** The software must be in line with developing market trends and expectations to guarantee stakeholder satisfaction. To take advantage of new opportunities and keep the software relevant and competitive, requirements must be prioritised.
- **Alignment with Business Goals:** Addressing requirements that closely connect with the main business goals is emphasised while setting priorities. Priority is given to features that have a direct bearing on corporate success and strategic objectives.

2.1.3.2 Effort

The resources, time, and labour required to fulfil a particular demand are all considered in the "Effort" factor of requirement prioritisation. It directs choices over which features to prioritise based on their viability within the constraints of the project.

The following are some aspects of effort:

- **Resource Allocation**: Evaluates the infrastructure, knowledge, technologies, and human resources that are available to meet the demand. This considers the team's skill level and any necessary training or additional resources.
- **Time and Schedule**: Involves calculating the amount of time needed to complete the requirement. It prioritises features that can be developed within the allotted time range while taking project deadlines and timelines into consideration.
- **Complexity and Technical Challenges**: Some criteria could be more challenging because of integration problems, technical challenges, or strange technologies. Technical viability within the project's technology stack is considered while setting priorities.

- **Developing and Testing Effort**: Coding, testing, debugging, and quality assurance are all part of implementation. To achieve extensive testing, effort estimation includes unit testing, integration testing, and debugging needs.
- **Documentation and Training**: Creating user guides, documentation, and training materials is a part of effort estimation. It guarantees thorough support for both current and upcoming maintainers.

Effort estimation incorporates assumptions and uncertainty. Realistic estimations are made using an experienced team, historical data, and estimating methodologies (expert judgement, parametric estimation, analogical estimation). Prioritising based on effort guarantees that attention is paid to realistic activities, which results in effective project outputs.

2.2 CLUSTERING ALGORITMS

The need to find knowledge in multidimensional data is growing since massive volumes of data are being continuously collected today. One of the crucial steps in mining or extracting massive information is data miming. It is intended to sift through enormous amounts of data in search of enduring patterns and to verify the findings by comparing the discovered patterns to a fresh selection of data. Clustering algorithms are applicable in this situation because its objective is to identify chunks of related objects within a data collection. Clustering separates data into sets of related things. Each group, or cluster, is made up of things that are dissimilar from those in other groups yet like one another [23]. Clustering of objects is required for a variety of reasons in various sectors of engineering, science, and technology, humanities, medical science, and everyday life [24].

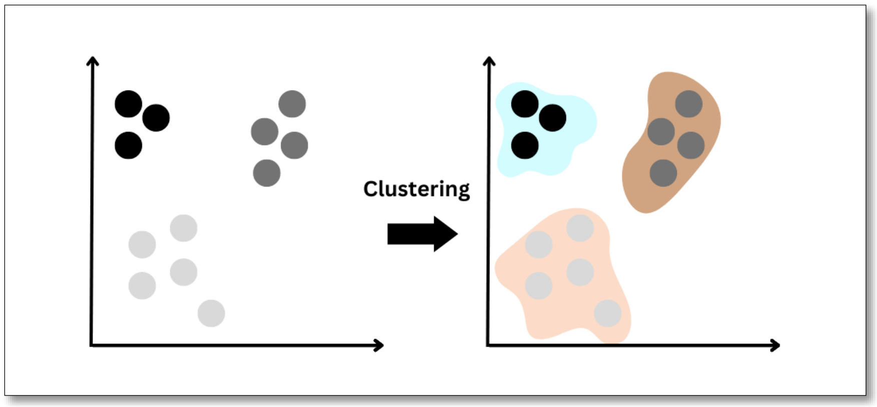


Figure 5 *Formation of Clusters*

Clusters are frequently regarded as the most important unsupervised learning topic, which addresses issues with data collection of unlabeled information [25]. Clustering is the most intriguing area of data mining, which seeks to identify underlying patterns in data and identify some useful subgroups for additional investigation. It is a typical method for statistical data analysis that is applied in a variety of domains, such as bioinformatics, machine learning, data mining, pattern recognition, and image analysis. Thus, the approach of grouping items into groups whose members share some characteristics might also be described as a cluster [26]. A visual representation of cluster formation is shown in figure 5 [27].

2.2.1 Types of Clustering

The primary technique of data mining is clustering. It includes assembling related data points based on predetermined standards. Data mining employs a variety of clustering techniques, each with a unique methodology and traits. Based on the characteristics of the created clusters, partitioning and hierarchical clustering methods can be used to generalise all clustering techniques [28]. A few of the most popular clustering types are as follows:

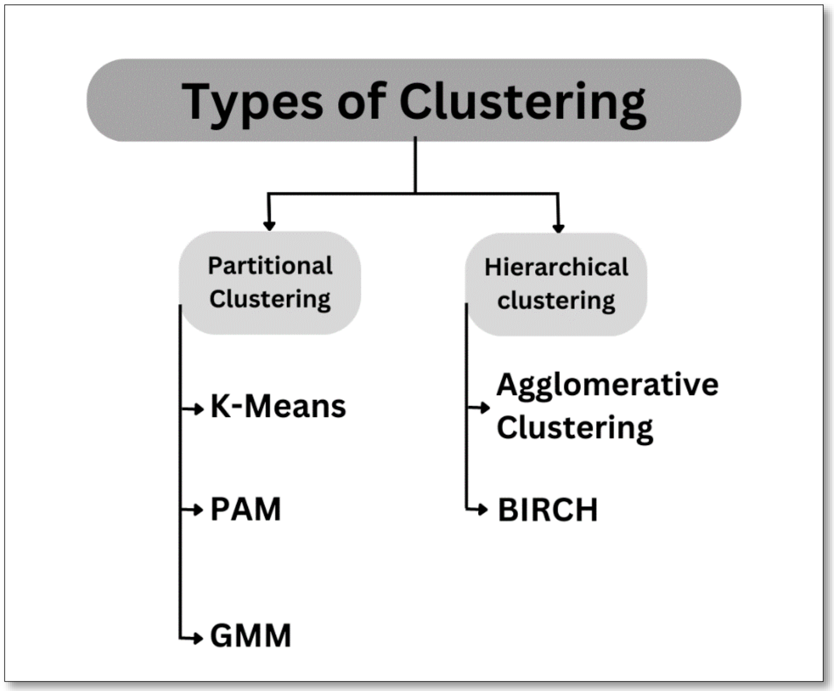


Figure 6 *Types of Clustering*

2.2.1.1 Partitional Clustering

The iterative relocation algorithm, commonly known as partitional clustering, is thought to belong to the most common class of clustering algorithms. These algorithms iteratively move data points across clusters until an ideal partition is reached to minimise a specific clustering criterion. The partition clustering technique divides the data points into k partitions, each of which represents a cluster. The data are partitioned using an objective function. A cluster's objects are "similar," but those of other clusters are "dissimilar." The clusters are created to maximise a distance-based dissimilarity function or another objective partitioning criterion. Partitioning clustering techniques might be useful for applications that need a specific number of clusters. K-means, PAM (Partition around mediods), and Clara are a few of the partitioning clustering algorithms.

2.2.1.1.1 K-Means

K-Means is a well-known and often used clustering algorithm in machine learning and data mining [29]. Prior to the operation, it requires the number of clusters to be defined [30]. It seeks to divide a given dataset into the specified number of clusters (K) according to how similar the data points are to one another to maximise certain clustering criteria. K-Means is an iterative method that produces results by minimising the sum of squared distances between the centroids of each cluster and the data points. It is a popular clustering method that minimises clustering error [31].

The K-Means algorithm is explained in depth below:

- **Initialization:** The initial cluster centroids are initialised at random using K data points. K is the number of clusters we have decided to divide the data into. The cluster centroids are created at random using K data points.
- **Cluster Assignment:** Each of the K centroids in the dataset and the distance between each data point are determined. Two popular distance measurements are the Manhattan distance and the Euclidean distance. Each data point is matched to the centroid of the nearest cluster. K clusters are produced by doing this.
- **Centroid Recalculation:** The new centroid for each cluster is calculated by calculating the mean of all the data points assigned to it. The centroid, which serves as a proxy for the cluster's core, is used to calculate distances in the subsequent iteration.
- **Iteration:** Assignment and Update procedures are iteratively repeated until convergence occurs. Either:
  - The centroids stop fluctuating noticeably, or convergence occurs.
  - There are no more iterations possible.
- **Result:** The centroids serve as the centers of each of the K clusters formed by the data points in the algorithm's final solution.

2.2.1.1.2 Partition Around Medoids

K-Medoids, a K-Means version, and Partitioning Around Medoids (PAM), a clustering algorithm, are closely connected. The PAM method partitions a distance matrix into a predetermined number of clusters [32]. The goal of PAM is to divide a dataset into a predetermined number of clusters by choosing actual data points, known as medoids, as representatives of the clusters. PAM is meant to work with dissimilarity or distance matrices. Like centroids, medoids are chosen from the actual data points, which makes PAM more resistant to noise and outliers.

An in-depth description of the Partitioning Around Medoids (PAM) algorithm is provided below:

- **Initialization:** K is the number of clusters one wants to divide the data into. K can either be chosen through data-adaptive selection or as a given [33]. It is chosen as the initial medoids data points at random from the dataset.
- **Cluster Assignment:** The distance or dissimilarity to each of the K medoids is calculated for each data point in the dataset. Then it is decided which nearby medoid each data point belongs to. This action creates K clusters.
- **Medoid Recalculation:** The total distance or dissimilarity between each data point in a cluster and all other data points in the cluster is calculated for each cluster. Data point for that cluster's new medoid is chosen that has the lowest overall dissimilarity.
- **Iteration:** Iteratively repeat the Assignment and Update procedures until convergence occurs. Either:
    - The medoids stop changing considerably is when convergence begins.
    - There are no more iterations possible.
- **Result:** The method groups the data points into K clusters, with the medoids acting as the nodes of each cluster.

2.2.1.1.3 Gaussian Mixture Models

A probabilistic approach called a Gaussian Mixture approach (GMM) is utilised for density estimation and grouping. Much research has been done on it due to its usefulness and efficiency [34]. It is assumed that the data points are produced by combining several normal distributions with Gaussian distributions. GMM is an effective tool for capturing intricate data distributions and locating underlying patterns.

A thorough explanation of the Gaussian Mixture Model (GMM) is given below:

Model Representation:

A Gaussian mixture model (GMM) depicts the data as the weighted sum of many Gaussian distributions, each of which has a mean and covariance matrix. The mathematical representation of the probability density function (PDF) of a Gaussian mixture model (GMM) is a weighted sum of Gaussian distributions:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

where:

- **x** is a data point.
- **K** is the number of components (Gaussian distributions).
- ***π*k** is the weight of the kth component, satisfying ∑k=1Kπk=1.

N(x|μk,Σk) is the Gaussian distribution with mean μk and covariance matrix Σk.

- **Parameter Estimation:** Finding the best values for the component means (μk), covariance matrices (Σk), and weights (πk) given the observed data is necessary for estimating the parameters of a GMM. The typical approach used for parameter estimation in GMMs is expectation-maximization (EM).
- **Expectation (E-step):** Responsibilities, or posterior probabilities that show the likelihood that each data point belongs to each Gaussian component are calculated.
- **Maximization (M-step):** By increasing the estimated log-likelihood of the data under the existing model, given the responsibilities, parameters are updated.
- **Number of Components:** It's crucial to select the right number of components (K). Too few or too many components may oversimplify the data or overfit the noise, respectively. The ideal number of components can be calculated using a variety of methods, including the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).

2.2.1.2 Hierarchical Clustering

A dataset is split or combined using hierarchical clustering methods into a series of nested divisions. The hierarchy of the nested partitions may be agglomerative (top-down) or divisive (bottom-up). The clustering process used in the agglomerative approach starts with each object being placed in its own cluster, then proceeds on to clustering the closest pairs of clusters until each object is found in a single cluster. Divisive hierarchical clustering, on the other hand, starts with all objects in a single cluster and keeps breaking larger clusters down into smaller ones until all objects are grouped together into unit clusters [35]. Both hierarchical approaches display the dendrogram, a type of naturally occurring cluster representation. Examples of these methods include CURE (Cluster Using REpresentatives), BIRCH (Balance Iterative Reducing and Clustering using Hierarchies), and ROCK.

2.2.1.2.1 Agglomerative Clustering

The process of clustering data points into a hierarchical structure of clusters is called agglomerative hierarchical clustering. Due to the exponential rise of real-world data, hierarchical clustering is crucial for data analytics [36]. Each item initially represents a different cluster in this kind of clustering. After that, by repeatedly merging clusters until all of the data points are a part of a single cluster or until a stopping condition is met, the suitable cluster structure is produced. This process results in a tree-like structure called a dendrogram, which visualises the clustering hierarchy.

Agglomerative algorithms are among the most frequently used algorithms. The technique is too slow for huge data sets in the general case since its complexity is $O(n^3)$. Agglomerative hierarchical clustering is superior to divisive clustering because divisive clustering is poorer when combined with an exhaustive search, which results in the expression (n^2).

The operation of agglomerative hierarchical clustering is described in the following steps:

- **Initialization:** Each data point is considered as its own cluster. A distance matrix by computing the pairwise distances (e.g., Euclidean distance) between all the data points is created.
- **Merge Closest Clusters:** The distance matrix is used to identify the two closest clusters. To do this, several linkage criteria can be used, including:

    - **Single Linkage:** The shortest path between any two sites in the two clusters is used to compute the distance between two clusters.
    - **Complete Linkage:** The maximum distance between any two locations in the two clusters is used to define the distance between two clusters.
    - **Average Linkage:** The average distance between each pair of points in the two clusters is used to compute the distance between two clusters.

  The two neighbouring clusters are combined to make a single cluster. The distance matrix is updated to reflect the separations between the new cluster and the other clusters.

- **Repeat:** The process is repeated until all data points are in a single cluster or until the necessary number of clusters is reached by repeatedly locating the closest clusters and merging them.
- **Dendrogram:** A dendrogram is created to depict the hierarchy of clusters when they are combined. The dendrogram's horizontal axis shows the data points or clusters, and the vertical axis reflects the separation between clusters. The dendrogram's height of each fusion shows how far apart the clusters were when they joined.
- **Cutting the Dendrogram:** One can trim the dendrogram at a specified height to get a specific number of clusters. Based on the height threshold, this will produce the necessary number of clusters.

2.2.1.2.2 Balanced Iterative Reducing and Clustering Using Hierarchies

In 1996, Tian Zhang, Raghu Ramakrishnan, and Miron Livny presented BIRCH. It is an effective hierarchical clustering algorithm made for grouping huge datasets. BIRCH is a clustering method whose key characteristic is to employ low memory resources for high-quality clustering of large-scale data datasets and to only scan datasets once to reduce I/O overhead [37]. A comparable B + tree structure known as a Clustering Feature Tree (CF Tree) is used by Birch to perform clustering [38].

Clustering Feature Construction and Hierarchical Clustering are the two primary steps of BIRCH's operation. Here is a thorough breakdown of each action:

- **Clustering Feature Construction:** The clustering information is represented by BIRCH using a tree-like data structure known as the CF-tree (Clustering Feature tree). The

Clustering Feature (CF), a collection of summary data, is stored at each node in the CF-tree and corresponds to a cluster or sub-cluster.

Three elements make up the Clustering Feature (CF):

- **N:** The cluster's total number of data points.
- **LS:** The linear summing of the data.
- **SS:** The sum of the squared data points.

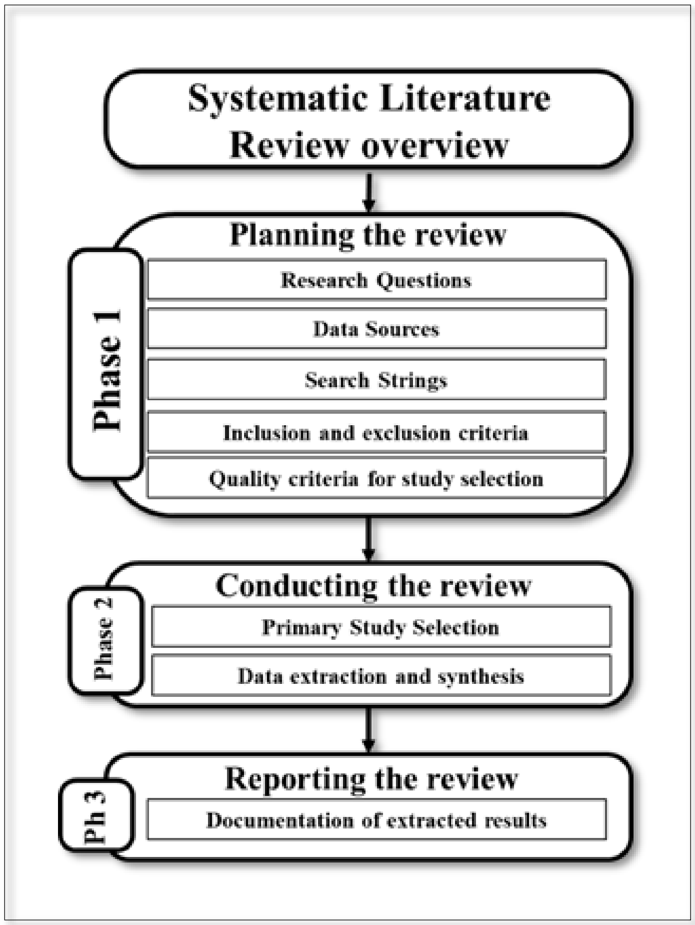
The branching factor is a parameter used by BIRCH to limit the number of sub-clusters a node can have. An algorithm that assures effective memory use inserts new data points into the CF-tree when they come in, updating the Clustering Features in the process. BIRCH may divide a node into two smaller sub-clusters and update the tree structure if a new data point cannot fit in any of the CF-tree's current nodes.

- **Hierarchical Clustering:** A user-defined distance threshold parameter is used by BIRCH to merge sub-clusters after the CF-tree has been built through the iterative insertion of data points. The CF-tree's leaves are combined first, then its root, in a procedure known as "bottom-up" merging. When deciding whether to merge two clusters, BIRCH applies a distance criterion. Utilising their Clustering Features (N, LS, and SS), two clusters are separated from one another. Until the entire dataset is clustered into one cluster or until a predetermined number of clusters are produced, the merging process is carried out.

CHAPTER 3

SYSTEMATIC LITERATURE REVIEW

Systematic Literature Review is a methodical approach to reviewing the body of knowledge regarding a certain issue. SLR is applied here to collect the information about requirements prioritisation techniques and clustering algorithms. By using the SLR approach, your work is objective, and the literature review's conclusions may be trusted. Many academics in the fields of computer and software engineering followed the SLR technique that Kitchenham and Charters [39] proposed and made substantial contributions utilising this approach. The planning, conducting, and reporting of the literature review are the phases of Systematic Literature Review as shown in the figure below.



Systematic Literature Review overview
Planning the review
Phase 1
Research Questions
Data Sources
Search Strings
Inclusion and exclusion criteria
Quality criteria for study selection
Conducting the review
Phase 2
Primary Study Selection
Data extraction and synthesis
Reporting the review
Ph 3
Documentation of extracted results

Figure 7 *Overview of SLR*

3.1 Planning Review

With our objectives in mind, we have articulated research queries that will steer our investigative efforts. The responses we gather will play a pivotal role in sculpting a novel solution that outperforms existing ones. To acquire these responses, we pinpointed pertinent repositories depicted in Figure: Systematic Literature Review Phases. These repositories house relevant literature, accessible for reading or downloading. Following this, we formulated tailored search strings to unearth the most pertinent literature from digital archives. Well-defined inclusion and exclusion criteria were subsequently outlined. The eventual selection of articles will undergo a rigorous quality evaluation, ensuring that our final research comprises articles of the utmost excellence. This methodical approach guarantees the integration of high-caliber information to effectively address our research questions.

3.1.1 Research Questions

Using the state of art, this SLR will try to find out the answer to the following questions.

- **RQ1:** What are the commonly employed techniques for prioritising requirements?
- **RQ2:** Has any paper tried integrating AHP with clustering for prioritizing requirements?
- **RQ3:** What clustering algorithms are currently utilized within the software industry?

3.1.2 Data Sources

Electronic data repositories that are appropriate and closely related have been identified. The repositories are sufficiently related to achieve research goals. Table 1 lists the electronic data repository.

Table 1 *Sources of Data*

Electronic Data Repository	Repository Link
IEEE Explorer	https://ieeexplore.ieee.org/
Springer	https://link.springer.com/
Elsevier	https://www.elsevier.com
Google Scholar	https://scholar.google.com/
Research Gate	https://www.researchgate.net

3.1.3 Search Strings

The research questions guided the construction of search strings. We identified precise keywords and their synonymous counterparts derived from the investigation of requirement prioritisation and clustering literature. All identified terms and their alternatives were established on the foundation of existing research. These terms were then integrated using logical AND and OR operators to generate the ultimate search strings. The compiled search strings have been documented as presented in the table below. This approach ensures that the eventual query is highly effective in unearthing the most pertinent literature in the present field. The utilization of logical operators has imbued the query with a notably optimistic quality.

Table 2 *Search Strings and Alternatives*

Keywords	Alternatives
Requirement Prioritisation Techniques (K1)	"Prioritisation of Requirements" OR "Requirements Ranking"
Comparison of Requirement Prioritisation Techniques (K2)	"Comparing Methods for Requirement Prioritisation" OR "Contrast of Requirement Ranking Approaches"
Analytical Hierarchy Process (K3)	“AHP” OR "Hierarchy Decision Making"
MoSCoW (K4)	"MoSCoW Technique" OR "Must Should Could Won't Technique" OR "Prioritisation using MoSCoW"
Clustering Algorithms (K5)	"Grouping Methods" OR "Cluster Analysis" OR "Contrasting Cluster Methods" OR "Comparative Study of Clustering Approaches"
K-Means (K6)	"K-Means Algorithm" OR "K-Means Clustering" OR "K-Means Method"
PAM (K7)	"PAM Algorithm" OR "Partitioning Around Medoids" OR "Medoid Clustering"
BIRCH (K8)	"BIRCH Algorithm" OR "Balanced Iterative Reducing and Clustering using Hierarchies" OR "BIRCH Clustering"
GMM (K9)	"GMM Algorithm" OR "Gaussian Mixture Model" OR "GMM Clustering"
AHC (K10)	"AHC Algorithm" OR "Agglomerative Hierarchical Clustering" OR "Hierarchical Cluster Analysis"

3.1.4 Inclusion Criteria

Before considering research papers in the current study, a suitable criterion was devised. The following list of bullets contains a definition of all inclusion requirements.

- **IQ1:** The paper gives valuable information for research questions.
- **IQ2:** The study appears in a peer-reviewed publication, conference, or in a workshop.
- **IQ3:** The research paper is not older than 2015.
- **IQ4:** The research paper is written in English language.

- **IQ5:** The paper discusses or compares clustering algorithms.
- **IQ6:** The paper discusses or compares requirements prioritisation techniques.

3.1.5 Exclusion Criteria

The criteria for excluding irrelevant literature from research papers that were extracted is described below.

- **EQ1**: Research paper is not available in PDF format.
- **EQ2**: Research paper does not add valuable information to the research.
- **EQ3**: Research paper does not answer research questions.

3.1.6 Study Quality Evaluation

After the paper extraction procedure, the final literature has been assessed using quality evaluation criteria. As shown in the table below, a tick list has been developed to evaluate selected papers. The five study quality questions on the study quality checklist are what the study quality score is based on. If a study responds to all SQ questions (SQ1–SQ5), the study will receive a quality score of 5, 2.5 for partial responses, and zero for no response. The final decision will be based on the benchmark study quality score and will consider the studies that have contributed the most to the SLR objectives.

Table 3 *Study Quality Score's Criteria*

Study Quality Score	Study Quality Score Criteria
SQS-1	The studies received a "5" for satisfactorily responding to all the checklist's questions.
SQS-2	Studies that only partially addressed the checklist's questions were given a "2.5" score.
SQS-3	Studies that failed to respond to any of the checklist's questions received a score of "0."

The goal of the quality evaluation of our acquired literature is to comment on the quality of the studies that were gathered, which will then be used to articulate the responses to our research questions, "RQs". The following quality evaluation questions regarding RQs are developed and used on each unique study to evaluate its level of quality.

Table 4 *Study Quality Questions' Checklist*

Study Quality Questions	Study Quality Questions Checklist
SQ-1	Does the paper provide a comparison among different prioritisation techniques?
SQ-2	Does the paper explicitly explain a clustering algorithm?
SQ-3	Is the paper able to conclude its results explicitly?
SQ-4	Does the paper provide a better prioritization mechanism?

3.2 Conducting the Review

The review's phases have been further broken down into subphases, as shown below.

3.2.1 Primary Study Selection

A total of twenty-three papers are included in this literature review from diverse sources. Out of these, twelve discuss the requirements prioritisation techniques while the rest talk about clustering algorithms.

Table 5 *Sources of Selected Study*

Electronic Repositories	IEEE Explorer	Springer	Elsevier	Google Scholar	Research Gate
Req. Prioritisation Techniques	3	3	1	4	1
Clustering Algorithms	3	1	3	4	0
Total	**6**	**4**	**4**	**8**	**1**

3.2.2 Data Extraction and Synthesis

The deployed methodology, the accuracy and robustness of those methodologies, the evaluation of models, and other significant data mining attributes specified for inclusion in the articles were all subjected to thorough scrutiny. The integrity of the studies was thoroughly examined and considered in the reference table used to evaluate the quality of the study.

3.3 Reporting the Review

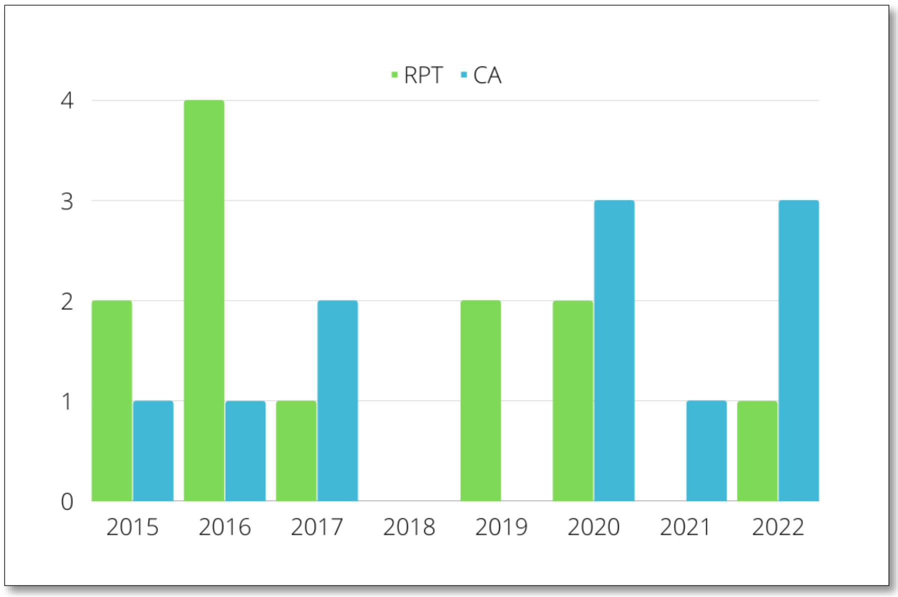
This section provides the reporting of a review with results based on studies that have been extracted.

3.3.1 Quality Attributes

To assess the quality of the research, study quality questions were created. Based on how well the study participated in responding to the research questions, a study's quality score was determined. The research questions should be targeted to those with a higher quality score because they are more likely to be connected to our study objectives.

3.3.2 Temporal Distribution of Selected Primary Studies

All the chosen papers have been categorically split into two groups: those which are related to Requirements prioritisation Techniques (RPT) and those related to Clustering Algorithms (CA). The articles have then been further divided according to the year in which they were published. Out of 23 articles, it has been determined that 12 are about Requirements prioritisation Techniques and 11 are of Clustering Algorithms.



Graph 1 *Yearly Distribution of Selected Studies*

3.3.3 Used Research Methods in Selected Studies

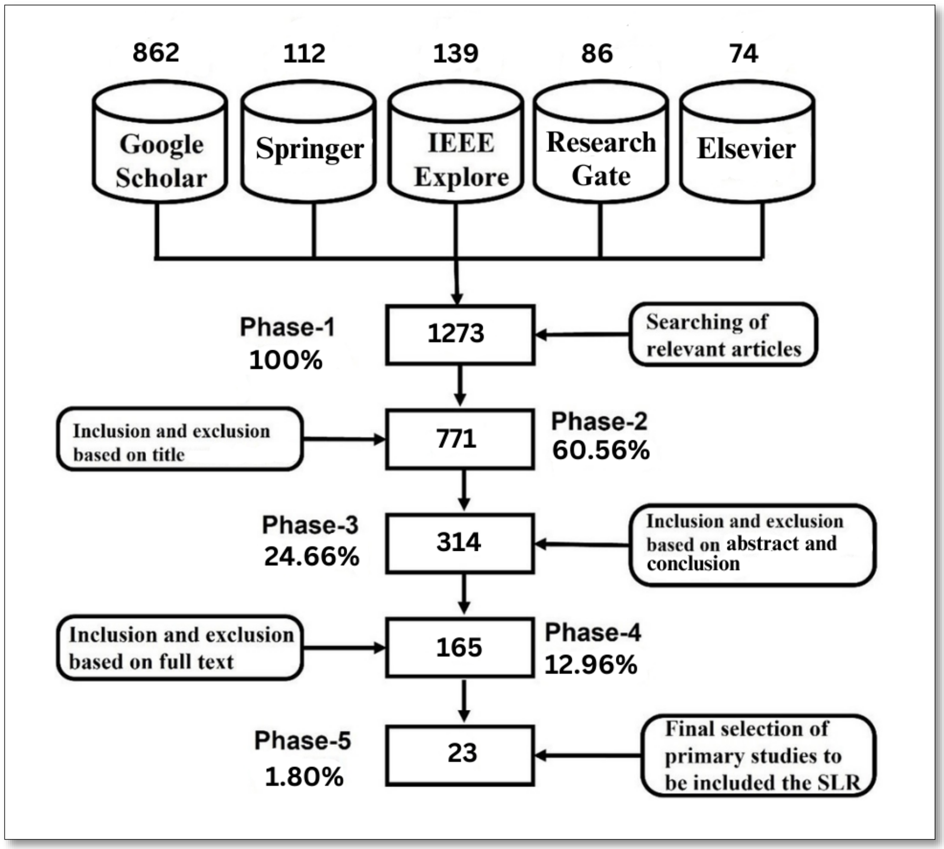

862
112
139
86
74
Google Scholar
Springer
IEEE Explore
Research Gate
Elsevier
Phase-1 100%
1273
Searching of relevant articles
Inclusion and exclusion based on title
771
Phase-2 60.56%
Phase-3 24.66%
314
Inclusion and exclusion based on abstract and conclusion
Inclusion and exclusion based on full text
165
Phase-4 12.96%
Phase-5 1.80%
23
Final selection of primary studies to be included the SLR

Figure 8 *Primary Study Selection*

This study aimed to address questions related to requirements prioritisation techniques and clustering algorithms. As a result, only papers that discussed these two were considered. This criterion was also detailed in the inclusion and exclusion criteria. While selecting the papers, a more focused selection was made to directly address the research questions. Out of the total 23 papers included in this study.

3.4 Results and Discussion of SLR

This section thoroughly presents the outcomes pertaining to the research inquiries. A comprehensive evaluation of all 23 articles was conducted, involving the extraction and thorough examination of their respective findings. The results expounded in this section are entirely grounded in the context of the research questions.

3.4.1 State of Art for Requirements Prioritisation Techniques

Kassab et al [40] proposes the AHP technique to quantitatively rank design choices and software requirement prioritisation process strategies considering how system quality requirements, design principles, and design methods interact. The suggested method is examined using a remote patient monitoring system and the strategy has been shown to be effective in removing discrepancies between stakeholders and the business.

Khan et al [41] declares AHP as the most promising method while comparing demand prioritisation approaches in software development, ranking them based on ease of use, completion time, and outcome accuracy.

Khan et al [42] provides a thorough evaluation of seven strategies for requirements prioritisation: priority groups, bubble sort, binary search tree, spanning tree matrix, analytic hierarchy process (AHP), analytic network process (ANP), and hierarchical AHP. Each prioritisation method is applied to the prioritised Mobilink Franchise system to understand it. The effectiveness of these prioritisation strategies is then assessed in relation to pre-established standards, such as ease of use, necessary completion time, dependability of results, and assessing interdependency of needs. It is concluded that even though ANP requires more time to finish the prioritisation process, it has been proven to be the most reliable and promising strategy among all other prioritising techniques.

Khan et al [43] recommends using RePizer, a framework for ranking software needs based on predetermined criteria like implementation cost, in conjunction with a chosen prioritisation technique. RePizer helps requirements engineers make informed decisions and provides a bird's-eye view of the project. Comparing RePizer with the planning game (PG) and analytical hierarchy process (AHP) revealed better performance in terms of accuracy and usability.

Kouhdaragh et al [44] establishes a cost function to rank the various requirement nodes according to their objectives. The evaluation process uses the numerical results approach. The nodes and the communication technologies are arranged in a priority table according to their weights. The numerical outcomes demonstrate that the suggested strategy enables the optimal communication technologies to be chosen for each type of smart grid node.

Parthasarathy et al [45] provides a framework using AHP to rank requirements according to costs and benefits. The multi-layered strategy is created specifically for enterprise resource planning systems and the unique levels of needs they have, and the framework gives better results.

Ahmad et al [46] suggests a fuzzy MoSCoW method for ranking the importance of software needs. It is applied on the Library Management System (LMS). The research shows that the Fuzzy-MoSCoW method incorporates uncertainty and vagueness in the prioritisation process and allows effective allocation of resources.

Abbas et al [47] introduces a model-based approach for requirements prioritisation using the PageRank algorithm, considering factors like risk, cost, business value, and dependencies. The meta-model and visualization tool is discussed. The modified PageRank algorithm is evaluated and found to be more accurate and efficient than other methods. The results are closer to human prioritisation.

Madzik et al [48] presents a fresh way to evaluate the importance of client requirements called the T4 technique. It combines the AHP, Kano model, and point direct scoring (PDS). The new approach considers both the class of specific requirements and their influence on the improvement ratio, as well as the impact of the fulfilment or non-fulfillment of a certain demand on customer satisfaction or dissatisfaction. Survey-based research verifies the tactic. After analysing the results, the study concludes that T4 is the best method out of the five considered (PDS, AHP, TSM, BITAF, and T4).

Shah Jahan et al [49] proposes a revolutionary method for requirement prioritisation called "MAHP" by incorporating the AHP ideas into the MOSCOW technique, which will decrease the number of pairwise comparisons and therefore the complexity. Using a case study of the Library Management System, MAHP is validated. Results indicate that the suggested strategy, MAHP, is more effective because it has produced 45 comparisons, as opposed to AHP's ability to produce more than 200 pairwise comparisons for only 21 functional requirements.

Yaseen et al [50] prioritises functional requirements using the Analytical Hierarchical Process (AHP) strategy, focusing on spanning trees. This approach minimizes dependencies among developer requirements and reduces waiting time, ensuring timely software project delivery without relying on concurrent developers' demands.

Muhammad et al [51] introduces the Enhanced Analytical Hierarchical Process (E-AHP) to improve scalability and minimize inconsistent results in large software projects. It compares E-

AHP with AHP and ReDCCahp, finding E-AHP suitable for large software projects due to its efficient handling of large requirements numbers.

Table 6 *Literature Review of Req. Prioritisation Techniques*

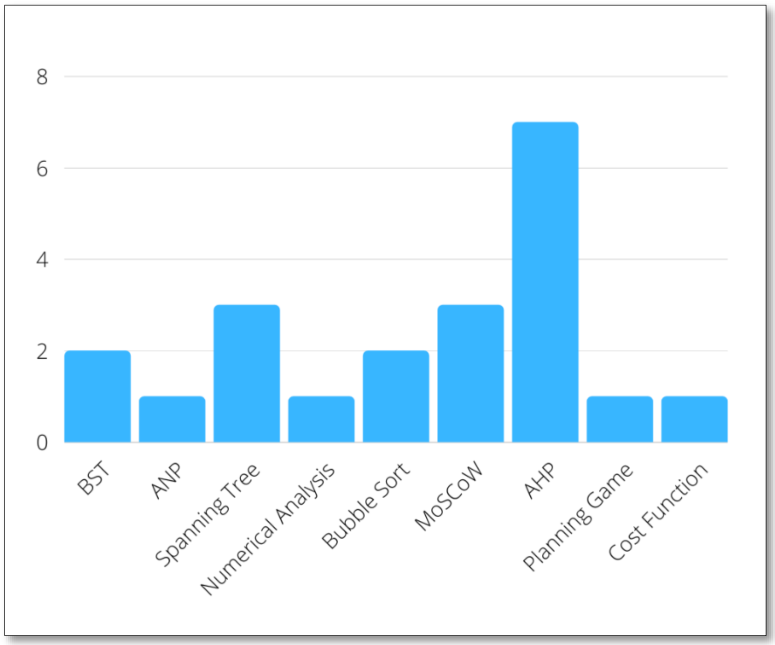
Year Of Pub	Title	Techniques Used	Results	Ref.
2015	Applying analytical hierarchy process to system quality requirements prioritisation	AHP	The AHP technique effectively removes discrepancies between stakeholders and the business.	[40]
2015	Comparison of Requirement Prioritisation Techniques to Find Best Prioritisation Technique	binary search tree, AHP, hierarchy AHP, spanning tree matrix, priority group/Numerical Analysis, bubble sort, MoSoW, simple ranking and Planning Game	AHP is the best requirements prioritisation technique amongst all the requirements prioritisation techniques	[41]
2016	An Evaluation of Requirement Prioritisation Techniques with ANP	ANP, binary search tree, AHP, hierarchy AHP, spanning tree matrix, priority group and bubble sort	ANP is the best technique among the seven techniques, though it consumes time	[42]
2016	Repizer: a framework	Repizer	Comparing RePizer with the	[43]

	for prioritisation of software requirements		planning game (PG) and analytical hierarchy process (AHP) revealed better performance in terms of accuracy and usability	
2016	A Cost Function Based Prioritisation Method for Smart Grid Communication Network	Cost Function	Cost Function optimizes communication technologies for smart grid nodes.	[44]
2016	An approach to estimation of degree of customization for ERP projects using prioritised requirements	Framework using AHP	AHP framework gave better results	[45]
2017	Fuzzy_MoSCoW: A fuzzy based MoSCoW method for the prioritisation of software requirements	Fuzzy MoSCoW	This technique incorporates uncertainty and allocates resource effectively.	[46]
2019	MBRP: Model-Based Requirements Prioritisation Using PageRank Algorithm	MBRP	MBRP can produce outcomes that are more like human prioritisation.	[47]
2019	Determining the Importance of Customer Requirements in QFD – A New Approach based on Kano Model and its Comparison with Other Methods	PDS, AHP, TSM, BITAF, and T4	"T4" can assist with achieving a higher accuracy rate.	[48]
2020	A Novel Approach for Software Requirement Prioritisation	MAHP, a combination of AHP and MoSCoW	MAHP reduces the number of comparisons and hence saves time	[49]

2020	Prioritisation of Software Functional Requirements from Developers Perspective	Spanning Tree and AHP	There was less dependency among requirements hence less waiting time for developers because of spanning tree	[50]
2022	E-AHP: An Enhanced Analytical Hierarchy Process Algorithm for Priotrizing Large Software Requirements Numbers	Enhanced AHP	E-AHP gives better results for large projects	[51]

3.4.1.1 Identified Requirements Prioritisation Techniques

The following graph illustrates the prevalent prioritisation techniques obtained from the state of the art. Most papers have primarily focused on foundational techniques, which were subsequently refined to achieve improved outcomes. In the graph, we have exclusively depicted the fundamental techniques.



Graph 2 *Identified Requirements Prioritisation Techniques*

Table 7 *Identified Requirements Prioritisation Techniques*

Requirements Prioritisation Techniques	Usage Count
Binary Search Tree	2
Analytic Network Process	1
Spanning Tree	3
Numerical Analysis	1
Bubble Sort	2
MoSCoW	3
Analytical Heirarchical Process	7
Planning Game	1
Cost Function	1

3.4.2 State of Art for Clustering Algorithms

Bouguettaya et al [52] proposes centroids as a replacement for raw data points in agglomerative hierarchical clustering techniques. The approach reduces computational cost without compromising clustering performance, and remains consistent regardless of settings like clustering methods, data distributions, and distance measures. A study also evaluates the effectiveness of this approach.

Fouedjio et al [53] provides a method for agglomerative hierarchical clustering that takes the spatial dependency of observations into account. To identify the best clusters and assess the contributions of the variables, it merges existing techniques and makes use of a dissimilarity matrix. In both simulated and real datasets, the approach produces compact, linked, and insightful clusters, yielding satisfying results.

Li et al [54] introduces a PAM (Partitioning Around Medoid) based method for recognizing tool wear state in milling. The LPP (locality preserving projections) method is used to increase clustering accuracy by dimension reduction. The experiments of Ti-6Al-4V alloy showed PAM to perform higher in accuracy and robustness compared to k-means and fuzzy c-means.

Pitolli et al [55] in his research, he compares two different ground truth datasets for malware families, one with labels generated by AVclass and the other based on clusters found by Malheur. It suggests a novel technique for using the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) clustering algorithm to find groups of related samples in an unlabeled dataset. The experimental evaluation shows that BIRCH is effective for identifying malware families, with an accuracy that is on par with or better than that of conventional clustering algorithms. The performance comparison draws attention to BIRCH's quick clustering.

Sinaga et al [56] introduces a new schema and learning framework for the k-means clustering algorithm, using entropy-type penalty terms for competition. The U-k-means algorithm uses points as the initial number of clusters, discarding extra clusters during iterations. It is robust to different data structures and cluster volumes and has been tested on synthetic and real data sets. The results show the superiority of the U-k-means clustering algorithm while comparison with R-EM, C-FS, k-means etc.

Faizan et al [57] explores the use of K-means clustering for analyzing customer purchase behavior in a car manufacturer company. It aims to identify sales-generating products and understand customer behavior, reducing time-consuming manual analysis. The paper presents a car dealership example and demonstrates its usefulness in analyzing customer data and making informed decisions.

Karthikeyan et al [58] analyses delivery fleet driver data sets and compares k-means clustering and hierarchical clustering, which correspond to the centroid and connection models, based on execution time and memory usage. The tests revealed that while agglomerative hierarchical performed better for smaller datasets due to lower memory utilisation, k-means performed better for larger datasets due to lower execution time requirements.

Zhang et al [59] offers a unified learning method that incorporates clustering using Gaussian Mixture Models and imputation. While imputed data is utilised for GMM clustering, the results of GMM clustering are used to fill in the missing data. The compromises made throughout these processes allow imputed data to perform as well as possible for GMM clustering. The optimisation problem is supposed to be solved by a two-step alternative algorithm with established convergence. Numerous tests on eight benchmark datasets from UCI confirm the efficiency of the suggested approach.

Weber et al [60] proposes the Penning trap mass spectrometry (PTMS) phase-imaging ion-cyclotron resonance (PI-ICR) technology, which has increased experiment speed and accuracy. To demonstrate how ions cluster into spots based on cyclotron frequency, PI-ICR generates data sets of individual ion hits. Determining spot centres is challenging because data sets can include several spots, non-spherical spots, or substantial noise. Improving precision and confidence in PI-ICR studies requires a mechanism for designating groups of ions to their appropriate places. The best method for accomplishing this has been shown to be the Gaussian mixture model (GMM) clustering algorithms.

Pezoulas et al [61] presents BGMM-OCE, a technique which uses Gaussian Mixture Models (GMMs) and optimal component estimation to generate synthetic data. It is a computationally efficient and unbiased method for generating synthetic data for large-scale clinical trials. It compares various approaches, highlighting their limitations in efficiency and bias. BGMM-OCE has the lowest coefficient of variation and good fit at a small execution time.

Zhao et al [62] presents an improved K-Means clustering algorithm to tackle local optimal solutions and slow clustering speed issues. It uses the elbow rule for optimum number of clusters and variance and the "triangular inequality principle” to determine cluster centers and minimize unnecessary distance calculations. Experimental results show the improved algorithm outperforms traditional K-Means and Canopy-K Means algorithms in accuracy and speedup ratio.

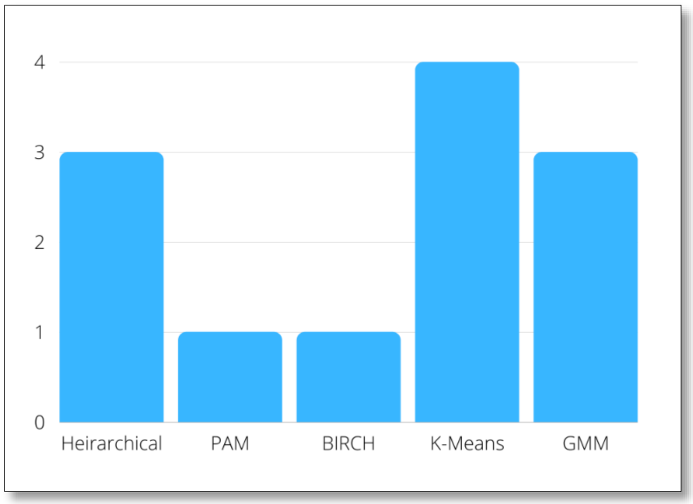
Table 8 *Literature Review of Clustering Algorithms*

Year Of Pub	Title	Techniques Used	Results	Ref.
2015	Efficient agglomerative hierarchical clustering	Efficient agglomerative hierarchical clustering	Experimental results show consistent performance across various settings, proving efficient and reliable.	[52]
2016	A hierarchical clustering method for multivariate geostatistical data	agglomerative hierarchical clustering	Proposed clustering method yields satisfactory results compared to other geostatistical methods.	[53]
2017	Milling tool wear state recognition based on partitioning around medoids (PAM) clustering	PAM	PAM outperforms k-means and fuzzy c-means in Ti-6Al-4V alloy end milling experiments.	[54]
2017	Malware family identification with BIRCH clustering	BIRCH	BIRCH excels in malware family identification with high accuracy and low clustering time.	[55]

2020	Unsupervised K-Means Clustering Algorithm	Unsupervised K-Means	The U-k-means algorithm is robust to data structure and performs better than existing algorithms.	[56]
2020	Applications of Clustering Techniques in Data Mining: A Comparative Study	K-Means, Hierarchical Clustering, DB Scan, OPTICS, Density-Based Clustering, EM Algorithm	The paper emphasises the value of K-means clustering in consumer data analysis and business decision-making	[57]
2020	A Comparative Study on K-Means Clustering and Agglomerative Hierarchical Clustering	K-Means and Agglomerative Hierarchy	K-means faster for large datasets, agglomerative hierarchical better for smaller ones.	[58]
2021	Gaussian Mixture Model Clustering with Incomplete Data	GMM	Experiments validate the effectiveness of the proposed algorithm.	[59]
2022	Gaussian mixture model clustering algorithms for the analysis of high-precision mass measurements	GMM	Results from GMMs were closely congruent with values that had previously been published.	[60]

2022	Bayesian Inference-Based Gaussian Mixture Models with Optimal Components Estimation Towards Large-Scale Synthetic Data Generation for In Silico Clinical Trials	BGMM-OCE	BGMM-OCE outperforms other synthetic data generators in terms of computational efficiency and unbiasedness	[61]
2022	Design and Implementation of an Improved K-Means Clustering Algorithm	Improved K-Means	Enhanced algorithm works better than conventional K-Means.	[62]

3.4.2.1 Identified Clustering Algorithms



Graph 3 Identified Clustering Algorithms

The graph portrays the prevalent clustering algorithms derived from the state of the art. A significant portion of the literature has centered around foundational algorithms, which have been subsequently fine-tuned to yield enhanced results. The graph exclusively showcases the core algorithms.

Table 9 *Identified Clustering Algorithms*

Clustering Algorithms	Usage Count
Agglomerative Hierarchical Clustering	3
Partition Around Medoids	1
BIRCH	1
K-Means	4
GMM	3

3.5 Research Gap

The research gap in this thesis revolves around the limited exploration of the Analytical Hierarchy Process (AHP) as a tool for clustering requirements in the context of planning for the next release of a project. There is a conspicuous lack of research that explore AHP's potential utility in grouping or clustering requirements to speed up the release planning process, even though most of the existing literature focuses on the application of AHP in requirements prioritisation and decision-making. AHP can be used to improve the organisation, categorization, and prioritisation of requirements in the context of release planning, which will ultimately lead to more effective and efficient project management.

3.6 Chapter Summary

This chapter meticulously examined 23 selected papers, with 12 focusing on requirements prioritisation techniques and 11 on clustering algorithms. Notably, a diverse array of approaches, such as Binary Search Tree, Analytic Network Process, Spanning Tree, Numerical Analysis, Bubble Sort, MoSCoW, Analytical Hierarchical Process, Planning Game, and Cost Function, were explored within the state of the art. Remarkably, the widely prevalent choice among researchers was the Analytical Hierarchical Process (AHP), attributed to its consistent delivery of superior results. Additionally, the chapter sheds light on various clustering algorithms, including K-means, Gaussian Mixture Model (GMM), BIRCH, Agglomerative Hierarchical Clustering, and Partition Around Medoids. This synthesis of literature sets the foundation for subsequent research phases, offering insights into the landscape and paving the way for the formulation of an innovative and effective framework.

CHAPTER 4

METHODOLOGY

In the dynamic landscape of software development and project management, prioritising requirements has emerged as a crucial practice to ensure efficient resource allocation, timely delivery, and customer satisfaction. Prioritising requirements is acknowledged as being a challenging task in software product development, nevertheless [63]. Requirement prioritisation involves the systematic process of determining the relative importance and urgency of various features, functionalities, and tasks within a project. By establishing a structured framework for evaluating and ranking requirements, organizations can make informed decisions about what to focus on first, thereby optimizing development efforts and aligning them with strategic goals.

4.1 Requirements Prioritisation Methodology

This thesis presents a method for prioritising requirements for the next release using requirements prioritisation methods. The prioritisation methods consider the effort required for implementing a requirement and the extent to which a particular requirement will be satisfying the stakeholders. After that clustering algorithms are applied to cluster the requirements. Finally, the prioritisation technique is used to extract the group of requirements that will be

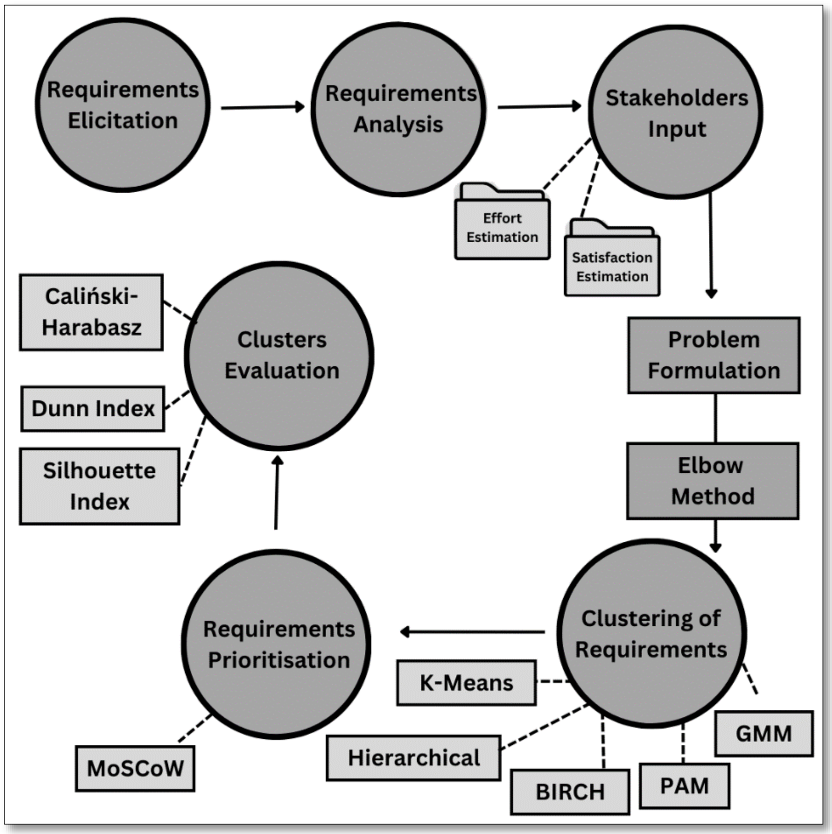


Figure 9 Workflow of Requirements Prioritisation

implemented for the next release. The evaluation of clusters is also done to check the validity of clusters. The figure below shows a bird eye view of the whole process. We will be focusing on the coloured part of the figure.

4.1.1 Requirements Elicitation

Requirement elicitation, a cornerstone of requirement engineering, is a critical step that involves gathering stakeholders' needs and expectations for a software project. This process encompasses discussions, interviews, workshops, and surveys to extract information about desired functionalities, features, and constraints. Its outcomes include comprehensive documented requirements that form the basis for subsequent project phases. By fostering shared understanding and aligning project goals, effective requirement elicitation reduces misunderstandings and enhances the likelihood of delivering a software solution that meets user and business needs. We will initiate the process of the Requirements Elicitation phase, employing systematic approaches like interviews, surveys, and workshops to comprehensively gather insights from stakeholders. Through this method, we aim to establish a clear understanding of project needs and aspirations, ensuring the identification and documentation of all pertinent requirements, thereby laying a solid groundwork for the subsequent requirements prioritisation process.

4.1.2 Requirements Analysis

Requirements analysis holds a pivotal role in software requirement engineering, entailing a comprehensive understanding and detailed scrutiny of the gathered requirements. This phase involves reviewing and clarifying requisites to eliminate ambiguity, addressing inconsistencies, and evaluating feasibility for practical implementation. Activities such as breaking down intricate requirements into manageable components, formulating detailed use cases, and validating the analyzed requirements with stakeholders are integral. Non-functional requirements are refined into quantifiable attributes, while dependencies, constraints, and potential risks are identified. The establishment of traceability links, potential prototyping, and maintaining iterative analysis further enriches the process. This step ensures that the requisites are well-defined, align with stakeholder expectations, and paves the way for subsequent developmental stages. In this thesis, our focus shifts to the following phase of Requirements Analysis. Building on the requirements gathered in the Elicitation phase, Requirements Analysis involves meticulous examination to ensure clarity, completeness, and precision within each requirement. Scrutinizing inconsistencies, conflicts, and gaps, we aim to create a refined and faithful representation of the software's functionalities and attributes. This phase serves as a vital link connecting the initial requirements to subsequent stages like prioritisation and evaluation.

4.1.3 Stakeholders Input

Stakeholders play a crucial part in the collaborative decision-making process by contributing their invaluable insight on both the work needed for implementation and the level of satisfaction expected from a given project or feature. Their ideas on effort consider things like time commitments, resource allocation, and potential problems that can occur throughout development. Stakeholders simultaneously share their expectations for how well the project would correspond with organisational goals, consumer wants, and market demands, contributing their perspective on satisfaction. This two-way input offers a thorough comprehension of the trade-offs involved in project planning and aids in directing decision-makers in choosing initiatives that maximise stakeholder satisfaction while optimising resource utilisation.

4.1.3.1 Problem Formulation

4.1.3.1.1 Quantitative Dataset

Consider a scenario where we have a set of requirements, denoted as **R** = {***R1, R2, …, Rn***}, which represent the new features suggested by various customers for an upcoming software release. These requirements are not of equal importance, as each customer's significance varies. To capture this, each customer ***i*** is assigned a weight ***Wi***, reflecting their importance to the overall software project. This means that some customers' preferences carry more weight than others in determining what should be addressed in the software release. The collection of customer weights is represented as ***W*** = {***W1, W2, …, Wn***}.

Every requirement ***Rj*** within the set R comes with an associated development effort value ***Ej,*** which estimates the cost or resources needed for its implementation. This set of effort values is denoted as ***E*** = {***E1, E2, …, En***}. It's worth noting that a single requirement can be proposed by multiple customers, each assigning it a distinct priority level. This is quantified through a value ***Vij***, indicating the importance of requirement ***Rj*** for customer ***i***. Essentially, higher ***Vij*** values correspond to higher priorities for customer ***i.***

The cumulative value of incorporating a requirement ***Rj*** into the next software release, referred to as its global satisfaction, ***Sj***, is calculated by summing up its importance values across all customers (***Sj = ∑m i=1 Wi · Vij***). In simpler terms, this reflects the combined satisfaction that the inclusion of requirement rj would bring to all customers, considering their individual priorities and weights. The resulting set of requirement satisfactions is represented as ***S*** = {***S1, S2, …, Sn***}. This conceptually summarizes how the varying importance of customers, the priority they assign to requirements, and the collective satisfaction from fulfilling these requirements all interplay to guide the process of selecting what to include in the upcoming software release [64].

4.1.3.1.2 AHP Dataset

Here in this study, will be we will be using the Quantitative data set for the pairwise comparisons of each requirement. This will give us the relative weight of each requirement and hence we will create the AHP data set for our requirements.

To calculate the relative weights of both criteria, Effort and Satisfaction, the eigenvector method is employed after obtaining pairwise comparison judgments. Then, a square matrix is created, known as the comparison matrix, where elements ***C_ij*** represent the importance of criterion Effort (C_i) in relation to criterion Satisfaction (C_j). The matrix is normalized by dividing each column by its sum, resulting in a matrix of normalized values. The average of the normalized values in each row is calculated to derive the priority vector for each level. A consistency check is carried out to ensure consistent pairwise comparisons, utilizing the consistency ratio (CR) to determine if the judgments align coherently. If the CR exceeds a designated threshold, typically set at 0.1, adjustments are made accordingly. Once the consistency check is successfully completed, the priority vectors represent the relative weights of the requirements.

4.1.4 Elbow Method

The elbow method serves as a heuristic approach in data science and machine learning for identifying the optimal number of clusters within a dataset for effective clustering. By employing this method, a range of potential cluster numbers is initially considered. Utilizing a clustering algorithm, often k-means, the sum of squared distances between data points and their corresponding cluster centers is computed. A smaller sum of squared distances indicates more cohesive clusters. The crucial step involves identifying the "elbow point" on a line plot depicting the number of clusters against the sum of squared distances. This point signifies a balance between minimizing the sum of squared distances and preventing excessive fragmentation of data, thereby offering the optimal cluster count for the dataset.

In this study, we apply the elbow method to our requirements dataset. Through iterations, we calculate the within-cluster sum of squares (WCSS) for varying cluster numbers and plot these values. This method aids in finding the optimum number of clusters that best captures underlying patterns and relationships within the requirements. By utilizing the elbow method, we enhance the precision of our requirement clustering process, ensuring that the final cluster configuration aligns effectively with the inherent structure of the requirements dataset.

4.1.5 Clustering of Requirements

Now, we will be advancing to the step of Clustering of Requirements. Recognizing the complexity of managing a multitude of requirements, this phase focuses on organizing and grouping similar requirements into clusters. Through techniques such as similarity analysis or domain categorization, we aim to identify common themes, functionalities, or attributes among the requirements. This process not only enhances the manageability of requirements but also provides a structured approach for analysis. By clustering related requirements together, we can streamline the subsequent prioritisation process and gain a holistic understanding of the software's various aspects. This phase acts as a bridge between requirements analysis and requirements prioritisation, facilitating a more organized and efficient workflow.

We will be employing a comprehensive selection of five distinct clustering algorithms, each designed to partition requirements into coherent groups based on shared characteristics. These algorithms encompass K-Means, which iteratively refines clusters by minimizing distances to cluster centroids; Agglomerative Hierarchical Clustering, which creates a hierarchy of clusters by iteratively merging or agglomerating data points; Partitioning Around Medoids (PAM), which seeks representative points within clusters; Gaussian Mixture Model (GMM), which models data as a mixture of Gaussian distributions; and BIRCH, a hierarchical algorithm that efficiently clusters large datasets by first creating a compact summary of the data. The deployment of these diverse algorithms enhances our ability to extract meaningful patterns and structures from the requirements, providing a comprehensive understanding of their relationships and aiding in informed decision-making during the subsequent prioritisation process.

4.1.6 Clusters Evaluation

The evaluation of clusters is a critical process in assessing the quality and validity of clustering results obtained from data analysis or machine learning algorithms. It involves using various metrics and techniques to determine how well the data points within each cluster are grouped together and separated from other clusters. There are different mechanisms to evaluate clusters. Here we will be using three mechanisms to assess the clusters. These are Dunn Index, Silhouette Index and Caliński-Harabasz Index. These indices are calculated once the clusters are formed. Their final values are used to rate the clusters.

4.1.6.1 Dunn Index

The Dunn Index is a clustering validation metric that quantifies the quality of clusters based on their separation and compactness. By taking into account both the distance between data points inside clusters and the distance between various clusters, it provides a measurement of how clearly defined and distinct the clusters are. A higher Dunn Index value indicates better clustering, as it signifies that clusters are well-separated and compact internally.

The formula for the Dunn Index is:

Dunn Index = Minimum Inter-Cluster Distance / Maximum Intra-Cluster Distance

Where:

- **Minimum Inter-Cluster Distance:** The smallest distance between any two centroids of different clusters.
- **Maximum Intra-Cluster Distance:** The largest distance between any two data points within the same cluster.

4.1.6.2 Silhouette Index

The Silhouette Index is a clustering evaluation measure that assesses the quality of clusters by measuring both how close data points are to their own cluster and how far they are from other clusters. It produces values between -1 and 1. A higher Silhouette Index suggests well-separated clusters.

The formula for Silhouette Index is:

$$S(i) = \{b(i) - a(i)\} / \max\{(a(i), b(i))\}$$

Where:

- ***a(i)*** is the average distance to other points in the same cluster as ***i,***
- ***b(i)*** is the smallest average distance to points in a different cluster.

The overall Silhouette Index is the average of ***S(i)*** values across all data points.

4.1.6.3 Calinski-Harabasz Index

The Caliński-Harabasz Index is a clustering validation metric used to evaluate the quality of clusters in a clustering solution. It calculates the difference between the variances within and between clusters. A higher Caliski-Harabasz Index value denotes clusters that are more well defined and well-separated.

The formula for Caliński-Harabasz Index is:

$$CH=(B/W)\times (N-K) / (K-1)$$

Where:

- **B** is the between-cluster variance (the sum of squared distances between cluster centroids and the overall mean).
- **W** is the within-cluster variance (the sum of squared distances between data points and their respective cluster centroids).
- **N** is the total number of data points.
- **K** is the number of clusters.

4.1.7 Requirements Prioritisation

The Requirements Prioritisation phase in Software Requirement Engineering encompasses the systematic evaluation and ranking of requirements, enabling the optimal allocation of resources by focusing on critical software system aspects. Through thorough assessment of scope, complexity, and stakeholder input, requirements are rigorously evaluated against criteria including effort, business value, feasibility, and risk. This evaluation, informed by stakeholder insights and organized clusters, guides priority assignment using frameworks like MoSCoW, Kano Model, or AHP, ensuring that development efforts target high-impact areas. This process's outcome directs resource allocation, aligning with project goals for maximum value delivery and effective software system development.

In this study, we will be systematically assigning priorities to each requirement using MoSCoW (Must Have, Should Have, Could Have, Won't Have). We will evaluate and rank requirements based on factors ‘Effort required for implementing a requirement’ and ‘Satisfaction level it provides to the stakeholders’. This process ensures that limited resources are allocated to address the most critical and impactful aspects of the software system.

4.1.7.1 MoSCoW

MoSCoW is a project management and requirements analysis technique that categorizes tasks into four groups: Must Have, Should Have, Could Have, and Will Not Have. After clustering the requirements stakeholders will use MoSCoW to prioritise requirements. The project team and stakeholders assess each requirement cluster and assign one of the labels of MoSCoW to it based on its priority. This categorization helps in making informed decisions about resource allocation, project scope, and timelines. By clearly defining what must be delivered versus what can be deferred or omitted, the MoSCoW method aids in managing expectations and focusing efforts on the most critical aspects of the project.

It ensures that the most essential requirements are addressed first while allowing flexibility for less critical elements. The cluster with overall least Effort and most Satisfaction score is likely to fall under ‘Must” label and the cluster with overall most Effort and least Satisfaction score is likely is fall under ‘Won’t” label. This approach helps streamline the development process and allocate resources efficiently to meet project objectives.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Databases

Two cases were chosen to evaluate the effectiveness of our methodology. The first one is a (20-Problem) drawn from [65] and the second data set is a (100-Problem) obtained from [66].

5.1.1 20 Requirements Problem

This dataset has five customers and twenty requirements. Table 10 displays the development effort related to each requirement as well as the level of importance or value given by each customer to each requirement. According to a uniform distribution, the customer weights are provided in the range of 1 to 5. These values can be viewed as language labels such as: without importance (1), less important (2), important (3), very important (4), and extremely important (5). They also correspond to the level of priority of each demand. Each requirement has a corresponding estimated effort score, which ranges from 1 to 10.

Table 10 Raw 20 Req. Problem

	C1	**C2**	**C3**	**C4**	**C5**	**Effort**
R1	4	4	5	4	5	1
R2	2	4	3	5	4	4
R3	1	2	3	2	2	2
R4	2	2	3	3	4	3
R5	5	4	4	3	5	4
R6	5	5	5	4	4	7
R7	2	1	2	2	2	10
R8	4	4	4	4	4	2
R9	4	4	4	2	5	1
R10	4	5	4	3	2	3
R11	2	2	2	5	4	2
R12	3	3	4	2	5	5
R13	4	2	1	3	3	8

R14	2	4	5	2	4	2
R15	4	4	4	4	4	1
R16	4	2	1	3	1	4
R17	4	3	2	5	1	10
R18	1	2	3	4	2	4
R19	3	3	3	3	4	8
R20	2	1	2	2	1	4

Table 11 Customers' Weights for 20 Req. Problem

Customers' Weights	C1	C2	C3	C4	C5
	1	4	2	3	4

5.1.1.1 20 Requirements Problem using Quantitative Approach

To convert the data into two dimensions to apply clustering on it, we considered section 4.1.3.1.1. Here:

R = {R1, R2, ……., R20},

E = {1, 4, 2, …….,4},

W = {1, 4, 2, ……., 4}.

This is how ‘S’ (Satisfaction) was calculated for r1.

S = $\sum (V_{ij} * W_i)$

S= {(4*1) + (4*4) + (5*2) + (4*3) + (5*4)}

S= 62

So, satisfaction for r1 was calculated to be 62 whereas the effort is 1. The rest was also calculated similarly, and this table was generated as a result.

Table 12 Quantitative Data for 20 Req. Problem

ID	Effort	Satisfaction
R1	1	62
R2	4	55
R3	2	29

ID	Effort	Satisfaction
R11	2	45
R12	5	49
R13	8	35

R4	3	41
R5	4	58
R6	7	63
R7	10	24
R8	2	56
R9	1	54
R10	3	49
R14	2	50
R15	1	56
R16	4	27
R17	10	39
R18	4	35
R19	4	46
R20	4	20

5.1.1.2 20 Requirements Problem using AHP

The same data set was used to derive AHP values for Effort and satisfaction and this table was generated as a result.

Table 13 AHP Data for 20 Req. Problem

ID	Effort	Satisfaction
R1	12.7640176	3.24660865
R2	3.19100441	3.65981339
R3	6.38200881	6.9410254
R4	4.25467254	4.90950577
R5	3.19100441	3.4705127
R6	1.82343109	3.19507518
R7	1.27640176	8.38707236
R8	6.38200881	3.59445958
R9	12.7640176	3.72758771
R10	4.25467254	4.10795381
R11	6.38200881	4.47310526
R12	2.55280353	4.10795381
R13	1.5955022	5.75113533
R14	6.38200881	4.02579473
R15	12.7640176	3.59445958
R16	3.19100441	7.45517543
R17	1.27640176	5.1612753
R18	3.19100441	5.75113533
R19	3.19100441	4.37586384
R20	3.19100441	10.0644868

5.1.2 100 Requirements Problem

This data set also consists of 5 customers but this time the number of requirements is 100. This dataset was chosen because of the challenge of choosing requirements from a larger set during the early timeboxes of developing genuine agile software projects. This is the reason now we have 100 requirements instead of just 20. Each requirement has a value for the development effort that ranges from 1 to 20. Here 20 units (4 weeks) represent maximum development effort which is basically the timeframe established by agile methodologies (e.g., Scrum's iteration proposal 2 to 4 weeks (about). The importance of requirements is valued by the customers in the range of 1 to 3. Here the numbers 1-3 are, (1) inessential, (2) desirable or (3) mandatory [67].

Table 14 Raw Data for 100 Req. Problem

	C1	C2	C3	C4	C5	Effort
R1	1	3	1	3	1	16
R2	2	2	1	2	2	19
R3	1	1	1	2	3	16
R4	1	2	2	1	1	7
R5	2	1	1	3	3	19
R6	3	2	1	1	1	15
R7	3	1	1	3	2	8
R8	1	2	3	2	3	10
R9	1	2	2	3	1	6
R10	3	1	2	2	1	18
R11	1	2	3	3	2	15
R12	1	3	3	2	2	12
R13	3	3	3	1	3	16
R14	2	2	1	3	1	20
R15	3	1	3	2	2	9
R16	2	3	1	3	1	4
R17	2	2	2	2	1	16
R18	3	3	2	1	1	2
R19	1	3	3	3	1	9
R20	3	1	3	3	3	3
R21	2	3	2	1	1	2
R22	1	3	1	1	1	10
R23	1	3	2	1	3	4
R24	1	2	3	2	3	2
R25	3	3	2	3	3	7
R26	3	1	3	3	2	15

R27	3	2	3	2	2	8
R28	3	2	1	1	3	20
R29	1	3	3	1	2	9
R30	2	3	3	1	3	11
R31	2	1	3	1	1	5
R32	3	3	2	2	1	1
R33	2	2	1	2	3	17
R34	1	2	2	2	3	6
R35	2	1	2	3	2	2
R36	2	2	1	2	2	16
R37	1	3	1	2	1	8
R38	3	2	3	3	1	12
R39	3	3	1	1	2	18
R40	2	3	2	1	1	5
R41	2	3	1	3	3	6
R42	2	3	3	1	1	14
R43	3	1	1	1	1	15
R44	1	1	3	3	2	20
R45	1	3	3	1	1	14
R46	1	2	3	2	2	9
R47	2	2	3	1	3	16
R48	2	2	1	1	3	6
R49	3	1	3	3	2	6
R50	3	3	2	2	2	6
R51	3	3	3	2	1	6
R52	3	3	1	1	3	2
R53	1	1	2	3	3	17
R54	3	2	3	2	2	18
R55	2	2	2	1	3	1
R56	1	3	3	3	1	3
R57	3	3	2	3	2	14
R58	1	2	1	1	1	16

R59	3	1	2	2	3	18
R60	1	1	3	3	2	7
R61	2	1	1	2	2	10
R62	2	3	1	2	2	7
R63	3	2	2	3	1	16
R64	3	3	3	3	2	19
R65	1	1	3	3	1	17
R66	3	2	1	1	3	15
R67	1	1	3	2	2	11
R68	3	2	3	1	1	8
R69	2	3	3	2	2	20
R70	3	1	1	1	1	1
R71	1	1	3	2	2	5
R72	3	3	1	3	2	8
R73	2	1	3	3	3	3
R74	3	3	1	2	2	15
R75	1	2	1	2	1	4
R76	1	1	2	2	3	20
R77	2	3	3	1	2	10
R78	3	3	3	3	3	20
R79	3	1	1	3	1	3
R80	1	2	2	1	3	20
R81	2	1	1	3	3	10
R82	1	2	2	1	2	16
R83	3	1	3	2	1	19
R84	1	2	2	2	2	3
R85	2	2	3	2	2	12
R86	2	1	1	1	2	16
R87	2	3	2	1	2	15
R88	1	2	2	1	1	1
R89	3	2	3	3	3	6
R90	2	3	3	1	3	7

R91	2	2	3	1	3	15
R92	3	2	3	3	1	18
R93	1	2	2	3	1	4
R94	1	3	1	1	3	7
R95	1	2	1	2	1	2
R96	2	2	2	1	3	7
R97	1	1	3	2	3	8
R98	3	3	3	3	3	7
R99	1	1	2	1	3	7
R100	1	1	3	3	3	3

Table 15 Customers' Weight for 100 Req. Problem

Customers' Weights	C1	C2	C3	C4	C5
	1	5	3	3	1

5.1.2.1 100 Requirements Problem using Quantitative Approach

The Satisfaction for 100 Requirements Problem was also calculated using the same methodology as 6.1.1.1. This is the resulting table.

Table 16 Quantitative Data for 100 Req. Problem

ID	Effort	Satisfaction
R1	16	29
R2	19	23
R3	16	18
R4	7	21
R5	19	22
R6	15	20
R7	8	22
R8	10	29
R9	6	27
R10	18	21
R51	6	34
R52	2	27
R53	17	24
R54	18	30
R55	1	24
R56	3	35
R57	14	35
R58	16	18
R59	18	23
R60	7	26

R11	15	31
R12	12	33
R13	16	33
R14	20	25
R15	9	25
R16	4	30
R17	16	25
R18	2	28
R19	9	35
R20	3	29
R21	2	27
R22	10	23
R23	4	28
R24	2	29
R25	7	36
R26	15	28
R27	8	30
R28	20	22
R29	9	30
R30	11	32
R31	5	20
R32	1	31
R33	17	24
R34	6	26
R35	2	24
R36	16	23
R37	8	26
R38	12	32
R39	18	26
R40	5	27
R41	6	32
R42	14	30

R61	10	18
R62	7	28
R63	16	29
R64	19	38
R65	17	25
R66	15	22
R67	11	23
R68	8	26
R69	20	34
R70	1	15
R71	5	23
R72	8	32
R73	3	28
R74	15	29
R75	4	21
R76	20	21
R77	10	31
R78	20	39
R79	3	21
R80	20	23
R81	10	22
R82	16	22
R83	19	24
R84	3	25
R85	12	29
R86	16	15
R87	15	28
R88	1	21
R89	6	34
R90	7	32
R91	15	27
R92	18	32

R43	15	15		**R93**	4	27
R44	20	26		**R94**	7	25
R45	14	29		**R95**	2	21
R46	9	28		**R96**	7	24
R47	16	27		**R97**	8	24
R48	6	21		**R98**	7	39
R49	6	28		**R99**	7	18
R50	6	32		**R100**	3	27

5.1.2.2 100 Requirements Problem using AHP

Below is the AHP table for the 100-Problem data set.

Table 17 AHP Data for 100 Req. Problem

ID	Effort	Satisfaction		ID	Effort	Satisfaction
R1	0.35245612	0.87906114		**R51**	0.93988298	0.74978744
R2	0.29680515	1.10838143		**R52**	2.81964894	0.94417678
R3	0.35245612	1.41626516		**R53**	0.3317234	1.06219887
R4	0.80561398	1.21394157		**R54**	0.31329433	0.8497591
R5	0.29680515	1.15876241		**R55**	5.63929788	1.06219887
R6	0.37595319	1.27463865		**R56**	1.87976596	0.72836494
R7	0.70491224	1.15876241		**R57**	0.40280699	0.72836494
R8	0.56392979	0.87906114		**R58**	0.35245612	1.41626516
R9	0.93988298	0.94417678		**R59**	0.31329433	1.10838143
R10	0.31329433	1.21394157		**R60**	0.80561398	0.98049127
R11	0.37595319	0.82234751		**R61**	0.56392979	1.41626516
R12	0.46994149	0.77250827		**R62**	0.80561398	0.91045618
R13	0.35245612	0.77250827		**R63**	0.35245612	0.87906114
R14	0.28196489	1.01971092		**R64**	0.29680515	0.67086245
R15	0.62658865	1.01971092		**R65**	0.3317234	1.01971092
R16	1.40982447	0.8497591		**R66**	0.37595319	1.15876241
R17	0.35245612	1.01971092		**R67**	0.51266344	1.10838143
R18	2.81964894	0.91045618		**R68**	0.70491224	0.98049127

R19	0.62658865	0.72836494
R20	1.87976596	0.87906114
R21	2.81964894	0.94417678
R22	0.56392979	1.10838143
R23	1.40982447	0.91045618
R24	2.81964894	0.87906114
R25	0.80561398	0.70813258
R26	0.37595319	0.91045618
R27	0.70491224	0.8497591
R28	0.28196489	1.15876241
R29	0.62658865	0.8497591
R30	0.51266344	0.79664915
R31	1.12785958	1.27463865
R32	5.63929788	0.82234751
R33	0.3317234	1.06219887
R34	0.93988298	0.98049127
R35	2.81964894	1.06219887
R36	0.35245612	1.10838143
R37	0.70491224	0.98049127
R38	0.46994149	0.79664915
R39	0.31329433	0.98049127
R40	1.12785958	0.94417678
R41	0.93988298	0.79664915
R42	0.40280699	0.8497591
R43	0.37595319	1.6995182
R44	0.28196489	0.98049127
R45	0.40280699	0.87906114
R46	0.62658865	0.91045618
R47	0.35245612	0.94417678
R48	0.93988298	1.21394157
R49	0.93988298	0.91045618
R50	0.93988298	0.79664915

R69	0.28196489	0.74978744
R70	5.63929788	1.6995182
R71	1.12785958	1.10838143
R72	0.70491224	0.79664915
R73	1.87976596	0.91045618
R74	0.37595319	0.87906114
R75	1.40982447	1.21394157
R76	0.28196489	1.21394157
R77	0.56392979	0.82234751
R78	0.28196489	0.65366084
R79	1.87976596	1.21394157
R80	0.28196489	1.10838143
R81	0.56392979	1.15876241
R82	0.35245612	1.15876241
R83	0.29680515	1.06219887
R84	1.87976596	1.01971092
R85	0.46994149	0.87906114
R86	0.35245612	1.6995182
R87	0.37595319	0.91045618
R88	5.63929788	1.21394157
R89	0.93988298	0.74978744
R90	0.80561398	0.79664915
R91	0.37595319	0.94417678
R92	0.31329433	0.79664915
R93	1.40982447	0.94417678
R94	0.80561398	1.01971092
R95	2.81964894	1.21394157
R96	0.80561398	1.06219887
R97	0.70491224	1.06219887
R98	0.80561398	0.65366084
R99	0.80561398	1.41626516
R100	1.87976596	0.94417678

5.2 Elbow Method

The elbow method was applied to both 20 and 100 Requirements Problem data sets to find the optimum number of clusters. The graphs below show the optimum number of clusters.

5.2.1 20 Requirements Problem

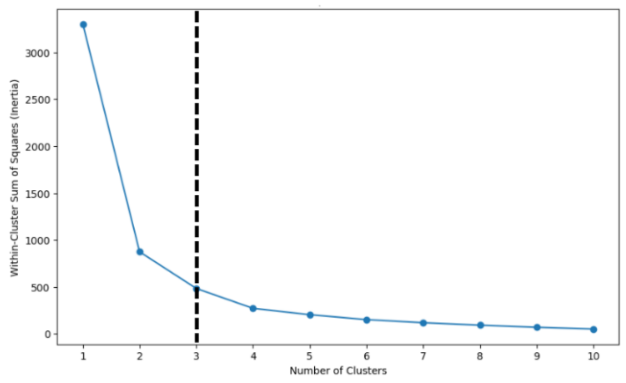


Figure 10 *Optimum Number of Clusters for Quantitative Approach*

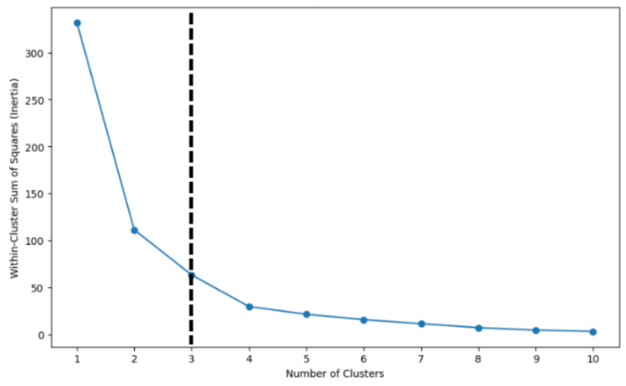


Figure 11 *Optimum Number of Clusters for AHP*

5.2.2 100 Requirements Problem

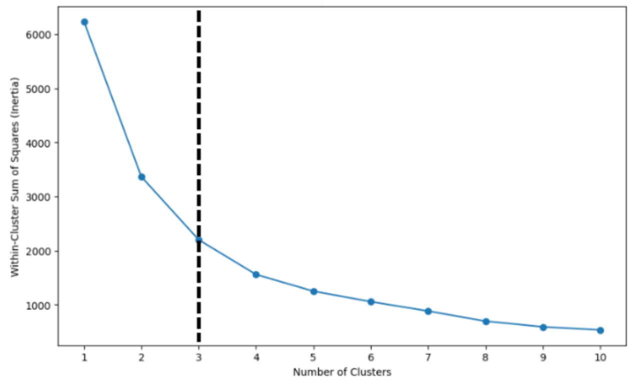


Figure 12 *Optimum Number of Clusters for Quantitative Approach*

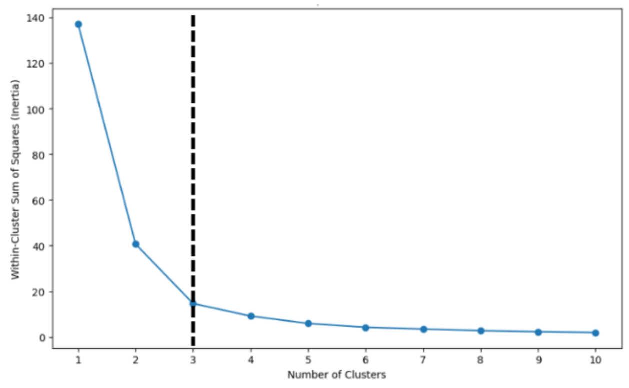


Figure 13 *Optimum Number of Clusters for AHP*

5.3 Clustering

As per the results of elbow method the optimum number of clusters for both 20 and 100 Requirements Problem was 3. Since we are also using MoSCoW along with AHP for requirements prioritisation, we created 3 as well as 4 clusters. The reason for this is MoSCoW has four attributes.

This research utilized five clustering algorithms, including K-Means, Partition Around Medoids, Agglomerative Hierarchical Clustering, Gaussian Mixture Models, and BIRCH, to group data points in quantitative and AHP datasets. The analysis was conducted on two scales of problem instances: 20 requirements and 100 requirements. Three and four clusters were generated for each dataset, acting as cohesive groups of related data points. Below are clusters for AHP datasets.

5.3.1 K-Means

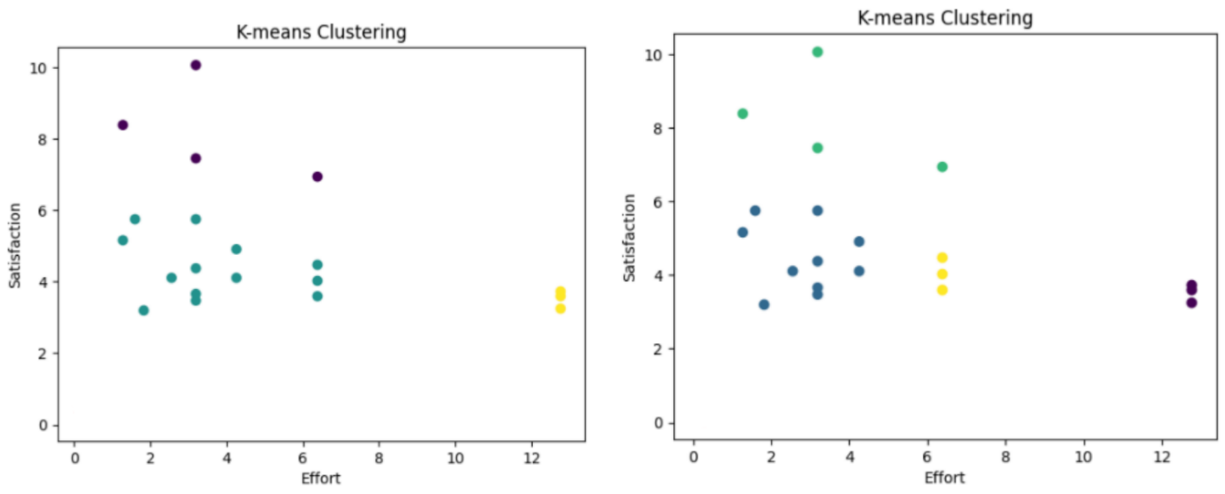


Figure 14 *AHP-based clustering: 3 & 4 Clusters from 20 Req*

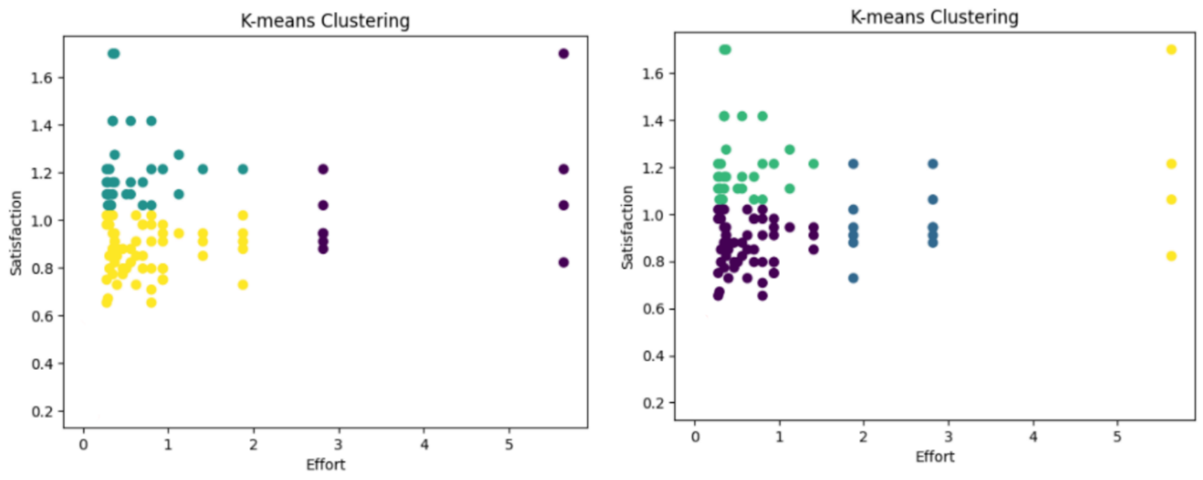


Figure 15 *AHP-based clustering: 3 & 4 Clusters from 100 Req*

5.3.2 Partition Around Medoids

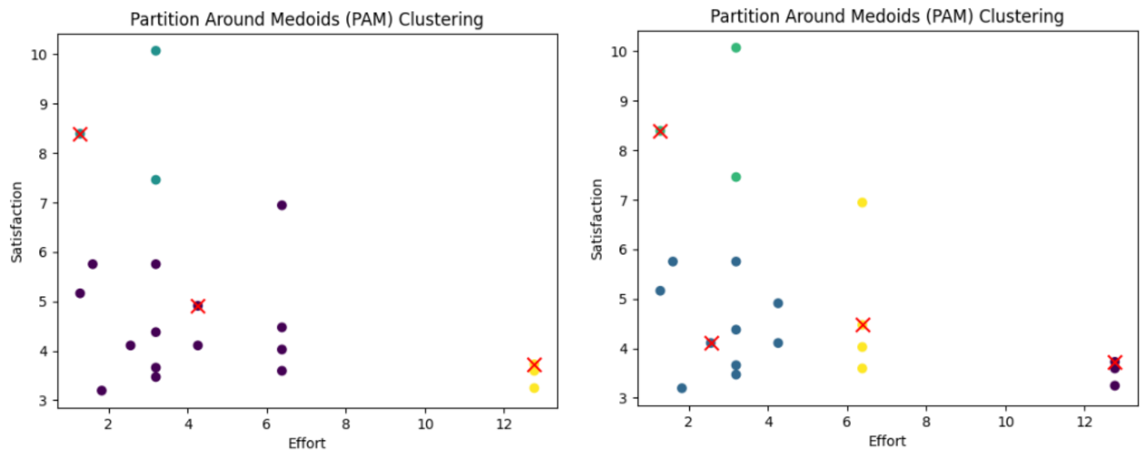


Figure 16 *AHP-based clustering: 3 & 4 Clusters from 20 Req*

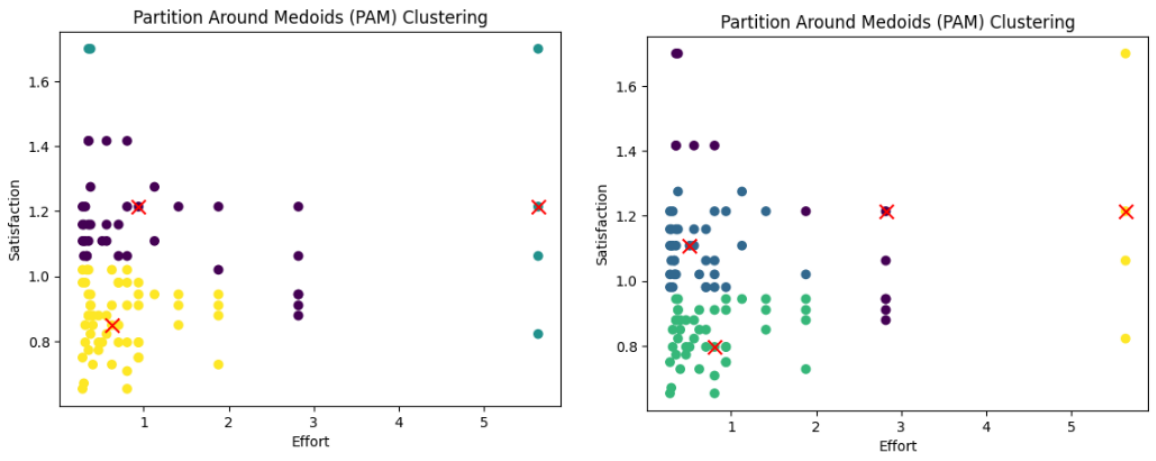


Figure 17 *AHP-based clustering: 3 & 4 Clusters from 100 Req*

5.3.3 Agglomerative Hierarchical Clustering

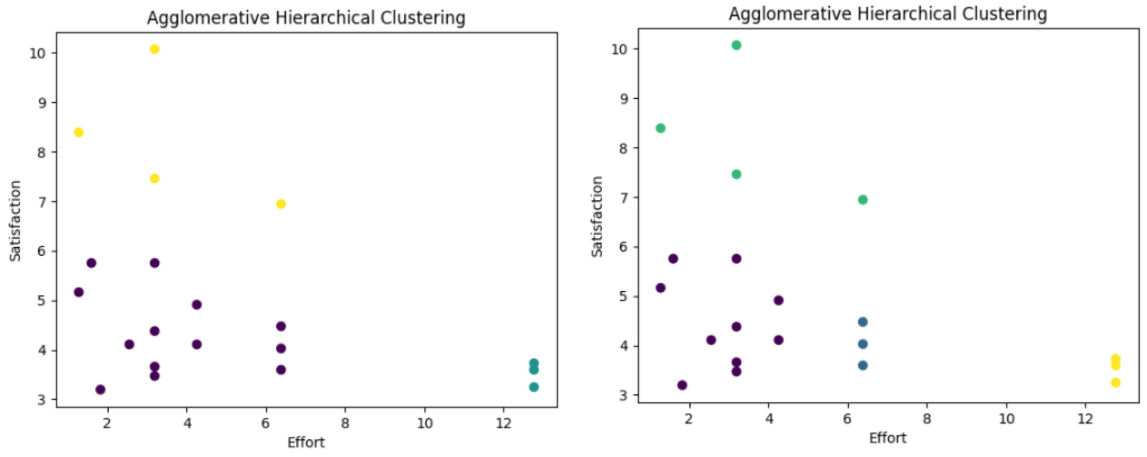


Figure 18 *AHP-based clustering: 3 & 4 Clusters from 20 Req*

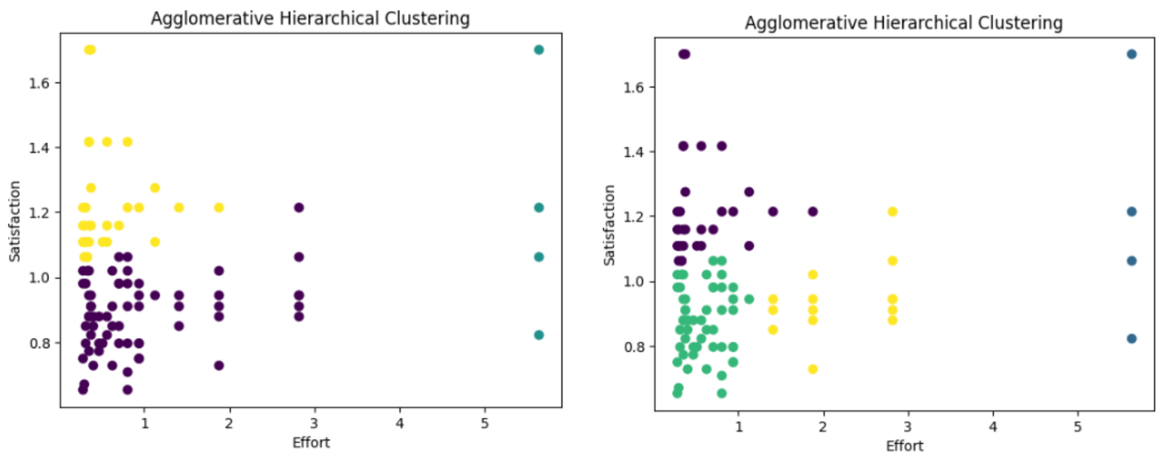


Figure 19 *AHP-based clustering: 3 & 4 Clusters from 100 Req*

5.3.4 Gaussian Mixture Models

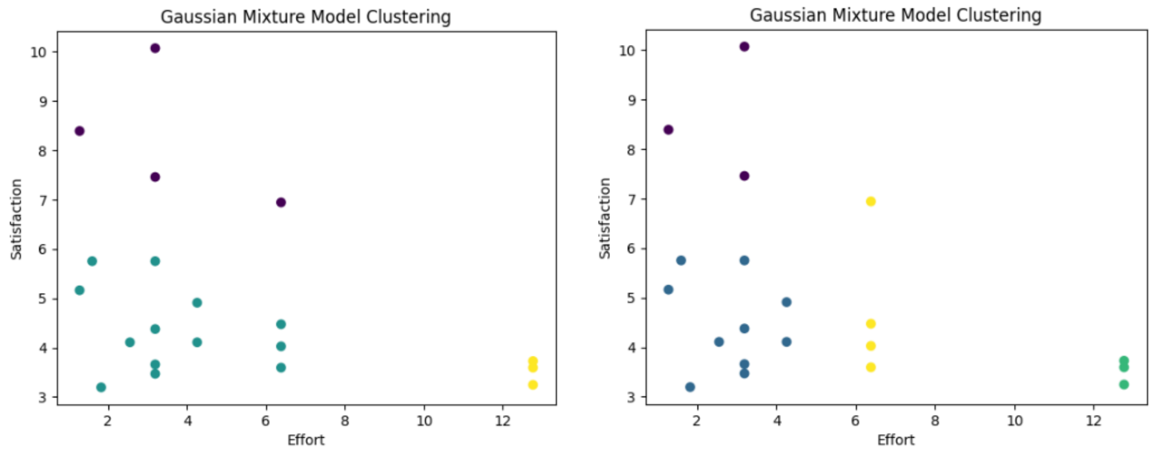


Figure 20 *AHP-based clustering: 3 & 4 Clusters from 20 Req*

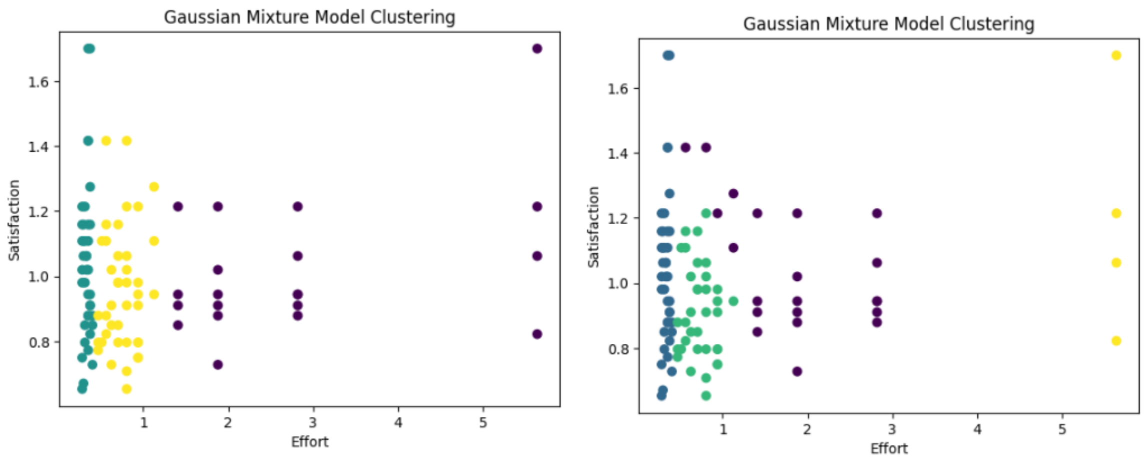


Figure 21 *AHP-based clustering: 3 & 4 Clusters from 100 Req*

5.3.5 BIRCH

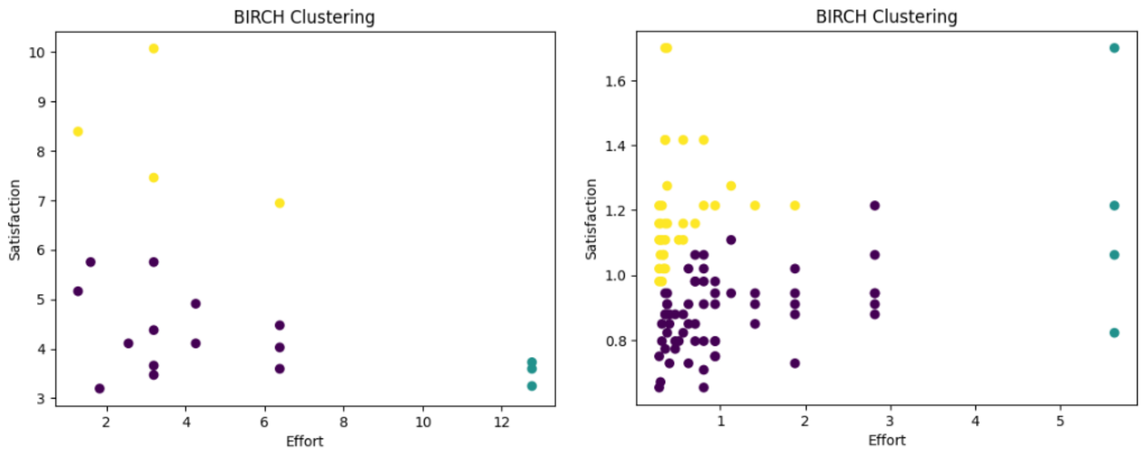


Figure 22 *AHP-based clustering: 3 Clusters from 20 & 100 Req*

5.4 Clusters Evaluation

To evaluate the quality of the clusters, three metrics were computed: the Dunn Index, the Silhouette Index, and the Calinski-Harabasz Index. These metrics assess the effectiveness of clustering in different ways, such as separation between clusters, well-separated clusters, and the ratio of between-cluster variance to within-cluster variance.

The paper [64] has assessed the values of three clustering algorithms: K-means, Hierarchical Clustering, and Partition Around Medoids (PAM) using quantitative computations. This thesis aims to provide a comparative analysis between the computed index values from [64] and the values obtained by the Analytic Hierarchy Process (AHP) approach. This analysis gauges the level of concurrence or divergence between the established methodology and the innovative AHP approach. Holistic comparison helps to understand the strengths and limitations of different clustering strategies and contributes to the broader understanding of effective clustering methodologies and their practical applications.

Furthermore, Gaussian Mixture Models (GMM) and BIRCH are also used to calculate the evaluation indices for both types of data sets, i.e., Quantitative and AHP to get a broader understanding of how proposed methodology acts with different clustering algorithms.

Given below are the results of all clustering algorithms for cluster evaluation metrics.

5.4.1 K-Means

Table 18 *K-Means Clusters Evaluation: 20 Requests*

20 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	0.209	**0.4336**
Silhouette	3	0.4666	**0.5690**
CH	3	22.9273	**33.7443**
Dunn	4	**0.2527**	0.2417
Silhouette	4	0.4176	**0.4863**
CH	4	24.3832	**34.1044**

Table 19 K-Means Clusters Evaluation: 100 Requests

100 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	0.0548	**0.2364**
Silhouette	3	0.4283	**0.4632**
CH	3	89.5132	**89.7174**
Dunn	4	0.0783	**0.2377**
Silhouette	4	0.3993	**0.4766**
CH	4	90.9959	**96.8018**

5.4.2 Partition Around Medoids

Table 20 *PAM Clusters Evaluation: 20 Requests*

20 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	0.2607	**2.7100**
Silhouette	3	0.4843	**0.5208**
CH	3	22.6144	**31.1727**
Dunn	4	0.3151	**1.5103**
Silhouette	4	0.4116	**0.4374**
CH	4	24.0329	**31.2174**

Table 21 *PAM Clusters Evaluation: 100 Requests*

100 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	0.0831	**0.3396**
Silhouette	3	**0.4308**	0.3943
CH	3	**89.5132**	46.9101
Dunn	4	0.0696	**0.3024**
Silhouette	4	0.3993	**0.3998**
CH	4	**88.7641**	64.6714

5.4.3 Agglomerative Hierarchical Clustering

Table 22 *AHC Clusters Evaluation: 20 Requests*

20 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	0.2576	**2.9804**
Silhouette	3	0.4549	**0.5690**
CH	3	18.6832	**33.7443**
Dunn	4	0.2482	**2.7427**
Silhouette	4	0.3561	**0.4863**
CH	4	**18.7909**	34.1044

Table 23 *AHC Clusters Evaluation: 100 Requests*

100 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	0.1096	**0.3472**
Silhouette	3	0.4278	**0.4327**
CH	3	**88.0933**	82.8722
Dunn	4	0.1096	**0.2518**
Silhouette	4	0.3964	**0.4576**
CH	4	82.5902	**95.1834**

5.4.4 Gaussian Mixture Models

Table 24 *GMM Clusters Evaluation: 20 Requests*

20 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	0.2739	**0.3723**
Silhouette	3	0.4568	**0.5690**
CH	3	22.5821	**33.744**
Dunn	4	0.1796	**0.310**
Silhouette	4	0.3839	**0.4905**
CH	4	22.0866	**33.633**

Table 25 *GMM Clusters Evaluation: 100 Requests*

100 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	**0.7259**	0.1706
Silhouette	3	**0.4285**	0.0743
CH	3	**90.674**	26.5032
Dunn	4	**0.5557**	0.077
Silhouette	4	**0.3721**	0.1082
CH	4	**90.7001**	36.2847

5.4.5 BIRCH

Table 26 *BIRCH Clusters Evaluation: 20 Requests*

20 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	**12.9526**	7.249
Silhouette	3	0.4672	**0.5690**
CH	3	18.9442	**33.744**

Table 27 *BIRCH Clusters Evaluation: 100 Requests*

100 Requirements Problem			
	Clusters	**Quantitative**	**AHP**
Dunn	3	**8.9139**	0.665
Silhouette	3	**0.4384**	0.4053
CH	3	**96.1607**	79.1779

5.5 Requirements Prioritisation

The prioritisation of requirements was meticulously carried out employing the MoSCoW method, a proven technique in project management. Within this process, a unique approach was taken by considering both overall satisfaction and the minimal effort required. Clusters displaying higher overall satisfaction and demanding the least effort were accorded the highest priority as "MUST" fulfillments. Conversely, as we moved to clusters that required slightly more effort but still provided substantial satisfaction, these were designated as "SHOULD" requirements, highlighting their significant yet negotiable nature. Similarly, clusters falling into the "COULD" category presented opportunities for further enhancement, as they delivered desirable satisfaction levels at a slightly higher effort cost. Lastly, clusters residing in the bottom right quadrant of the effort versus satisfaction graph were designated as "WON'T" for this iteration, indicating they were intentionally deferred due to higher effort requirements relative to the satisfaction gained. This dynamic prioritisation methodology encapsulates a comprehensive spectrum of considerations, offering a nuanced perspective for optimizing software requirements in alignment with project goals.

5.6 Results

A total of 54 distinct comparisons were conducted between the Analytic Hierarchy Process (AHP) and the quantitative datasets. These comparisons were aimed at evaluating the performance and effectiveness of the AHP approach in contrast to the quantitative data representation.

Among these 54 comparisons, it was observed that the AHP approach exhibited superior performance in 39 instances. This means that, in majority of the cases, AHP yielded more favorable outcomes or results compared to the quantitative data approach.

The significance of this finding lies in the consistent tendency of the AHP approach to outperform the quantitative data representation across a significant portion of the comparisons. This pattern of results underscores the potential benefits of using the AHP method for clustering or analyzing the given dataset, suggesting that it might be a more effective and reliable approach for generating meaningful insights or groupings.

Furthermore, for requirements prioritisation MoSCoW offers a framework with clear priority levels for requirements: "Must Have," "Should Have," "Could Have," and "Won't Have." This distinct classification minimises the possibility of forgetting important project components by ensuring that significant and essential requirements are recognised and addressed. Additionally, by offering a common language to discuss and comprehend demand priorities, MoSCoW facilitates effective communication amongst stakeholders. Making informed judgements about resource allocation and project scope is made easier because of the alignment of expectations. The MoSCoW technique also permits project planning to be flexible and adaptable. By reevaluating requirements and the categories, they fall under, the prioritisation can be changed as the conditions of the project change. This adaptability is especially useful when limitations or unforeseen circumstances have an impact on the project's course.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This study emphasises the value of employing data mining techniques as an efficient way to prioritise requirements in software engineering. It also highlights the Analytic Hierarchy Process' (AHP) outstanding superiority in software requirement prioritisation within the context of software engineering. Our results consistently show that AHP outperforms conventional quantitative data representations in the majority of the 54 comparisons carried out, based on a thorough evaluation of five clustering algorithms and three cluster evaluation indices. In addition to producing superior results, the integration of AHP with the MoSCoW requirement prioritisation framework also improved resource allocation, allowed for flexible planning, and increased stakeholder satisfaction. This study promotes the use of data mining techniques and AHP along with the MoSCoW framework as the recommended approach for upcoming projects in this important area.

6.2 Contribution

The thesis makes contributions to the fields of project management and software engineering by:

- Introducing innovative software requirement prioritisation by merging clustering and AHP.
- Integrating clustering methodologies with prioritisation techniques which enhances software project management by bridging technical advancements and strategic practices.
- Enhancing stakeholder communication, resource allocation, flexible planning, and decision-making in project management through structured framework, MoSCoW.

6.3 Future Work

In the current research, data sets were generated manually with the help of stakeholders. In the future, we can use machine learning algorithms. These algorithms can be trained on historical project data to learn the underlying patterns and characteristics of similar projects. By improving the overall efficiency of requirements prioritisation techniques, this integration could pave the way for more sophisticated and context-sensitive approaches to managing software requirements.

REFERENCES

[1] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin, “A systematic literature review of software requirements prioritization research,” *Inf Softw Technol*, vol. 56, no. 6, pp. 568–585, Jun. 2014, doi: 10.1016/j.infsof.2014.02.001.

[2] X. Franch and G. Ruhe, “Software release planning,” in *Proceedings of the 38th International Conference on Software Engineering Companion*, New York, NY, USA: ACM, May 2016, pp. 894–895. doi: 10.1145/2889160.2891051.

[3] M. Azzolini and L. I. Passoni, “Prioritization of Software Requirements: a Cognitive Approach,” in *Procedings of the Fourth International Workshop on Knowledge Discovery, Knowledge Management and Decision Support*, Paris, France: Atlantis Press, 2013. doi: 10.2991/.2013.13.

[4] I. Olaronke, I. Rhoda, and G. Ishaya, “An Appraisal of Software Requirement Prioritization Techniques,” *Asian Journal of Research in Computer Science*, pp. 1–16, Apr. 2018, doi: 10.9734/ajrcos/2018/v1i124717.

[5] A. Ahmad, M. Goransson, and A. Shahzad, “Limitations of the Analytic Hierarchy Process Technique with Respect to Geographically Distributed Stakeholders ,” *World Acad Sci Eng Technol*, pp. 111–116, 2010.

[6] P. Govender and V. Sivakumar, “Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019),” *Atmos Pollut Res*, vol. 11, no. 1, pp. 40–56, Jan. 2020, doi: 10.1016/j.apr.2019.09.009.

[7] K. El Emam and A. G. Koru, “A Replicated Survey of IT Software Project Failures,” *IEEE Softw*, vol. 25, no. 5, pp. 84–90, Sep. 2008, doi: 10.1109/MS.2008.107.

[8] Apoorva Mishra and Deepty Dubey, “A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios,” *International Journal of Advance Research in Computer Science and Management Studies* , vol. 1, no. 5, pp. 64–69, Oct. 2013.

[9] I. Sommerville, “Requirements Engineering Processes,” in *Softw. Eng.* , 7th ed.2004, pp. 1–52.

[10] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek, and P. Abrahamsson, “What Do We Know about Software Development in Startups?,” *IEEE Softw*, vol. 31, no. 5, pp. 28–32, Sep. 2014, doi: 10.1109/MS.2014.129.

[11] B. Bergman and B. Klefsjö, *Quality from customer needs to customer satisfaction*, 3rd ed. Studentlitteratur AB, 2010.

[12] Lena Karlsson, Åsa G. Dahlstedt2, and Johan Natt och Dag, “Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study ,” in *Proceedings of Eighth International Workshop on Requirements Engineering: Foundation for Software Quality*, Sep. 2002, pp. 101–112.

[13] R. B. Svensson *et al.*, “Prioritization of quality requirements: State of practice in eleven companies,” in *2011 IEEE 19th International Requirements Engineering Conference*, IEEE, Aug. 2011, pp. 69–78. doi: 10.1109/RE.2011.6051652.

[14] J. Karlsson, C. Wohlin, and B. Regnell, “An evaluation of methods for prioritizing software requirements,” *Inf Softw Technol*, vol. 39, no. 14–15, pp. 939–947, Jan. 1998, doi: 10.1016/S0950-5849(97)00053-0.

[15] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah, “Requirements Prioritization Techniques Comparison,” *Mod Appl Sci*, vol. 12, no. 2, p. 62, Jan. 2018, doi: 10.5539/mas.v12n2p62.

[16] T. L. Saaty, “The Analytic Hierarchy Process Mcgraw Hill,” *Agricultural Economics Review*, no. 70, p. 34, 1980.

[17] B. Regnell, M. Höst, J. N. och Dag, P. Beremark, and T. Hjelm, “An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software,” *Requir Eng*, vol. 6, no. 1, pp. 51–62, Feb. 2001, doi: 10.1007/s007660170015.

[18] Bruce L. Golden, Edward A. Wasil, and Patrick T. Harker, Eds., *“The analytic hierarchy process.” Applications and Studies*. Heidelberg, 1989.

[19] Sommerville, Iain, and Peter Sawyer, *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc.., 1997.

[20] S. Hatton, “Early Prioritisation of Goals,” in *Advances in Conceptual Modeling – Foundations and Applications*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 235–244. doi: 10.1007/978-3-540-76292-8_29.

[21] D. Tudor and G. A. Walter, “Using an Agile Approach in a Large, Traditional Organization,” in *AGILE 2006 (AGILE’06)*, IEEE, pp. 367–373. doi: 10.1109/AGILE.2006.60.

[22] P. Berander and A. Andrews, “Requirements Prioritization,” in *Engineering and Managing Software Requirements*, Berlin/Heidelberg: Springer-Verlag, pp. 69–94. doi: 10.1007/3-540-28244-0_4.

[23] Pradeep Rai and Shubha Singh, “A Survey of Clustering Techniques,” *Int J Comput Appl*, vol. 7, Oct. 2010.

[24] A. Saxena *et al.*, “A review of clustering techniques and developments,” *Neurocomputing*, vol. 267, pp. 664–681, Dec. 2017, doi: 10.1016/j.neucom.2017.06.053.

[25] Prof. Neha Soni and Dr. Amit Ganatra, “Comparative study of several Clustering Algorithms,” *International Journal of Advanced Computer Research*, vol. 2, no. 6, Dec. 2012.

[26] Shraddha K.Popat and Emmanuel M., “Review and Comparative Study of Clustering Techniques,” *International Journal of Computer Science and Information Technologies*, vol. 5, no. 1, pp. 805–812, 2014.

[27] R.Saranya and P.Krishnakumari, “Clustering with Multi view point-Based Similarity Measure using NMF,” *nternational Journal of scientific research and management*, vol. 1, no. 6, pp. 316–322, 2013.

[28] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering,” *ACM Comput Surv*, vol. 31, no. 3, pp. 264–323, Sep. 1999, doi: 10.1145/331499.331504.

[29] J. A. Hartigan and M. A. Wong, “Algorithm AS 136: A K-Means Clustering Algorithm,” *Appl Stat*, vol. 28, no. 1, p. 100, 1979, doi: 10.2307/2346830.

[30] K. P. Sinaga and M.-S. Yang, “Unsupervised K-Means Clustering Algorithm,” *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.

[31] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern Recognit*, vol. 36, no. 2, pp. 451–461, Feb. 2003, doi: 10.1016/S0031-3203(02)00060-2.

[32] L Kaufman and PJ Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons., 2009.

[33] M. Van der Laan, K. Pollard, and J. Bryan, “A new partitioning around medoids algorithm,” *J Stat Comput Simul*, vol. 73, no. 8, pp. 575–584, Aug. 2003, doi: 10.1080/0094965031000136012.

[34] Y. Zhang *et al.*, “Gaussian Mixture Model Clustering with Incomplete Data,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 17, no. 1s, pp. 1–14, Jan. 2021, doi: 10.1145/3408318.

[35] E. M. Voorhees, “Implementing agglomerative hierarchic clustering algorithms for use in document retrieval,” *Inf Process Manag*, vol. 22, no. 6, pp. 465–476, Jan. 1986, doi: 10.1016/0306-4573(86)90097-X.

[36] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, “Efficient agglomerative hierarchical clustering,” *Expert Syst Appl*, vol. 42, no. 5, pp. 2785–2797, Apr. 2015, doi: 10.1016/j.eswa.2014.09.054.

[37] K. Peng, L. Zheng, X. Xu, T. Lin, and V. C. M. Leung, “Balanced Iterative Reducing and Clustering Using Hierarchies with Principal Component Analysis (PBirch) for Intrusion Detection over Big Data in Mobile Cloud Environment,” 2018, pp. 166–177. doi: 10.1007/978-3-030-05345-1_14.

[38] T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH,” *ACM SIGMOD Record*, vol. 25, no. 2, pp. 103–114, Jun. 1996, doi: 10.1145/235968.233324.

[39] “Guidelines for performing Systematic Literature Reviews in Software Engineering,” Keele, Durham , Jul. 2007.

[40] M. Kassab and N. Kilicay-Ergin, “Applying analytical hierarchy process to system quality requirements prioritization,” *Innov Syst Softw Eng*, vol. 11, no. 4, pp. 303–312, Dec. 2015, doi: 10.1007/s11334-015-0260-8.

[41] J. Ali Khan, I. Ur Rehman, Y. Hayat Khan, I. Javed Khan, and S. Rashid, “Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique,” *International Journal of Modern Education and Computer Science*, vol. 7, no. 11, pp. 53–59, Nov. 2015, doi: 10.5815/ijmecs.2015.11.06.

[42] J. A. Khan, Izaz-ur-Rehman, S. P. Khan, I. Qasim, and Y. H. Khan, “An Evaluation of Requirement Prioritization Techniques with ANP,” *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 7, 2016.

[43] S. U. R. Khan, S. P. Lee, M. Dabbagh, M. Tahir, M. Khan, and M. Arif, “RePizer: a framework for prioritization of software requirements,” *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 8, pp. 750–765, Aug. 2016, doi: 10.1631/FITEE.1500162.

[44] V. Kouhdaragh, D. Tarchi, A. Vanelli-Coralli, and G. E. Corazza, “A Cost Function Based Prioritization Method for Smart Grid Communication Network,” 2017, pp. 16–24. doi: 10.1007/978-3-319-47729-9_2.

[45] S. Parthasarathy and M. Daneva, “An approach to estimation of degree of customization for ERP projects using prioritized requirements,” *Journal of Systems and Software*, vol. 117, pp. 471–487, Jul. 2016, doi: 10.1016/j.jss.2016.04.006.

[46] K. S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, “Fuzzy_MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements,” in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, IEEE, Jul. 2017, pp. 433–437. doi: 10.1109/ICICICT1.2017.8342602.

[47] M. Abbas, I. Inayat, N. Jan, M. Saadatmand, E. Paul Enoiu, and D. Sundmark, “MBRP: Model-Based Requirements Prioritization Using PageRank Algorithm,” in *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, Dec. 2019, pp. 31–38. doi: 10.1109/APSEC48747.2019.00014.

[48] P. MADZÍK, Ľ. LYSÁ, and P. BUDAJ, “Determining the Importance of Customer Requirements in QFD – A New Approach based on Kano Model and its Comparison with Other Methods,” vol. 20, no. 168, pp. 3–15, Feb. 2019.

[49] M. S. Jahan, F. Azam, M. W. Anwar, A. Amjad, and K. Ayub, “A Novel Approach for Software Requirement Prioritization,” in *2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*, IEEE, Oct. 2019, pp. 1–7. doi: 10.1109/CONISOFT.2019.00012.

[50] M. Yaseen, A. Mustapha, and N. Ibrahim, “Prioritization of Software Functional Requirements from Developers Perspective,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020.

[51] N. Mohamed, S. Mazen, and W. Helmy, “E-AHP: An Enhanced Analytical Hierarchy Process Algorithm for Priotrizing Large Software Requirements Numbers,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 7, 2022, doi: 10.14569/IJACSA.2022.0130725.

[52] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, “Efficient agglomerative hierarchical clustering,” *Expert Syst Appl*, vol. 42, no. 5, pp. 2785–2797, Apr. 2015, doi: 10.1016/j.eswa.2014.09.054.

[53] F. Fouedjio, “A hierarchical clustering method for multivariate geostatistical data,” *Spat Stat*, vol. 18, pp. 333–351, Nov. 2016, doi: 10.1016/j.spasta.2016.07.003.

[54] Z. Li, G. Wang, and G. He, “Milling tool wear state recognition based on partitioning around medoids (PAM) clustering,” *The International Journal of Advanced Manufacturing Technology*, vol. 88, no. 5–8, pp. 1203–1213, Feb. 2017, doi: 10.1007/s00170-016-8848-1.

[55] G. Pitolli, L. Aniello, G. Laurenza, L. Querzoni, and R. Baldoni, “Malware family identification with BIRCH clustering,” in *2017 International Carnahan Conference on Security Technology (ICCST)*, IEEE, Oct. 2017, pp. 1–6. doi: 10.1109/CCST.2017.8167802.

[56] K. P. Sinaga and M.-S. Yang, “Unsupervised K-Means Clustering Algorithm,” *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.

[57] M. Faizan, M. F., S. Ismail, and S. Sultan, “Applications of Clustering Techniques in Data Mining: A Comparative Study,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 12, 2020, doi: 10.14569/IJACSA.2020.0111218.

[58] K. B, “A Comparative Study on K-Means Clustering and Agglomerative Hierarchical Clustering,” *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 5, pp. 1600–1604, May 2020, doi: 10.30534/ijeter/2020/20852020.

[59] Y. Zhang *et al.*, “Gaussian Mixture Model Clustering with Incomplete Data,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 17, no. 1s, pp. 1–14, Jan. 2021, doi: 10.1145/3408318.

[60] C. M. Weber, D. Ray, A. A. Valverde, J. A. Clark, and K. S. Sharma, “Gaussian mixture model clustering algorithms for the analysis of high-precision mass measurements,” *Nucl Instrum Methods Phys Res A*, vol. 1027, p. 166299, Mar. 2022, doi: 10.1016/j.nima.2021.166299.

[61] V. C. Pezoulas, N. S. Tachos, G. Gkois, I. Olivotto, F. Barlocco, and D. I. Fotiadis, “Bayesian Inference-Based Gaussian Mixture Models With Optimal Components Estimation Towards Large-Scale Synthetic Data Generation for *In Silico* Clinical Trials,” *IEEE Open J Eng Med Biol*, vol. 3, pp. 108–114, 2022, doi: 10.1109/OJEMB.2022.3181796.

[62] H. Zhao, “Design and Implementation of an Improved K-Means Clustering Algorithm,” *Mobile Information Systems*, vol. 2022, pp. 1–10, Sep. 2022, doi: 10.1155/2022/6041484.

[63] L. Lehtola and M. Kauppinen, “Suitability of requirements prioritization methods for market-driven software product development,” *Software Process: Improvement and Practice*, vol. 11, no. 1, pp. 7–19, Jan. 2006, doi: 10.1002/spip.249.

[64] J. del Sagrado and I. M. del Águila, “Assisted requirements selection by clustering,” *Requir Eng*, vol. 26, no. 2, pp. 167–184, Jun. 2021, doi: 10.1007/s00766-020-00341-1.

[65] D. Greer and G. Ruhe, “Software release planning: an evolutionary and iterative approach,” *Inf Softw Technol*, vol. 46, no. 4, pp. 243–253, Mar. 2004, doi: 10.1016/j.infsof.2003.07.002.

[66] J. del Sagrado, I. M. del Águila, and F. J. Orellana, “Multi-objective ant colony optimization for requirements selection,” *Empir Softw Eng*, vol. 20, no. 3, pp. 577–610, Jun. 2015, doi: 10.1007/s10664-013-9287-3.

[67] E. Simmons, “Requirements triage: what can we learn from a ‘medical’ approach?,” *IEEE Softw*, vol. 21, no. 4, pp. 86–88, Jul. 2004, doi: 10.1109/MS.2004.25.