An Appraisal of Software Requirement Prioritization Techniques

Iroju Olaronke[1*], Ikono Rhoda[2], Gambo Ishaya[3]

[1]Department of Computer Science, Adeyemi College of Education, Ondo, Nigeria
[2,3]Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria

ABSTRACT

In recent times, there is a paradigm shift from the use of paper based systems to the use of software systems in all spheres of life. However, the development of high quality, cost effective and useable software systems is a major challenge. One of the major obstacles confronting the successful implementation of software systems is the inability to implement all stakeholders' requirements in software development projects. This constraint is usually due to limited human resources, budget and time. Thus, most software systems have failed. It, therefore, becomes pertinent to prioritize software requirements. Requirement prioritization involves the selection of requirements that are considered more important from an accumulated list of stakeholders' requirements. There are two techniques that are used for categorizing software requirements. These techniques include the requirement prioritization methods and the negotiation methods. Requirement prioritization methods are based on different scales which include nominal scale, ordinal scale and ratio scale. The accuracy of these methods, however, is a challenge especially when prioritizing large number of requirements.
Aims: Hence, this paper reviews different techniques for prioritizing requirements by highlighting their strengths and weaknesses. Techniques such as binary search tree, AHP, hierarchy AHP, priority group/Numerical Analysis, bubble sort, MoSoW, simple ranking and Planning Game were analyzed and compared in this study.
Methodology: The study is based on previous literature on requirement prioritization.
Results: The study showed that numerical assignment and simple ranking methods require less time in the prioritization process and they also have low scalability and reliability. The study also showed that the analytic hierarchy process requires more time for requirement prioritization; it is reliable but it is not scalable. The study also revealed that it is difficult to prioritize requirements with the existing prioritization techniques when multiple stakeholders are involved.
Conclusion: The study suggests that future researches should be based on the design of requirement prioritization techniques that will have the ability to accommodate large stakeholders and requirements.

Keywords: requirements, requirement engineering, requirement prioritization, software,

1. INTRODUCTION (ARIAL, BOLD, 11 FONT, LEFT ALIGNED, CAPS)

The use of software systems is increasingly becoming an integral part of all spheres of lives such as banking system, educational system and the healthcare system. For instance, in 2009, President George Bush set the goal of providing every American citizen with the ability to have Electronic Health Records (EHRs) by 2014 [1]. Furthermore, President Barack H. Obama signed into law the American Recovery and Reinvestment Act, which reserves $17 billion exclusively for the development of EHR systems [2]. This is because software systems play key roles in healthcare by reducing medical errors and cost, facilitating interoperability amongst healthcare providers and patients as well as providing facilities for remainders and appointment scheduling. However, as the use of software system

* E-mail address: irojuolaronke@gmail.com.

becomes prevalent in all sectors of life so also is the volume of requirements derived from its stakeholders. Consequently, the requirements elicited from stakeholders are usually competing, conflicting and ambiguous. Hence, research has shown that about 70% of software systems have failed or have not satisfied the end users [3]. Consequently, people avoid the use of software systems even after investing much expense and time. One of the major factors responsible for this challenge is the impossibility of implementing all stakeholders' requirements into the software system due to limited human resources such as budget and time. Hence, there is a need to prioritize software requirements.

Requirements prioritization according to Karlsson and Ryan [4] is the process of selecting the appropriate set of requirements from a multitude of competing and conflicting expectations elicited from various stakeholders involved in a software development project. Conversely, Firesmith [5] defines requirements prioritization as the process of determining the implementation order of the requirements for implementing a software system. The benefits of requirement prioritization are numerous. For instance, Karlsson and Ryan [4]. pointed out that requirement prioritization reduces software development cost and time by 40%. In addition, requirement prioritization helps to identify the most valuable requirements from a set of voluminous and ambiguous requirements as well as reduce software failure. Berander and Andrews [6] emphasized that requirement prioritization resolve disagreements or disparities amongst stakeholders. Hence, requirement prioritization improves the quality of a software release. Requirement prioritization can therefore be said to be one of the most crucial stages in software development process.

There are two techniques that are used for categorizing software requirements. These techniques include the requirement prioritization method and the negotiation method. Requirement prioritization method is categorized on three basic scales which include nominal scale, ordinal scale and ratio scale [7]. In nominal scale, requirements are assigned to different priority groups, where all requirements in one priority group are of equal priority. The ordinal scale method usually results in an ordered list of requirements while the ratio scale method provides the relative difference between requirements. Requirements negotiation focuses on the assignment of priorities to requirements through the consensus of the stakeholders. Typical example of requirement negotiation is winwin method. Each of these techniques is characterized by different challenges which include reliability problems, problems of consistency check, reliability with multiple stakeholders and difficulty when prioritizing large numbers of requirements [7]. Hence, it becomes difficult for stakeholders to choose the right technique when prioritizing requirements. Consequently, this study examines diverse requirement prioritization techniques with the aim of assisting stakeholders to choose the right requirement prioritizing technique during a software development project.

2. OVERVIEW OF SOFTWARE REQUIREMENT

There is no precise definition of software requirement as different authors view software requirement in different contexts. For instance, IEEE [8] defines a requirement in the following ways:

a. A condition or capability needed by a user to solve a problem or achieve an objective.
b. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
c. A documented representation of a condition or capability as in (a) or (b).

Furthermore, International Standard Organization [8] defined a requirement as a statement which translates or expresses a need and its associated constraints and conditions. In addition, Saranya et al. [9] viewed software requirement as what a system is required to do along with the environment it is intended to operate in. According to Saranya et al. [9], software requirements provide the description of a system, its behavior or functionalities, application domain information, system constraints, specifications and attributes. Gambo et al. [10] also viewed a requirement as an expression of what a software product must do to add value to its users and an expression of the limitations on the choices that developers make when implementing the software. Software requirements may be categorized as user requirements and system requirements [11]. Maiden [12] refers to a user requirement as an instruction which a user provides while a system requirement expresses a desirable property of a system that leads to the achievement of at least one user requirement. Requirements may also be classified as primary requirements and derived requirements. The primary requirements are those requirements that are elicited directly from the stakeholders while the derived requirements are those requirements that are obtained from the primary requirements. Requirements according to Aggarwal

* E-mail address: irojuolaronke@gmail.com.

and Singh [13] may also be known, unknown or undreamt. The known requirements are those that the stakeholders believe must be implemented, the unknown requirements are those that are usually forgotten by the stakeholders because they are not urgently needed while the undreamt requirements are those requirements that the stakeholders may not think of due to the limitation in the domain knowledge. The requirements elicited from stakeholders may also be functional or non-functional regardless of its category. Functional requirements (FRs) describe system services or functions while non-functional requirements (NFRs) are quality requirements that stipulate how well a system performs its functions. Non-functional requirements are therefore attributes that the system must possess.

3. REQUIREMENT PRIORITIZATION

Software is critical to the advancement of almost all areas of human endeavors. Software however is more than writing programs. It consists of programs; documentation of any facet of the program and the procedures used to setup and operate the software system [13]. There have however been serious challenges on the cost, timeliness, maintenance as well as the quality of several software products. Software engineering has no doubt helped in resolving these problems by producing software systems that are developed within a stipulated time and budget and are also of high quality and useable. One of the most crucial activities in software engineering is requirement engineering. Requirement engineering can be defined as a disciplined application of proven principles, methods, tools and notations that are used to describe proposed software intended behavior and its associated constraints [13]. The processes used for requirement engineering vary widely depending on the application domain, the people involved and the stakeholders involved in the software development process. However, there are generic activities that are common to the requirement engineering process. These activities are iterative in nature and they include requirements elicitation, requirements analysis, requirements documentation and requirements review. Requirement elicitation is basically the process of gathering requirements by reviewing available documents, observing existing systems and interacting with the stakeholders through techniques such as interviews, scenarios, brainstorming, Facilitated Application Specification Technique (FAST) as well as Quality Function Deployment (QFD). The requirements gathered are analyzed to identify inconsistencies, contradictions, defects and omissions. This is to ensure that the requirements are correct, complete, consistent and unambiguous. Requirement prioritization is one of the major activities carried out during requirement analysis. Requirement documentation ensures that requirements are presented in a consistent format while requirement review ensures that the quality of the software requirement specification is improved.

Requirements prioritization as one of the most important activities of requirement engineering is concerned with the selection of the most significant requirements from a list of voluminous requirements gathered from diverse stakeholders. Requirements prioritization according to Avesani et al. [14] is the process of deriving an order relation on a given set of requirements, with the ultimate goal of obtaining a shared rationale for partitioning them into subsequent product releases. Hence, the basic goal of requirement prioritization according to Achimugu [15] is to select the most important requirements from the set of candidate requirements as perceived by relevant stakeholders. In addition, Firesmith [5] viewed requirements prioritization as the process of determining the implementation order of the requirements that are to be implemented in a software system. Requirement prioritization can therefore be said to be one of the most essential activities of software engineering as it involves making critical decisions that will make the software useable, cost effective and of high quality. Fig.1. shows the relationship between requirements prioritization, requirements engineering and software engineering.

* E-mail address: irojuolaronke@gmail.com.

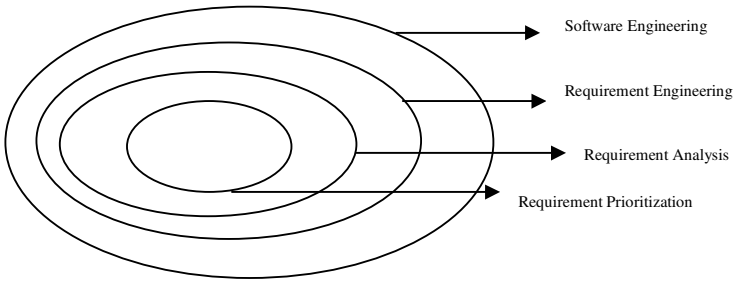

Fig.1. Relationship between software engineering, requirements engineering, requirements analysis and requirements prioritization (Adapted from [7])

3.1 Dimensions for Prioritizing Software Requirements

There are different aspects or dimensions that are used for prioritizing requirements [16]. These factors include importance, time, cost, value, penalty, risk as well as precedence.

3.1.1 Importance

Importance according to Ma [7] is a multifaceted term that denotes high market value, high quality of the product, or urgency of implementation. Hence, requirements of high importance are given higher priorities.

3.1.2 Time

Time refers to the total time spent on successfully implementing a candidate requirement.

3.1.3 Risk

Mursu [17] defined risk in software development as a product of uncertainty associated with other project risk factors and the magnitude of potential loss due to project failure. In addition, Mustafa and Al-Bahar [18] view risk as the degree of likelihood that a project will fail to achieve its time, cost or quality goals. The definitions of risk basically have two characteristics which are probability and loss. Risk probability is the likelihood that a risk would occur while loss is defined as an outcome that fall short of what is expected by the stakeholders. Boban et al. [19] identified the causes of requirement risk as lack of clear product vision, lack of agreement on product requirements, un-prioritized requirements, and new applications with uncertain requirements, rapidly changing requirements, ineffective requirements, change management process and inadequate impact analysis of requirements changes. If requirements related risk factors are not controlled, a software system that does not meet the users' expectations will be developed, the cost of maintaining and enhancing such systems may be high, and the system may be unreliable and prone to errors [20].

3.1.4 Cost

Cost is the amount spent on successfully implementing candidate requirements.

3.1.5 Value

Value is the importance attached to a set of requirements when compared to other requirements

3.1.6 Penalty

Penalty refers to the consequences that are faced if a requirement is not implemented.

3.1.7 Precedence

Precedence refers to the condition that requires the completion of a set of requirements before another set of requirements are implemented. Hence, requirements prioritization is aimed at selecting the most appropriate requirements from a set of candidate requirements in order to satisfy the interests, technical constraints and preferences of the stakeholders involved in the software development process [21].

Karlsson et al. [22] posit that requirement prioritization consist of three consecutive stages which include preparation, execution and presentation stages. In the preparation stage, the stakeholders structure their requirements according to the principles of the requirement prioritization method to be used. In the execution stage, the actual requirement prioritization is done using the information

* E-mail address: irojuolaronke@gmail.com.

provided in the preparation stage while the result of the prioritization is presented to the stakeholders in the presentation phase. This process is illustrated in Fig. 2.

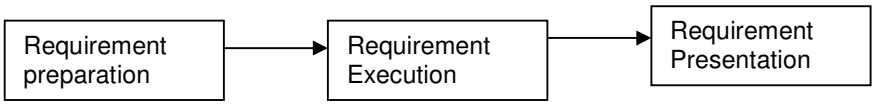

Fig.2. Stages of Requirement Prioritization

3.3 Benefits and Challenges of Prioritizing Software Requirements

The advantages of requirement prioritization are numerous. Hence, different authors have highlighted the benefits of requirement prioritization at various times. For instance, Kukreja [23] reported that requirement prioritization facilitates users' involvement in the software development process. Hence, all stakeholders involved in the system development agree on the requirements that a particular software release should contain. Another benefit of requirement prioritization as viewed by Hatton [16] is that requirements prioritization reduces software failure which is a major challenge in software development process. This is in line with the research result of Chaos [24] which reported that only 32% of software development projects are delivered on time, on budget, and with the required features and functions, 44% are delivered late, over budget, and/or with less than the required features and functions while 24% of software development projects are usually terminated before completion or delivered but never used. Hence, prioritizing requirements reduces the probability of software failures. Requirement prioritization helps to manage limited resources such as cost and duration of a software project by addressing high priority requirements before the consideration of low priority ones. Furthermore, requirement prioritization helps in planning for software releases since not all the requirements elicited from multiple stakeholders can be implemented in a single software release [25]. In addition, prioritizing requirements leads to increased users' satisfaction because it increases the likelihood that the stakeholders' most significant requirements are implemented and delivered first [5].

In spite of the numerous benefits of requirements prioritization, there are several challenges that are associated with prioritizing requirements. One of the major factors hindering requirement prioritization is changing priorities which are usually caused by changing stakeholders and requirements. The stakeholders in a software development project may change and this might result in the change of the software requirements as well as the change in the priorities of requirements [26]. In addition, the diverse views of stakeholders make it very difficult for the system analyst to identity requirements that are of high priorities. It is also difficult to efficiently prioritize requirements in domains such as the healthcare domain with large requirements. This is because this domain is made up of diverse user communities who have different requirements and priorities that may be incompatible, conflicting or contradictory. It is also difficult to prioritize requirements when resources such as cost and time are limited. Lack of trust amongst stakeholders can also be a major challenge of requirement prioritization. This is because customers that desire to implement all software requirements might assume that the developers only desire to prioritize the requirements because of their desire to eliminate some of the more difficult or risky requirements [5].

4. REQUIREMENT PRIORITIZATION TECHNIQUES

Requirement prioritization techniques can be divided into two categories. These include requirement prioritization methods and requirement negotiation approaches.

4.1 Requirement Prioritization Methods

The requirement prioritization methods are based on assigning values to software requirements. There are several methods used for prioritizing requirements. These prioritization techniques can be categorized as nominal scale, ordinal scale, ratio scale, machine learning methods and hybridized techniques. This is as shown in Fig.3.

* E-mail address: irojuolaronke@gmail.com.

4.1.1 Nominal Scale Method

In the nominal scale method, requirements are assigned to different priority groups such that the requirements in a particular group have equal priorities. Examples of nominal scale requirements prioritization method include numerical assignment, MoScoW and top-ten requirements.

4.1.1.1 Numerical assignment

In numerical assignment method, requirements are classified into different groups based on the levels of their priority, such that the requirements in a group are of equal priority. Typical examples of priority groups include low, medium and high [27]. Another example of a priority group include critical, standard and optional. Another approach to this method is presented by Brackett [28] who classified requirements on a scale of 1 to 5, where the numbers 1-5 indicate the following:

1. Does not matter
2. Not important
3. Rather important.
4. Very important and
5. Mandatory.

Finally, the average value given by all stakeholders are considered as the ranking for the requirement. The difficulty of this approach is that the requirements in each group have the same priority, which means that each requirement does not get a unique priority [29, 30]. In addition, it is difficult to determine absolute information because different stakeholders have diverse opinions which make the information obtained relative [31]. In addition, the classification of requirements into different groups might confuse the stakeholders [32, 33]

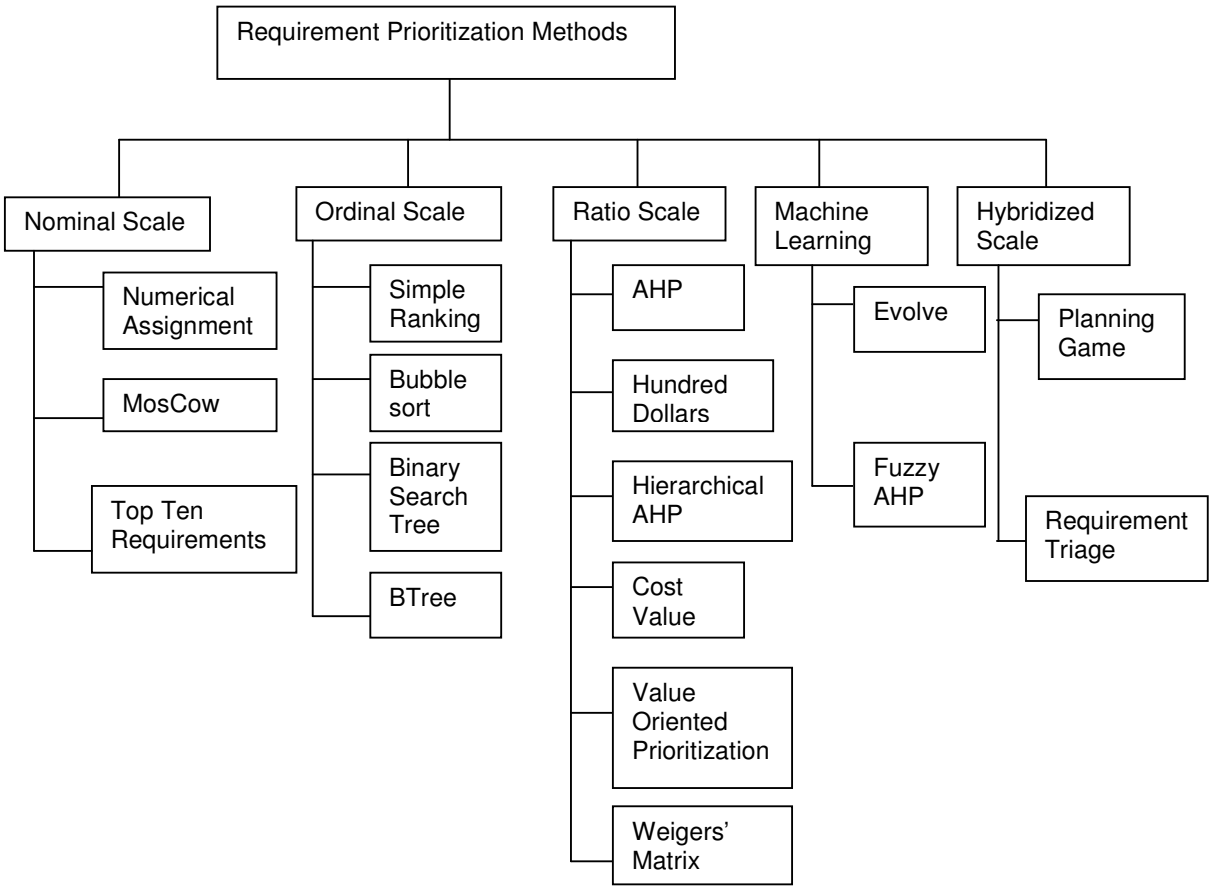

Fig.3. Requirement Prioritization Methods (Adapted from [15])

* E-mail address: irojuolaronke@gmail.com.

4.1.1.2 MoScoW

This is another form of numerical assignment method that groups requirements into four priority groups. These groups include MUST have, SHOULD have, COULD have and WONT have. Requirements in each of these groups are also of equal priority. The requirements in the "Must have" group are those that must be implemented in the software before it is released while the "Should have" group represents requirements that are also important but not necessary for delivery in the current software release. In addition, the requirements in the "Could have" group are desirable but not necessary, and the requirements in this group will only be implemented if time and resources are abundant. The "Won't have" group contains requirements that are of low priority and are not appropriate for the software release. The disadvantage of the MoSCoW prioritization technique is that requirements are ambiguous when it comes to the "Wont have" group as it does not specify when the requirements in this group would be implemented.

4.1.2.3 Top-Ten Requirements

In the top-ten requirements approach, the stakeholders are allowed to choose their top-ten requirements from a set of requirements. This approach is suitable when there is a small number of requirements and stakeholders.

4. 1. 2 Ordinal Scale

The ordinal scale techniques according to Ma [7] result in an ordered list of requirements. Examples of methods that employ the ordinal scale techniques are simple ranking method, the bubble sort method, binary search tree and the btree method.

4.1.2.1 Simple Ranking Method

In the simple ranking method, requirements are ranked from 1 to n, where n is any integer value that represents the number of requirements. Requirements with higher priority are ranked 1 while lower priority requirements are ranked n. The advantage of this method is that each requirement has its own numerical value. The drawback of this technique is that it does not provide criteria for categorization [35]. It is also difficult to implement on large number of requirements.

4.1. 2.2 Bubble sort

The bubble sort method according to Aho et al. [36] is a method for sorting elements. In this technique, adjacent requirements are compared in a pair-wise manner and swapped if they are in the wrong order. This process continues until no more swaps are required which is an indication that the list of requirement is sorted [37]. The major drawback of the bubble sort method is that it is difficult to evaluate the relative priority differences among the requirements [15].

4.1. 2.3 Binary search tree

The binary search tree is a tree that has a parent-child relationship. The parent node usually contains at most two children [36]. Each parent node in the tree represents a requirement. Requirements with lower priority are arranged on the left side of the parent node while requirements placed on the right side of the parent node are of higher priority. A parent node with no child node is referred to as a leaf. It is however difficult to evaluate the relative priority differences among requirements [37]. In addition, the binary search tree produces unreliable results [38]. Another major criticism of the binary search tree is that it only provides a simple ranking of requirements without assigning any priority values [38].

4.1. 2.4 Btree

Btree uses the same concept as the Binary search tree. The advantage of the Btree method over the binary search tree is that it reduces the number of requirements compared.

4.1.3 Ratio Scale Method

The ratio scale is usually believed to be the most viable of all the scales because it has the capacity of ordering, as well as determining intervals or relative difference between requirements. Typical examples of requirement prioritization method that employ the ratio scale method include the hundred dollars or cumulative voting method, Analytic Hierarchy Process (AHP), Cost-Value Approach and the value orientated prioritization.

* E-mail address: irojuolaronke@gmail.com.

4.1. 3.1 Hundred Dollar Method

The hundred dollars or cumulative voting method is otherwise known as proportional technique. It was introduced by Leffingwell et al. [39]. The hundred dollars method was designed to determine important requirements by distributing a fabricated $100 note across requirements according to their degree or level of importance. The requirements are then sorted in ascending order in order to determine the number of dollar notes each requirement has earned. The drawback of this technique is that it is not suitable for a large number of requirements.

4.1. 3.2 Analytic Hierarchy Process (AHP)

The AHP was introduced by Thomas Saaty in 1980 for complex decision making. AHP was applied to software engineering by Joachim Karlsson and Kevin Ryan in 1997 [40]. AHP involves the pair-wise comparison of requirements in order to determine which of the two is of higher priority and to what extent. If n requirements are to be prioritized using AHP, then n*(n-1)/2 pair-wise comparisons are required. AHP results in an n by n matrix for n requirements. AHP uses a preference scale which generally ranges from 1 to 9, where 1 indicates requirements of equal value and 9 indicates extreme value [41]. The preference scale used for the AHP technique is as shown in Table.1.

Table 1. Preference Scale Used for the AHP Technique [4]

Particulars	Value	Methods
1	of equal value	two requirements are of equal value
3	slightly more value	experience diligently favours one requirement over the other
7	essential or strong value	a requirement is strongly favoured and its dominance is demonstrated in practice
9	extreme value	the evidence favouring one over another is of the highest possible order of affirmation
2,4,6,8	intermediate values between two adjacent judgments	when compromise is needed
Reciprocals	if requirement *i* has one of the above numbers assigned to it when compared with requirement *j*, then *j* has the reciprocal value when compared with *i*	

The AHP method according to Mead [34] consists of five basic steps which include:

1. The review of candidate requirements for completeness.

* E-mail address: irojuolaronke@gmail.com.

2. The application of pair-wise comparison method to assess the relative value of the candidate requirements.
3. Assessing the relative cost of implementing each candidate requirement.
4. Calculating the relative value and implementation cost of each requirement candidate and plot each on a cost-value diagram.
5. Using the cost-value diagram as a map for analyzing the candidate requirements.

One of the major advantages of the AHP is that it is reliable because it computes consistency ratio across requirements to enhance clarity [15]. Another benefit of the AHP is that the resulting priorities are relative and this provides useful assessments of requirements [42]. However, one of the major drawbacks of this technique is that it is not reliable in environments with multiple stakeholders. Hence, the use of AHP seems not feasible with large number of requirements. In addition, AHP is very time consuming. It has also been observed that AHP contains a huge amount of redundancy [43].

4.1. 3.3 Cost-value approach

This approach was introduced by Karlsson and Ryan [25]. This approach uses the AHP technique to compare requirements in pair-wise manner based on the relative values and cost of implementing the requirements. However, studies have shown that the Cost-Value approach is also time consuming [15].

4.1.3.4 Wiegers' Matrix Approach

This technique uses a simple spreadsheet model to estimate the relative priorities for a set of software product feature. This technique is based on weighted assessments of perceived value, relative penalty, anticipated cost, and technical risks. The fundamental difficulty with Wiegers' matrix approach is that the value assigned to a given requirement does not necessarily determine if the requirement meets key business core values.

4.1. 3.5 Hierarchical AHP

The Hierarchy AHP was developed by Karlsson et al. [22] as a refinement of the AHP method. The hierarchical AHP structures requirements in a hierarchy and uses the AHP method to prioritize requirements at the same level of hierarchy. The hierarchical AHP reduces the number of requirements compared in a pair-wise manned as against the AHP method. Hence, redundancy is reduced.

4.1. 3.6 Value Oriented Prioritization (VOP)

VOP prioritizes requirements based on business values and the relative relationships among those values. This method is not suitable for large projects.

4.1.4 Machine Learning Techniques

The machine learning techniques use weighting scales or linguistic weights to prioritize requirements based on predefined criteria using learning algorithms [15]. Typical machine learning techniques include fuzzy AHP and Evolve [15].

4.1. 4.2 Fuzzy AHP

The Fuzzy AHP method involves the use of normalized triangular fuzzy weights to rank requirements [44]. This technique is however not scalable and it does not give room for requirements interdependencies [45].

4.1. 4. 2 Evolve

Evolve is based on generic algorithm in numerous iterations but has its roots in AHP. In the Evolve prioritization technique, a prioritization matrix is constructed from the weights of the business values of the stakeholders. The major benefits of the Evolve technique are its ability to allow late changes in requirements and changes in the weights assigned to stakeholders [31]. Evolve does not support requirements evolution or rank update [46].

* E-mail address: irojuolaronke@gmail.com.

4.1. 5 Hybridized Techniques

These techniques combine the features of the nominal scale, ordinal scale and the ratio scale methods. Requirement prioritization methods under this category include the Planning Game method and requirement triage [7].

4.1. 5.1 Planning game

This method is based on extreme programming [47]. The planning game combines the numerical assignment technique and ranking technique to perform requirements prioritization. Requirements according to Beck [48] are first prioritized into three groups which include those without which the system will not function, those that are less essential but provide significant business value, and those that would be nice to have. After assigning the requirements into the three groups, the programmer estimates how long time each requirement would take to implement and then begin to sort the requirements into three different groups. This method does not scale well with large number of requirements.

4.1. 5.2 Requirements triage

The word triage has its origin in the medical field. The word triage was used to classify patients based on how much benefit they derive from medical treatment. For instance, during a triage, disaster victims were categorized into three groups; those that would only recover if they receive medical attention, those that would recover regardless of treatment, and those with no hope of recovery. Hence, requirements triage is aimed at establishing the relative priorities for requirements by estimating the resources necessary to satisfy each requirement [34]. At this point, the stakeholders determine the requirements that must be included and those that must be excluded. This process leaves a set of requirements which the stakeholders can choose from. Thereafter, requirements that are more important are prioritized by AHP, simple ranking or $100 method.

4.2 Requirement Negotiation

Requirements negotiation focuses on the assignment of priorities to requirements through the consensus of the stakeholders. Typical examples of requirement negotiation include winwin method, multi-voting system and ping pong balls.

4.2.1 Winwin

In the winwin technique, the stakeholders agree on the requirements that are of utmost importance to them. The major difficulty of this technique is that the stakeholders might have difficulty in agreeing on the requirements to be implemented [14]

4.2.2 Multi voting system

This technique allows stakeholders to vote for the requirements that are of utmost importance to them. The requirements with the highest number of votes are considered the most important.

4.3.3. Ping Pong Balls

In the ping pong balls technique, stakeholders are given ping pong balls which represent each of the requirements. Stakeholders are then asked to choose from the ping pong balls. The requirement with the highest priority is determined by the highest number of ping pong balls chosen. This method is not effective when large number of stakeholders is involved.

4.3.4. Dot Voting

In dot voting, stakeholders are required to rank requirements with sticky dots. Requirements with higher dots are considered more important. Again, this technique is not suitable for larger number of stakeholders and requirements.

5.0 A COMPARATIVE ANALYSIS OF REQUIREMENT PRIORITIZATION TECHNIQUES

This section reviews requirement prioritization techniques based on the following measures; scalability and reliability with multiple stakeholders and requirements, time consumed, ease of use and consistency check. These measures are employed in this study in order to assist stakeholders

* E-mail address: irojuolaronke@gmail.com.

make decisions during requirement prioritization. The justifications for these measures are highlighted below:

5.1 Scalability/ Ability to Handle Large Number of Stakeholders and Requirements

Software development process is usually a collaborative effort of diverse stakeholders such as users, designers, software architects and the coders. However, the number of stakeholders as well as their requirements may increase during a software development life cycle. Hence, there is a need for a software requirement prioritization technique that will be able to handle large number of stakeholders and their requirements.

5.2 Time

The development of software systems is required to be very fast. This is because the software industry is evolving fast and thus a variety of a particular software is easily available off-the-shelf. It is therefore necessary to complete a software development project within a stipulated time by using a software requirement prioritization technique that requires less time.

5.3 Ease of use and users' confidence

Software development projects are usually characterized by limited resources such as cost and time. Hence, the need for a requirement prioritization technique that is simple, effective, easy to use as well as boosts the confidence of users.

5.4 Consistency check

Consistency check is a measure that is used to show consistency in the judgment of decision makers since human judgment can be inconsistent.

Table 2 shows a comparative analysis of requirement prioritization techniques based on the aforementioned measures.

Table 2. A Comparison of Requirement Prioritization Techniques

	Requirement Prioritization Technique	Scale of Measurement	Scalability/ ability to handle large number of requirements	Reliability with multiple stake holders	time consumed	ease of use	users' confidence	consistency check
1	Numerical Assignment	Nominal	low	low	low	high	high	none
2	Moscow	Nominal	low	low	low	high	high	none
3	Top-Ten Requirement	Nominal	low	low	low	high	high	none
4	Simple Ranking Method	Ordinal	low	low	low	high	high	none
5	Bubble sort	Ordinal	low	low	low	high	high	none
6	Binary Search Tree	Ordinal	low	low	low	high	high	none
7	BTree	Ordinal	low	low	low	high	high	none
8	Hundred Dollar Method	Ratio	low	low	low	high	high	none
9	Cost-Value	Ratio	low	low	low	high	high	high
10	Weigers' Matrix	Ratio	low	low	low	high	high	none
11	AHP	Ratio	low	low	high	low	low	high

* E-mail address: irojuolaronke@gmail.com.

12	Hierarchical AHP	Ratio	low	low	high	low	low	high
13	VOP	Ratio	low	low	low	low	low	high
14	Fuzzy AHP	Machine learning	low	low	low	low	low	high
15	Evolve	Machine learning	low	low	low	low	low	none
16	Planning Game	norminal/ ordinal	low	low	high	high	high	none
17	Requirement Triage	norminal/ ratio	low	low	low	high	high	none
18	Winwin	negotiation	low	low	low	high	high	none
19	Multivoting	negotiation	low	low	low	high	high	none
20	Ping pong balls	negotiation	low	low	low	high	high	none
21	Dot voting	negotiation	low	low	low	high	high	none

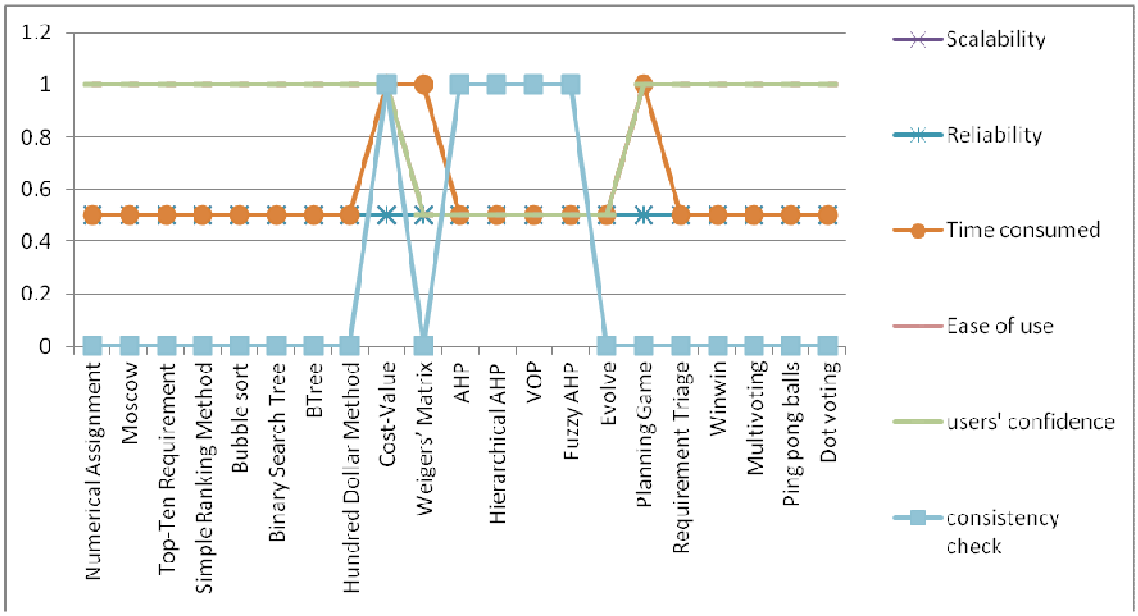

Fig.4. Characteristics of Requirement Prioritization Techniques

In Table 2 and Fig. 4, twenty one requirement prioritization techniques were compared. The study examined three nominal scale requirement prioritization methods. These methods include numerical assignment, top-ten requirements and Moscow. Table 2 shows that these methods are not scalable when multiple requirements are involved in requirement prioritization. In addition, these methods are not reliable when multiple stakeholders are involved in requirement prioritization. The time taken to prioritize requirements with these techniques is low. These methods are easy to use and users' confidence is high when these techniques are deployed for requirement prioritization. However, these methods lack the facility for consistency check. The study also examined four ordinal scale methods,

* E-mail address: irojuolaronke@gmail.com.

specifically, simple ranking method, bubble sort, binary tree and B-Tree. These methods are also not scalable when diverse requirements are involved in requirement prioritization; they are not reliable when multiple stakeholders are involved in requirement prioritization and they do not consume so much time during requirement prioritization. The findings from the study also revealed that these techniques are easy to use and the users' confidence is high. AHP, hierarchical AHP, Weigers' Matrix, cost value and VOP were the requirement prioritization techniques studied based on ratio scale. The study revealed that these methods are not scalable when multiple requirements are involved in requirement prioritization; they are not reliable when multiple stakeholders are involved in requirement prioritization, they are not easy to use and users' confidence is low. Cost value, Weigers' Matrix and VOP do not consume so much time during requirement prioritization, while the time taken to prioritize requirements with AHP and hierarchical AHP is high. However, the ratio based requirement prioritization techniques compared have the facility for consistency check. Furthermore, two hybridized requirement prioritization methods were studied. These include the planning game method and the requirement triage. These methods are also not scalable when multiple requirements are involved in requirement prioritization and they are not reliable with multiple stakeholders. However, the time taken to prioritize requirements with planning game techniques is high; while it takes less time to prioritize requirements with requirement triage. Both methods are easy to use and users' confidence with both methods is high. In addition, four requirement negotiation techniques were studied. These techniques include multi-voting, win-win, dot voting and ping-pong balls. The result of this study further revealed that these methods are not scalable when numerous requirements are involved, they are not reliable with several stakeholders, they require less time for requirement prioritization, they are easy to use and users' confidence is high when using these techniques for requirement prioritization. However, they lack the facility for consistency check.

It can therefore be deduced from Table 2 that the users' confidence and ease of use are directly proportional. Hence, the easier the requirement prioritization technique, the more confident the user is with the technique. Hence, user confidence is low for techniques such as Analytic Hierarchy Process, Fuzzy AHP and Cost-Value approach which are relatively difficult to use.

6. CONCLUSION

Requirement prioritization is one of the major activities carried out during requirement analysis. Requirements prioritization refers to the process of selecting the appropriate set of requirements from a multitude of competing and conflicting expectations elicited from various stakeholders involved in a software development project. There are several techniques used for prioritizing requirements. These prioritization techniques can be categorized as requirement prioritization method and requirement negotiation. The requirement prioritization method is based on five basic scales which include nominal scale, ordinal scale, ratio scale, machine learning and hybridized techniques. The requirement negotiation involves techniques such as winwin, multi-voting system and ping pong balls. However, the reliability of these requirement prioritization techniques is usually a challenge when multiple requirements and stakeholders are involved during requirement prioritization. Hence, this study compared twenty one requirement prioritization techniques such as Numerical Assignment Technique, Simple Ranking Method, Hundred Dollar Method, Analytic Hierarchy Process, Fuzzy AHP, Cost-Value approach, bubble sort method, binary search tree, winwin, multi-voting system and the MosCow technique. The comparison was based on scalability, reliability with multiple stakeholders, time consumed, ease of use, users' confidence and consistency check. This was with a view of assisting stakeholders to determine which technique is best to prioritize software requirements. The study revealed that AHP and hierarchical AHP consume more time than other techniques during requirement prioritization. It can also be deduced from the study that AHP, hierarchical AHP, Fuzzy AHP, VOP and Evolve are more difficult to use when compared with other techniques. Furthermore, of all the techniques compared, AHP, hierarchical AHP, cost value, fuzzy AHP and VOP have facilities for consistency check. The result of the study also showed that all the techniques compared are not scalable. Hence, it is difficult to prioritize requirements when multiple stakeholders and requirements are involved in the software development process. Hence, future researches should be based on the design of a requirement prioritization technique that will have the ability to accommodate large stakeholders and requirements.

* E-mail address: irojuolaronke@gmail.com.

REFERENCES

1. Taiwo OO, Awodele O, Kuyoro SO. A usability framework for electronic health records in Nigerian healthcare sector. International Journal of Computer Science Engineering. 2016; 5(1): 16-20.
2. Massey AK, Otto PN, Annie IA. Prioritizing legal requirements. Second International Workshop on Requirements Engineering and Law. 2009; 1-6.
3. Youngbae Y, Mincheol K. Analysis of user requirement on U-Healthcare system. International Journal of Business Tourism and Applied Sciences, 2013; 1 (2), 1-10.
4. Karlsson J, Ryan K. A Cost-Value approach for prioritizing requirements. IEEE Software. 1997;14(5), 67-74.
5. Firesmith D. Prioritizing requirements. Journal of Object Technology. 2004; 3(8), 35-47.
6. Berander P, Andrews A. Requirements Prioritization. In: A. Aurum and C. Wohlin. Engineering and Managing Software Requirements. Springer Verlag; 2005.
7. Ma Q. The *effectiveness of requirements* prioritization techniques for a medium to large number of requirements: a systematic literature review. M.Sc Dissertation, Auckland University of Technology. 2009.
8. IEEE-STD 610.12-1990. Standard glossary of software engineering terminology. Institute of Electrical and Electronics Engineers. 1990.
9. Saranya B and Subha R. International Journal of Computer Science and Business Informatics. 2014;12 (1), 32-44.
10. Gambo IP, Soriyan HA, Ikono RN. A proposed process model for requirements engineering using delphi techniques for prioritization. International Journal of Information Technology and Computer Science. 2015; 01, 73-80.
11. Machado RJ, Ramos I, Fernandes J M. Specification of requirements models. In: A. Aurum and C. Wohlin, Engineering and managing software requirements, Springer Berlin Heidelberg , 2005.
12. Maiden N. User requirements and system requirements. IEEE Software, 2008; 25(2), 90 - 91.
13. Aggarwal K.K, Singh Y. Software Engineering. 3rd ed. New Age International Limited; 2005.
14. Avesani P, Bazzanella C, Perini A, Susi A. Facing scalability issues in requirements prioritization with machine learning techniques. Proceedings of 13th IEEE International Conference on Requirements Engineering. 2005; 297-305.
15. Achimugu P, Selamat A, Ibrahim R, Naz'ri Mahrin M. A systematic literature review of software requirements prioritization research. Information and Software Technology. 2014; 56, 568–585.
16. Hatton S. Early prioritization of goals. Advances in Conceptual Modeling – Foundations and Applications. 2007; 235-244).
17. Mursu A. Information systems development in developing countries. Risk management and sustainability analysis in Nigeria software companies. PhD Thesis University of Jyvaskyla. 2002.
18. Mustafa MA, Al-Bahar JF. Project risk assessment using the analytic hierarchy process. IEEE Transactions on Engineering Management. 1991; 38(1), 46-52.
19. Boban M, Pozgaj Z, Sertic H. Strategies for successful software development risk management. Journal of Management. 2003; 8 (2), 71-91.
20. Whitten L, Bentley DKC. Systems Analysis and Design Methods. Irwin: Mcgraw-Hill; 2001.
21. Dahlstedt Å, Persson A. Requirements interdependencies – moulding the state of research into a research agenda. Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '03). Universität Duisburg-Essen, Essen. 2003; 71-80.
22. Karlsson J, Wohlin C, Regnell B. An evaluation of methods for prioritizing software requirements. Information and Software Technology. 1998; 39(14-15), 939-947.
23. Kukreja, N. Decision theoretic requirements prioritization: a two-step approach for sliding towards value realization. Proceedings of the 2013 International Conference on Software Engineering. 2013; 1465–1467.
24. New dawn technologies. Beat the odds, making your IT projects a success. 2009. Accessed 29 March 2017. Available: http://www.newdawntech.com/webinar/beattheoddspdf.pdf
25. Karlsson L, Höst M, Regnell, B. Evaluating the practical use of different measurement scales in requirements prioritization. Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering. 2000; 326–335.
26. Wiegers KE, First things first: prioritizing requirements. Software Development. 1999; 7(9).
27. Sommerville I, Sawyer, P .Requirements engineering-A good practice guide. Lancester University, Wiley, 1997.

* E-mail address: irojuolaronke@gmail.com.

28. Brackett JW. Software requirements. Technical Report SEI-CM-19-1.2, Software Engineering Institute, Camegie Mellon University, USA, 1990.
29. Karlsson L, Host M, Regnell B. Evaluating the practical use of different measurement scales in requirements prioritisation. Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering. 2006; 326-335.
30. Aaron KM, Paul N, Anton AI, Prioritizing legal requirements. Second International Workshop on Requirements Engineering and Law, 2009 (relaw'09), IEEE. 2009; 27–32.
31. Aaqib I, Farhan M, Khan S, Khan A. A critical analysis of techniques for requirement prioritization and open research issues. International Journal of Reviews in Computing. 2009; 8-18.
32. Babar M, Ramza M, Ghayyur S., Challenges and future trends in software requirements prioritization. Computer Networks and Information Technology (ICCNIT). 2011; 319–324.
33. Berry DM. The importance of ignorance in requirements engineering. Journal of System Software. 1996; 28 (2): 179–184.
34. Mead NR. Requirements prioritization introduction. Software Engineering Institute Web Publication Carnegie Mellon University, Pittsburgh, 2006.
35. Ramzan M, Jaffar A, Shahid A. Value based intelligent requirement prioritization (VIRP): expert driven fuzzy logic based prioritization technique. International Journal of Innovative Computing. 2011; 7 (3): 1017–1038.
36. Aho, AV, Hopcroft JE, Ullman, JD. *Data structures and algorithms*. Reading, MA: Addison-Wesley;1983.
37. Thakurta R. A framework for prioritization of quality requirements for inclusion in a software project, Software Quality Journal. 2012; 21: 573–597.
38. Duan C, Laurent P, Cleland-Huang J, Kwiatkowski C.Towards automated requirements prioritization and triage. Requirement Engineering. 2009; 14 (2): 73–89.
39. Leffingwell D, Widrig D. Managing software requirements: a unified approach, Addison-Wesley Longman Inc., 2000.
40. Saaty TL. The Analytic Hierarchy Process. New York: McGraw-Hill, 1980.
41. Sadiq M, Ahmed J, Asim M, Suman Q, More on elicitation of software requirements and prioritization using AHP. International Conference On Data Storage and Data, Engineering, IEEE. 2010.
42. Iqbal M, Zaidi A, Murtaza S. A new requirements prioritization model for market driven products using AHP. International Conference on Data Storage and Data Engineering, IEEE. 2010.
43. Franceschini F, Rupil A, Rating scales and prioritization in QFD. International Journal of Quality Reliability and Management. 1999; 16 (1):85–97.
44. Raharjo H, Xie M, Brombacher A. Prioritizing quality characteristics in dynamic quality function deployment. International Journal of . Production Research. 2006: 44 (23):5005–5018.
45. Lima D, Freitas F, Campos G, Souza J. A fuzzy approach to requirements prioritization. Search Based Software Engineering, Springer, Berlin Heidelberg. 2011;. 64–69.
46. Greer D, Ruhe G. Software release planning: an evolutionary and iterative approach. Information and Software Technologies. 2004: 46 (4): 243–253.
47. Karlsson L, Thelin T, Regnell B, Berander P, Wohlin C, Pair-wise comparisons versus planning game partitioning-experiments on requirements prioritisation techniques. Empirical Softw. Eng. 2006; 12 (1) :3–33.
48. Beck K. Extreme programming explained. Reading, MA: Addison-Wesley; 2000.
49. Achimugu P, Selamat A, Ibrahim R. Using the fuzzy multi-criteria decision making approach for software requirements prioritization. Jurnal Teknologi. 2015: 77 (13): 21–28.
50. Azar J, Smith RK, Cordes D. Value-oriented requirements prioritization in a small development organization. IEEE Software. 2007; 32–73.

* E-mail address: irojuolaronke@gmail.com.