The Impact of Analytical Assessment of Requirements Prioritization Models: An Empirical Study

Aneesa Rida Asghar
Dept. of software engineering
Bahria University Islamabad, Pakistan

Atika Tabassum
Dept. of software engineering
Bahria University Islamabad, Pakistan

Dr. Shahid Nazir Bhatti
Department of Software Engineering
Bahria University Islamabad, Pakistan

Dr. S Asim Ali Shah
Dept. of Electrical Engineering
Bahria University Islamabad, Pakistan

***Abstract*—Requirements prioritization is one of the important parts of managing requirements in software development process which plays its role in the success or failure of a software product. A software product can go wrong or fail if right requirements are not prioritized at right time. Thus, there is a need of a vast or complete requirements prioritization technique or model that spans all the factors that must be considered while prioritizing requirements whether it's for a traditional software development or agile software development. There are several requirements prioritization methodologies that aid in decision making and in prioritizing requirements but importantly many lacks to account the important factors that have significant influence in prioritizing requirements. A requirement prioritization methodology that takes account of important factors such as time and human behavioral factors that has an influence in prioritizing requirements is required. This new model/ technique expected to overcome the lack that is in existing prioritization techniques because of not considering time gap factor and human behavioral factor. Extensive study on literature of agile methodology, requirements elicitation and prioritization has been done to find out factors that influence the decision making process of requirement prioritization. It is found that as agile methodologies such as XP, SCRUM deliver products in increments, there is a time gap between each increment of approximate 4 weeks or more, this time lapse could cause human behavioral to change either because of market demand or any other personal reason and, thus, influences the prioritization decision. These factors could be termed as time factor and human behavioral factors. Thus, a requirement prioritization technique or model is needed that takes account of all such factors while prioritizing requirements whether it's for a traditional software development or agile software development.**

Keywords*—*Agile Software Engineering (ASE); Agile Software Development (ASD); Scrum Software Development Process; SCRUM; Product Owner (PO); Extreme Programming (XP); Requirements Prioritization techniques; Analytical Hierarchy Process (AHP); Cummulative Voting (CV); Numerical Assignment (NAT)

I. INTRODUCTION

Managing requirements is one of most important aspect of software development system. Developing software is entirely based on requirements as it contains the functionality or quality that the customer or stakeholder/s needs. Requirements emerge throughout the development process of software and, thus, they are needed to be addressed properly through communication between stakeholders, developers and documentation. A lot of factors play their role in the success or failure of a software product such as eliciting right and unambiguous requirements, managing unrealistic requirements and focusing on quality requirements etc. Requirements prioritization is one of the important parts of managing requirements in software development process which plays its role in the success or failure of a software product. A software product can go wrong or fail if right requirements are not prioritized at right time. Thus, there is a need of a vast or complete requirements prioritization technique or model that spans all the factors that must be considered while prioritizing requirements whether it's for a traditional software development or agile software development. Agile methodology is an innovative and iterative process that is currently the most widely used methodology for software development around the world as it supports changing requirements and helps in addressing changes throughout the development process.

There are many existing requirements prioritization methodologies that aid in decision making and in prioritizing requirements but importantly many lacks to account the important factors that have significant influence in prioritizing requirements. A requirement prioritization methodology that takes account of important factors such as time and human behavioral factors that has an influence in prioritizing requirements is required. This new model/technique is expected to overcome the lack that is in existing prioritization techniques because of not considering time gap factor and human behavioral factor. Agile methodology such as XP, SCRUM delivers software products in increments; especially in case of SCRUM, since there are time gaps between sprints, human behavioral factor plays an important role here as the time passes and the requirements changes. In sprints, requirements will be prioritized both on the basis of influencing factors such as cost, value, risk, time to market etc. and through the effect of non-functional requirements over

functional requirements. This will improve the overall quality of software product when it is included in the development process of scrum or could at least reduce the wastage of time, effort and resources. Requirements will not only be prioritized based on sprints, human decision but by critically analyzing the factors (sub characteristics) that can cause the product to success/ fail repeatedly thus ensuring the consistency in right requirements and hence the right prioritized requirements will be selected for a particular sprint at a time.

Problems arise when new requirements evolve due to change in business needs, time to market, time and human behavioral factors during the development of a software product [4] [3] [1]. This is why the software market is moving towards an approach that supports changing requirements and managing them. As agile software development contains this attribute of managing changing requirements it is being widely used in software developments process worldwide where speedy development process is required. There are many factors involved in the success or failure of a product, one of them is collecting and prioritizing requirements while keeping influencing factors in mind [2]. After carefully eliciting requirements it is essential to arrange them so right requirements are delivered at right time in order to ensure the success of a product. There are many well-known existing requirements prioritization techniques but not one of them spans all the different types of software development projects. Some techniques work well with short projects and some with large projects, some with traditional development where extensive documentation is needed with no changes during the development process and some with agile development where little or no documentation is needed and where changes are welcomed. Although requirements can evolve at any stage during the development process, it is very unlikely to be able to handle the factors that could be the cause of new emerging requirements or the cause of changing in existing requirements. But what could be done is that a new method or techniques could be introduced that considers those factors which are expected to be the root cause of these changes which lead to the waste time and effort; and thus reduces the wastage of time, effort and resources or could at least minimize the damage.

Requirements are elicited at the beginning of every software development process and project (product) and later are prioritized according to their relative importance to the market and to the product itself by keeping several factors in mind that could affect the prioritization decision. Prioritizing right requirements at right time helps the software team to understand the existence and importance of a particular requirement, its importance of use and its urgency to time to market and many other factors. There are many existing requirements prioritization techniques with their relative strength and weaknesses depending on many aspects they consider while prioritizing requirements. However, many of them fail to take account all the factors that must be considered while prioritizing requirements such as cost, value, risk, time to market, number of requirements and effect of non-functional requirements on functional requirements, time constraints and human behavior factor.

One of the most popular methods among agile family where software is delivered in increments called sprints is known as SCRUM [8] [6]. A sprint consists of 2-4 week iteration. Scrum methodology comprises of a planning meeting and daily scrum meeting, the planning meeting is conducted at the beginning of every sprint. In this meeting team members determine the number of requirements they can oblige to manage that is they create a sprint backlog out of that. Sprint backlog contains the list of all the tasks that should be perform during a particular sprint. Daily scrum meetings are not more than 15 minutes, where product owner (PO) gets continuous updates about the development process and can provide feedback about the features being included. This way if a PO decides to add new feature to a sprint, he/she can discuss it with the development team and save time rather than reviewing it at the end and demanding change at the end. The team conducts a sprint review at the end of each sprint where they demonstrate new features and functionality to the PO or to other stakeholders that can provide any kind of feedback which could be beneficial or helpful in any way for the next sprint. This loop of feedbacks results in modifications to the recently delivered functionality, then again it is more likely reviewing or adding new requirements to the product backlog. Another activity in Scrum project management is Sprint retrospective. The Scrum Master, PO and the development team participates in this meeting. It is the chance to reproduce or review the sprint that has ended, and identify new ways to improve. Scrum consists of three artifacts, sprint backlogs, product backlogs and burn down charts. The Product backlog, prioritized by the PO is a complete list of the functionality (written as user stories) that is to be added to the product eventually. It is prioritized so that the team can always work on the most important, urgent and valuable features first. On the other hand, sprint backlog is the list of all those tasks that the team obliged to and needs to perform during the sprint in order to deliver the required functionality. The remaining amount of work either in a sprint or a release is shown by 'Burn down' charts. It is an effective tool to conclude whether a sprint or release is on schedule to have all planned work finished in time. The traditional requirements engineering is very time consuming and requires speedy process to timely meet the needs of market so modern software industry demands rapid and iterative process like agile development to cope with the changing requirements and time.

As XP, SCRUM and other agile methodologies allow engineers to handle changing requirements as they evolve; however, it is still a challenging task to comprehend which prerequisites are sufficiently vital to have high need and to be incorporated into early sprints because later on this decision could be influenced by other factors which particularly in case of SCRUM could be time gap and human behavioral factors. Organizing requirements into Priority requirements helps the project team to comprehend which requirements are most essential and most urgent to implement and execute. Prioritization is likewise a helpful activity for decision making in other phases of software engineering. Therefore there should be a well-managed requirement prioritization technique

included in scrum processes that minimizes changes later in the process and save time, effort and other resources.

II. Literature Review

In this work [1], author presented the 10 years progress of agile research and proposed some future research areas for agile researchers to hold on to an approach that is theoretical or hypothetical. A survey based methodology was used to get reliable information about the progress of agile methodologies. It is significant to remember that one can produce and enhance fields as a scientific discipline only if energies are able to convey a solid theoretic system to conduct research on agile development. Therefore, it is a need that in future when investigating into agile development proficient research areas, agile researchers hold on to a more theoretical based approach.

Ming Huo et al [3] proposed that agile methods can assure quality even agile methods are faster and have to manage changing requirements. Author basically presented a comparison between waterfall model and agile model and presented the results. Agile methods contain some practices that have QA abilities, so with the help of this quality can be achieved more appropriately through agile methods. However one thing that must be considered when documenting agile RE is that in complex software development processes, less documentation can bring some issues/ problems.

Lan Cao et al [4], presented an empirical study on agile RE practices. This study shows the difference between agile RE and traditional RE is an iterative finding approach. Developing clear and complete requirements specification is impossible in agile development. Because of such important differences a new set of agile RE practices had come into practices that are reported in this paper. The study participants recognized that the most important practice in RE is thorough communication between the developers and customers.

Numerous participants highlighted that the efficiency of this practice depends deeply & effectively on exhaustive communication and interaction between customers and developers. Risks such as incomplete requirements, ineffectively developed requirements or wrong requirements are possessed if high quality interaction lacks in any project.

In this work Pekka et al [6], proposed that there are different methods of agile process that needs the empirical evidences. Authors emphasized on the quality of methodology not the quantity. This approach was chosen for comparative analysis of these processes. Five perspectives are included in the analytical lenses. SDLC include the process aspect abstract principles vs. concrete guidance, empirical evidence, project management and universally predefined vs. situation appropriate. New directions are offered based on these 5 perspectives that focus on quality not on quantity of methods.

Amin et al [7], proposed that some lessons of RE must be considered by the agile methods if the most emphasized thing is quality. Some major aspects of RE that are not a much emphasized in agile are analysis (verification and validation), non-functional requirements and managing change. Author suggested that these practices of RE can be adopted in agile and high quality can be achieved. RE practices such as simplicity, continuing validation, short releases and frequent refactoring, can be implemented in the perspective of agile main ideas.

Deepti Mishra et al [8], proposed that agile process can be helpful for the development of complex software projects. Author supported his argument with the help of a case study. A medium enterprise (SME) that practiced agile methods, achieved many successful results. Starting a project with agile methods and then achieving optimum methods by tailoring agile methods according to vision and benefits is the main reason of the success of supply chain management. The architectural design of this large scale complex project was supported with formal documentation. In the successful completion of the project an important role was played by this design documentation.

Franek et al [11], proposed different ways of RE methods from which agile software development can get advantages. Some common and different features and attributes of traditional approaches and agile approaches are also discussed. Agile approaches such as XP involves feedback from development teams and customers, communication and simplicity. Similarly RE process also includes dictation, analysis and validation. But in agile process phases are not as clearly distinguished as in RE process and techniques can also vary. Overall both are pursing same objectives. The major difference is of documentation that is really important to communicate with the stakeholders.

V. N. Vithana [12], conducted a research using qualitative methods to find out which requirement engineering practices are mostly being used in SCRUM methodology when developing a software product offshore. In order to collect data different job holders from nine organizations were questioned. It was found that RE practices such as Customer Involvement, prototyping, test driven development and Interaction are the least practiced activities of Requirement Engineering, although most of the team members were successfully practicing iterative requirements engineering, face to face communication, managing requirements change and requirements prioritization of SCRUM RE practice.

In this Anna Perini et al [14], proposed a strategy called Case-Based Ranking (CBRank). This method joins the preferences of the stakeholders of the project with the approximation of requirements ordering that is computed over machine learning methods. On simulated data the properties of CBRank are performed and then matched with a method called state-of-the-art prioritization, thus provided empirical results. However there are some assumptions in the CBRank prioritization process such as arbitrary selection as pair sampling policy and the monotonicity of the elicitation process. To improve the efficiency of real complex sitting methods the authors intend to work in future on non-monotonic formal logic case and pair sampling strategies that are more refined.

DAN HAO et al, [10] in this article, have presented a strategy that comprises the total and additional strategies for unified test case prioritization. These tactics prioritize test cases in light of components secured per test case, the aggregate number of program segments (or code-related) and the number of others (not yet covered) program segments (or code-related) components covered per test case, respectively. The proposed

approach includes basic and extended models, which define a spectrum of test case prioritization from a purely total to a purely additional technique by specifying the value of a parameter referred to as the fp value [10].

Rahul Thakurta [15], proposed a quantitative structure that determines the priority of a list of non-functional requirements. This framework involves members from business organization and the project to provide a measurable ground for assessing the level of value addition that is considered while choosing a new non-functional requirement to the project's requirement set. However, the inputs provided to the framework by members were subjective which may result in non-optimal results. Additionally, as the requirements assessment process involves stakeholders from both business organizations and the project, there are odds of irreconcilable interests and priorities of requirements. The author has also set the directions for future work which is to build a heuristic to bind the number of stakeholders to be preferred for assessment process.

Naila Sharif et al [16], devised a new requirements prioritization technique called FuzzyHCV which is a hybrid of two domains (SE and Computational Intelligence). It is a fusion of two methodologies which are Hierarchical Cumulative Voting (HCV) and Fuzzy Expert System. In FuzzyHCV, rather than a single crisp value a triangular fuzzy number is used. The proposed technique has been applied on 3 case studies and the results obtained are very close to the results of actual prioritization used in all of the three case studies. It is found that FuzzyHCV produces more precise results than HCV by comparing them with actual results for the chosen datasets. Authors intend to carry on work in this area by using fuzzyHCV for other domains problem such as decision making problems in employee selection and by incorporating fuzzyHCV to already existing decision making or requirements prioritization techniques so that less risky choices are made in future.

Mukhtar A. Abo Elsood et al, 2014 [10] in this research paper conducted a survey on the most popular requirements prioritization techniques being used and their reported drawbacks. The authors have devised a goal-based requirements prioritization technique that is based on generating a relative weight for the requirements with respect to the identified goals by stakeholders after conducting the survey. This technique is expected to overcome requirements prioritization problems such as time consumption, scalability and complexity. This technique has been evaluated by a case study and has been compared with AHP; it has proven to be more effective than AHP. However this technique has only been compared with AHP and not others, which leaves the effectiveness of this technique into an unanswered question. The authors intend to solve problems of data vagueness and uncertainty by enhancing goal-based RP technique.

Mr. Seyed Ali Marjaie et al [11] stated in this paper that there are many factors in the requirements prioritization process which have not been observed carefully other than risk, cost and value; these factors have significant impact on prioritization result itself. A statistical method has been proposed by authors which is based on attributes such as elicitation, numeral assignment, and factor analysis. This method combines two or more attributes into a single factor thus reducing the number of attributes and tries to identify groups of inter-related attributes, to find out how they are related. This improves the stability of factors involved in prioritization process and also the existing prioritization techniques effectively. Attributes that have been selected as important attributes are cost, time, risk, reuse of code, complexity, desirability and frequency. However this proposed method has not yet been applied on any real time software project and the results are merely based on theoretical assumptions.

Nikita Garg et al, 2015 [12] in this research paper explained all the requirements elicitation and requirements prioritization techniques. The requirements prioritization techniques that have been discussed in this paper are Analytical Hierarchy Process, The 100 Dollar Test, Numerical Grouping, Ranking and Top-Ten requirements. The authors have explained why it is important to select right requirements elicitation and prioritization techniques when building software as it acts as a backbone for the project. The authors have explained how each type is suitable for a particular situation but have not compared any two techniques nor they have suggested any new or hybrid technique.

Muhammad Imran Babar et al [13] to overcome the limitations of existing software requirements prioritization techniques, the authors have proposed an extension in VIRP model. The proposed technique will be automated for better understanding by adding heuristics using Neural Network and thus the interpretation of important requirements will be better so that there would be less chances of error. It is expected to be more time efficient, scalable as well as high overall performance than other techniques. However the proposed technique has not yet been implemented and the results are just expected to be good when compared to other techniques, there is no validity of this proposed technique.

Richard Berntsson Svensson et al, 2011 [14] in this paper found out that the dominant method that is being used in different companies developing software intensive systems are ad-hoc prioritization and priority grouping of requirements. The authors conducted a survey in 11 successful software companies. They also found out that customer input was being used as criteria for prioritization but not all the time. The results also suggested that functional requirements were given more importance than the quality requirements. The non-functional requirements (quality requirements) are prioritized if only time and resources are still available after implementing functional Requirements.

M. Waseem Asghar et al, 2013 [15] in this paper devised a tool called SWTMetrics. Using artifacts traceability information this method prioritizes changing requirements. A set of code-based metrics is also being used to locate requirements implementation as well as it measures several properties of requirements being implemented such as size, coupling, scattering. Authors have applied the proposed tool on three java applications and the results achieved are considerably different than those defined by experts but not entirely. This diversity is because of analyst selecting those requirements that are weakly related to main functionality

(provided by the application) with respect to SWTMetrics. Hence, the tool determines the ordering of requirements based on how these are implemented in a subject software system but its effectiveness has not yet been confirmed by the importance of being applicable in the software industry and providing some promising improved results.

Muhammad Aasem et al [8] proposed a framework in this research paper that combines existing approaches and techniques to help software engineer in performing prioritization. The proposed framework has α, β, and γ processes where the first two processes α, β, are subjective and require human involvement; they (α, β) include 100-dolors test prioritization method. Whereas, because of the algorithmic nature of the third process γ; it can be made fully automated as by using AHP technique it can automatically perform pair wise comparisons when the outputs produced by process-α and process-β are in the form of batches of n size each and Ranked Criteria respectively. Release scheduler sub-process of process-γ is executed (which is based on Numerical Assessment) after mapping all requirements into B-Tree. Thus a series of releases of prioritized requirements is obtained. This framework is expected to be effective but has not yet been tested on real scenarios. Feasibility of processes α and β for semi or full automation should also be checked.

Nupul Kukreja et al [17], in this have proposed a prioritization methodology to prioritize requirements of system and software. This methodology is a two-step approach and is based on decision theoretic model using a prioritization algorithm called TOPSIS viz. In the proposed approach [13], initially, the system is fragmented into high-level Minimal Marketable Features (MMFs). The proposed methodology allows measuring the effect of fluctuating business priorities on individual requirements without much overhead. This methodology also authorizes stakeholders to perform numerous analyses which also help in accurately judging the impact of fluctuating business priorities on individual requirements.

Here authors have also presented a validation report of this methodology by implemented this with 24 project teams of students at the Software Engineering project course in the University of Southern California. Although this approach has some drawbacks that need to be tackled in future; such as, one of the drawbacks of TOPSIS is reversal of ranks i.e. the original order of requirements prioritization may change if irrelevant requirements are entered into the prioritization. This limitation was not considered while implementing the approach as the teams were result oriented therefore they resisted in adding irrelevant requirements for prioritization. Another drawback is that the ordered prioritization of requirements may not accurately reflect the anticipated rank ordering of requirements

To overcome the drawbacks of TOPSIS, several other prioritization algorithms could be used instead of TOPSIS viz such as Cost of Delay, Simple Additive Weighting or Weigers' Prioritization. Also one can simply record items to eliminate the overhead of winbook's incapability to record the items.

III. Related Work

Missing or poorly specified quality requirements can lead to project failure or huge loss. Eliciting quality requirements effectively is a difficult task altogether especially in SCRUM where one person i.e. the product owner [PO] has to make the list of all the requirements to be included in the project. It can be a hectic and difficult task. As 'Quality' requirements drive the architecture of software-intensive systems, they are more important than the functional requirements. Thus the success or failure of mission critical systems depends on how well the quality requirements are engineered and implemented. Prioritizing requirements is also another challenging task while developing a software product. Product Owner's commonly use following backlog prioritization techniques: Kano analysis, Moscow and Relative weighting (Karl wieger) [3] [8].

A. Analytical Hierarchy Process

In AHP the priorities of requirements is calculated to estimate their relative importance by comparing all unique pairs of requirements. In other words, the individual performing the comparison will decide manually which requirement has more significant, and to what extent using a scale 1-9.[14] AHP provides better results than any other tested methods as it is a ratio scale methodology, and also includes a consistency check.

Steps involved in AHP are:

1) Make an v×v matrix (v represents the number of requirements) requirements are latter inserted in rows and columns of the matrix.

2) For each pair of requirements, insert their relative intensity of importance (where the row of X meets the column Y). At the same point, insert the reciprocal values to the transposed positions (e.g. if cell XY=4 then cell YX=1/4)

3) Now, calculate the eigenvalues of this matrix to get the relative priority of each requirement. The final result will be the relative priorities of the requirements.

Total no. of comparisons that AHP requires is $v\times(v-1)/2$. Redundancy is produced in pair-wise comparisons in AHP, therefore AHP also calculates the consistency ratio to check the accuracy of the comparisons [14].

B. Cumulative Voting (CV)

CV is a ratio-scale requirements prioritization technique where the customers/stakeholders are given a fixed number of 'units' which are used for prioritization of requirements by giving vote to the requirements that the customers/stakeholders think are important or delivers the highest functionality. Another important feature of CV is the 'weightage'. For example there are 3 stakeholders then; Stakeholder with highest authority/share is given the highest weight (e.g 10) and the stakeholders 2 and 3 with lower shares are given lower weights e.g. 7 and 5 respectively. Their weight is multiplied by the number of units the stakeholders assigns to requirements. In this way if stakeholder 2 and 3 vote for a particular requirement say 'reqA' which stakeholder 1 does not vote for

and instead votes for 'reqB' then 'reqB' will be of high priority even though reqA got 2 votes and reqB got only 1 vote.

C. MoSCoW

MoSCoW stands for Must have, Should have, Could have and Would have requirements. It is based on human opinion, based on their experience, desire and influencing factors at that time such as market demand, cost, risk, time and resources. Must have requirements are critical to the current increment in order to be a success, these are time critical requirements. Should have requirements are important to be included in the product but are not necessarily important to add to the current increment and can be added on later increments, these are not time critical requirements. Could have requirements are nice to have in the products but they do not participate in the success or failure of the product but are still nice to have if included. These requirements could improve user experience or satisfaction. Won't have requirements are the ones that are least critical to time or success of the product and hence can be added later any time of the time and resources permit.

D. Numerical Assignment

In Numerical assignment numerous requirements are grouped into different priority groups such as high, medium and low priority groups. All the requirements in a particular group will have same priority. For example if there are 7 requirements in medium category then all these 7 requirements will have same priority for this group.

E. Bubble Sort

In bubble sort prioritization, two requirements are taken and then compared manually; if the person doing the comparison feels that 1st requirement should have higher priority than the other requirement then he/she swaps the priority and continues this process until all the requirements have been compared. The result will be a prioritized set of requirements.

F. Ranking

In simple ranking, requirements are ranked manually from 1 to n number. 1 being the highest priority rank and n (the last integral valued requirement) being the lowest priority rank.

G. Hundred Dollar Method

Hundred dollar method is sometimes considered as Cumulative voting, however it is different than CV in a sense that 'weight' is not assigned to stakeholders in hundred dollar test. Each stakeholder is expected to distribute 100 dollars to the set of requirements being considered; however one may distribute full 100 dollars to a single requirement if he/she feels it's the most important requirement but is being neglected.

H. Binary Search Tree

In binary search tree, each node represents a requirement. Each base node has two child nodes with lower priority requirements on left child node and higher on right child node. We take a random requirement and compare it to the root node. If that requirement has lower priority than the root node then it is compared to the left child node; and now if it has higher priority than this child node it is placed on the right side of this node, however if it has lower priority then it is placed to the left side or compared to the left side child; or placed to the left side of the root node of there is not already any child on the left side of root node, otherwise if it has higher priority than it is compared/placed on right side of the root node and so on. This process of comparing nodes (requirements) to the root node and so on is done until all the nodes have been placed in their right priority.

I. Five Whys

Often stakeholders want a certain requirement implemented which does not have any great function or quality aspect and does not even have founded on logical arguments or the business interests of the company but still keep on insisting on that particular requirement. In such case, the team members (engineers) ask 5 whys (repeatedly 5 times or less) to why this requirement is important enough for the stakeholder to be implemented until the importance of the requirement is either found or established. The answer found could either determine the priority/importance of the requirement or that it could be cancelled or postponed for later increments.

IV. Proposed Methodology

The proposed model is based on several techniques that are being used to prioritize requirements. However when combined, they are expected to give better results. The First step in this model is cumulative voting, in cumulative voting each stakeholder distributes a total of 100 points ($, euro or coins) on the requirements, the Product Owner then will sum up the points and present the derived ordering of the requirements. Although the desired features will be selected at this point but there could be the chance that the selected feature will not provide benefit in terms of cost, time or easiness as much as it could have provided with other features selected at this time. The second step is Numerical assignment of requirements; it's the most common technique for prioritizing requirements and is based on grouping requirements into different priority groups. For example group the requirements gathered from first steps into different groups based on their nature such as risk requirements, value requirements, and complex requirements etc. After this, requirements will be grouped based on influencing factors that could be effecting

these requirements in any way. For example R1 and Rn are risk requirements [11] (see fig 2 below) and they are in any way contradicting with other requirements at the moment that have also been selected to implement in the sprint. Fig.1 depicts the steps of the proposed methodology.

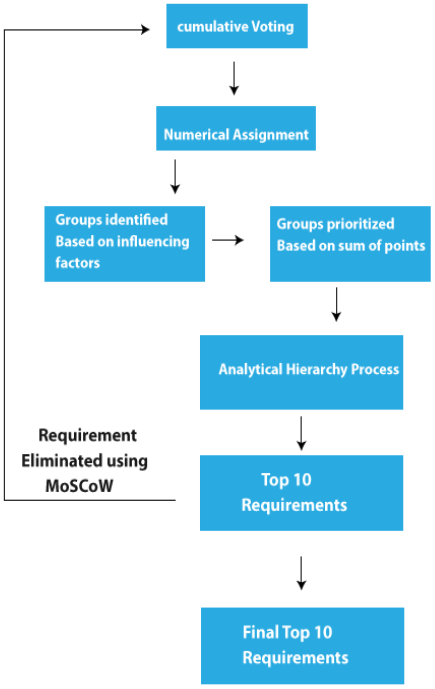

Fig. 1. Proposed hybrid model for requirements prioritization

This will cause trouble in implementing all of these requirements, therefore, it should be taken care of while selecting and prioritizing requirements for a sprint. Next, the groups will be prioritized based on highest points (see fig 2). Groups with requirements R1, R3, R4 have greater number of points as a whole then the other group therefore it has higher priority than other. After this the next step is AHP, in AHP the priorities of requirements is calculated to estimate their relative importance by comparing all unique pairs of requirements. In other words, the individual performing the comparison will decide manually which requirement has more significant, and to what extent using a scale 1-9.[14] AHP provides better results than any other tested methods as it is a ratio scale methodology, and also includes a consistency check.

Steps involved in AHP are:

1) Make an v×v matrix (v represents the number of requirements) requirements are latter inserted in rows and columns of the matrix.

2) For each pair of requirements, insert their relative intensity of importance (where the row of X meets the column Y). At the same point, insert the reciprocal values to the transposed positions (e.g. if cell XY=4 then cell YX=1/4)

3) Now, calculate the eigenvalues of this matrix to get the relative priority of each requirement. The final result will be the relative priorities of the requirements.

Total no. of comparisons that AHP requires is $v\times(v-1)/2$. Redundancy is produced in pair-wise comparisons in AHP, therefore AHP also calculates the consistency ratio to check the accuracy of the comparisons [14].

At this point when small number of requirements have been selected and grouped, it is best to apply AHP at this point as grouping the requirements based on their nature and influencing factors will make it easy to check requirements with other groups and find out their relative importance, or contradiction between them. As Agile development team and PO have best idea because of their experience in the field about the implementation of such requirements that are conflicting each other to some extend and/or the risk or cost while implementing them it is suggested to apply MoSCoW at this point. MoSCoW is based on human opinion based on their experience, desire and influencing factors at that time such as market demand, cost, risk, time and resources, the resultant selected requirements are then again filtered using MoSCoW, this is expected to filter out those requirements that may have gotten higher points during the 100 dollar test (cumulative voting) but are causing contradiction to other requirements or may be less beneficial to get them implemented in this sprint. New requirements from the backlog are added after such requirements have been filtered out. If the number of newly added requirements is greater than 3 or 4 then all the steps are repeated on those newly added requirements. If small number of requirements is being added then only MoSCoW should be applied.

Detailed diagram of the Proposed Model is presented below.

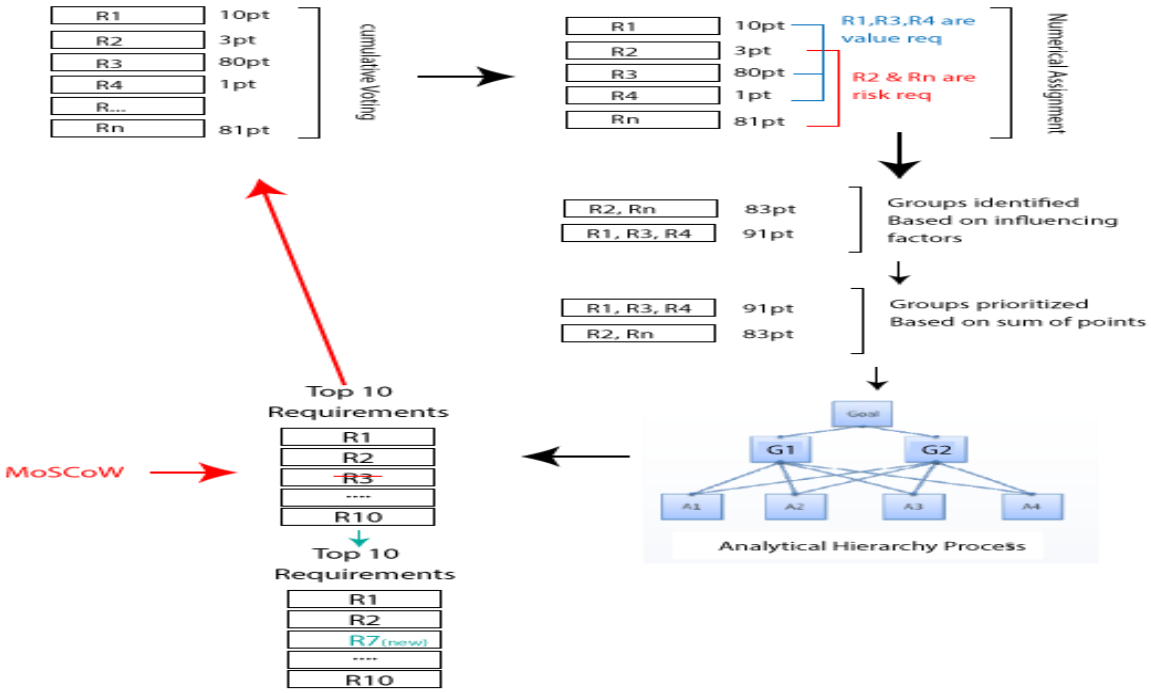

Fig. 2. Detailed presentation of the proposed model

V. Findings & Assessments

The detail findings and assessments in this are comprehensively discussed and are evaluated in this section. We have devised a requirement prioritization framework to be considered in scrum model that can provide softw are requirements engineers/ testers encouraging feedback regarding adopting appropriate prioritization (validation) approach at a particular stage of software requirements process, the future direction could be the complete automation of software requirements process.

TABLE I. Comparison of Most Used Techniques

Technique	Scale	Granularity	Sophistication	Aspect	Perspective
AHP	Ratio	Fine	Very Complex	Strategic Importance, Penalty	Product Manager
100-Dollars Test	Ratio	Fine	Complex	Customer importance	Customers
Ranking	Ordinal	Medium	Easy	Volatility	Requirements Specialist
Numerical Assignment	Ordinal	Coarse	Very Easy	Time, Risk	Project Manager, Requirements Specialist
Top 10	---	Extremely Coarse	Extremely Easy	Customer importance	Customers

TABLE II. COMPARISON OF EXISTING WORK RELATED TO REQUIREMENTS PRIORITIZATION

Author	Year	Contribution	Limitation
Balsam A. Mustafa	2014	Comparison between AHP, Cumulative Voting and Numerical Assignment is made	Less number of attributes such as time consumption, accuracy and ease of use is being considered and other important attributes have been left out such as cost, value, time to market, penalty, risk, volatility and other important attributes.
Naila Sharif	2014	FuzzyHCV; it is a hybrid of Hierarchical Cumulative Voting (HCV) and Fuzzy Expert System.	Relies largely on the initial segmentation of the vesselness image.
Falak Sher	2014	Comparison of existing techniques is conducted	Critical analysis of the existing techniques based on their support for prioritization aspects is not performed comprehensively.
Nupul Kukreja	2013	A two-step prioritization approach using a decision theoretic model to prioritize system and software requirements using a prioritization algorithm called TOPSIS viz.	Drawback of TOPSIS is rank reversals; The mathematical normalization is inherently biased towards 'lesser children'.
DAN HAO	2014	A unified test case prioritization approach that encompasses both the total and additional strategies.	This approach was more effective when applied to test cases at the test-method level than at the test-class level and when applied to Java programs with unit tests than to C programs with system tests.
Mukhtar A. Abo Elsood	2014	A goal-based requirements prioritization technique	This technique has only been compared with AHP and not others, which leaves the effectiveness of this technique into an unanswered question.
Mr. Seyed Ali Marjaie	2010	A statistical method based on attributes elicitation, numeral assignment, and factor analysis that reduce the number of attributes	This proposed method has not yet been applied on any real time software project and the results are merely based on theoretical assumptions
Muhammad Imran Babar	2007	An extension in VIRP model for requirements prioritization	The proposed technique has not yet been implemented; there is no validity of this proposed technique.

TABLE III. COMPARISON TABLE OF TECHNIQUES IN TERMS OF TECHNICAL AND BUSINESS ASPECTS

No.	Techniques	Technical Aspects						Business/Client Aspects			
		Citations	Scalability	Ease of Use	Time Complexity	Decision making	Accuracy	Sales	Marketing	Customer Satisfaction	Strategic
1	Analytic Hierarchy Process (AHP)	50		Yes		Yes	Yes				
2	Binary-Tree Prioritize	8		Yes			Yes				
3	Bubble Sort	8		Yes		Yes				Yes	
4	Cumulative voting (CV)	20		Yes	Yes		Yes				
5	Kano Analysis	5		Yes		Yes	Yes		Yes	Yes	
6	MoSCoW	6	Yes	Yes							Yes
7	Pair-wise analysis	10		Yes	Yes	Yes	Yes				
8	Numeral Assignment	15		Yes	Yes	Yes	Yes				
9	Ranking	8	Yes	Yes	Yes	Yes	Yes	Yes			
10	Relative weighting	2		Yes							
11	Top Ten Requirements	18		Yes	Yes	Yes	Yes				
12	Wieger's Prioritization	14	Yes	Yes	Yes	Yes	Yes			Yes	

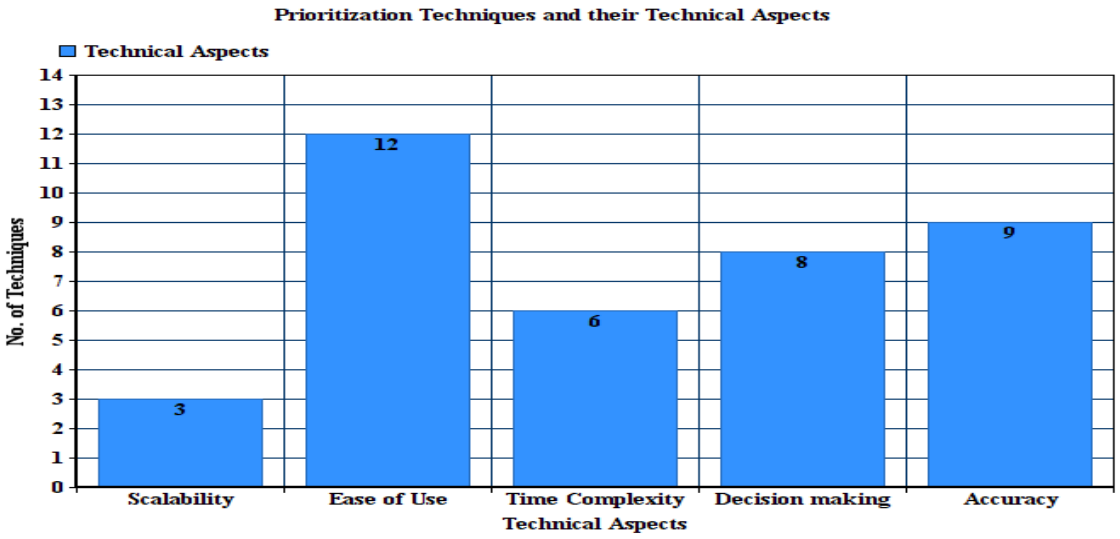

Fig. 3. A detail representation between requirements prioritization models and corresponding quality attributes (Technical Aspect)

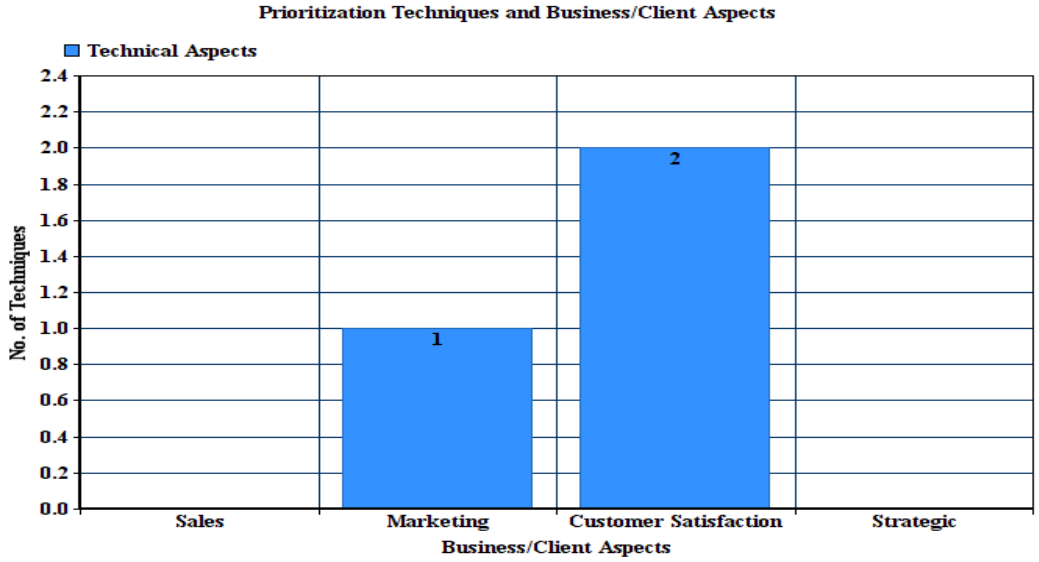

Fig. 4. A detail representation between requirements prioritization techniques and Business Client Aspect

VI. Systematic Methodology

Systematic literature review (SLR) has been done in order to discover new findings. Based on different research problems (mentioned in the literature) that are associated to both different prioritization aspects and techniques; we found the motivation for this research and hence the research questions are designed accordingly.

VII. Conclusion

As requirements emerge throughout the software development process and are needed to be prioritized and managed with highest priority, especially in the case of Agile Software Development process. As disused and highlighted in this research work, there are many requirements prioritization techniques, methodologies proposed and been followed but most of them fail to take account of all those factors that play an important role in prioritizing requirements and in overall quality of software product being developed. After a comprehensive literature, it is found that the existing prioritization techniques do not span over all type of projects. Some techniques work well on agile development process and some on traditional development. Therefore there is a need of a prioritization technique that considers the above mentioned factors (time factor and human behavioral factor) while prioritizing requirements.

References

[1] Elsevier (2012) A decade of agile methodologies: Towards explaining agile software development, The Journal of Systems and Software

[2] Mohd. Muqeem, Dr.Mohd.Rizwan, Validation of Requirement Elicitation Framework using Finite State Machine", IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp 1210 – 1216, 2014.

[3] Ming Huo, June Verner, Liming Zhu, Muhammad Ali Babar (2004) Software Quality and Agile Methods, IEEE

[4] Lan Cao, Balasubramaniam Ramesh (2008) Agile Requirements Engineering Practices:An Empirical Study, IEEE.

[5] Shahid Nazir, SEN-2005, Why Quality? ISO 9126 Software Quality Metrics (Functionality) Support by UML Suite, NY, USA.

[6] DOI= 1050849.1050860

[7] Pekka Abrahamssona, Juhani Warstab, Mikko T. Siponenb and Jussi Ronkainen (2003) New Directions on Agile Methods: A Comparative Analysis, IEEE.

[8] Armin Eberlein, Julio Cesar Sampaio do Prado Leite (2002) Agile Requirements Definition: A View from Requirements Engineering, Proceedings of the International Workshop on Requirement engineering.

[9] S. N. Bhatti, Deducing the complexity to quality of a system using UML. ACM SIGSOFT Software Engineering Notes 34(3): 1-7 (2009). DOI=1527202.1527207

[10] DAN HAO, LINGMING ZHANG, LU ZHANG, GREGG ROTHERMEL, HONG MEI, (2014) A Unified Test Case Prioritization Approach, ACM Transactions on Software Engineering and Methodology, Vol. 24, No. 2, Article 10, Pub. date: December 2014.

[11] Frauke Paetsch, Frauke Paetsch, Dr. Frank Maurer (2003) Requirements Engineering and Agile Software Development, IEEE

[12] V. N. Vithana (2015) Scrum Requirements Engineering Practices and Challenges in Offshore Software Development, International Journal of Computer Applications (0975 – 8887), Volume 116 – No. 22, April 2015.

[13] Azar, J.,Smith, R.K., "Value-Oriented Requirements Prioritization in a Small Development Organization", IEEE Computer society, 2007, pp 32 – 37, 2007.

[14] Anna Perini , Angelo Susi , Paolo Avesani (2013) A Machine Learning Approach to Software Requirements Prioritization, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 4, APRIL 2013

[15] Rahul Thakurta (2013) A framework for prioritization of quality requirements for inclusion in a software project, Software Quality Journal (2013) 21:573–597

[16] Naila Sharif, Kashif Zafar, Waqas Zyad (2014) Optimization of Requirement Prioritization using Computational Intelligence Technique, 2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE) Islamabad, Pakistan, April 22-24, 2014

[17] Nupul Kukreja, Barry Boehm (2013) Integrating Collaborative Requirements Negotiation and Prioritization Processes: A Match Made in Heaven, Proceedings of the 2013 International Conference on Software and System Process

[18] Rubaida Easmin, Alim Ul Gias, Shah Mostafa Khaled (2014) A Partial Order Assimilation Approach for Software Requirements Prioritization 3rd INTERNATIONAL CONFERENCE ON INFORMATICS, ELECTRONICS & VISION 2014

[19] Shahid N. Bhatti, Maria Usman, Amr A. Jadi, 2015, Validation to the Requirement Elicitation Framework via Metrics. ACM SIGSOFT Software Engineering Notes 40(5): 17, USA. DOI= 2815021.2815031

[20] J. Karlsson and K. Ryan. 1997, "Prioritizing requirements using a cost-value approach," IEEE Software 14 (5), pp. 67–74.

[21] John A Mcdermid, Software Engineer's Reference Book, Butterworth-Heinemann, 1991.

[22] Muhammad Ramzan, M. Arfan Jaffar and Arshad Ali Shahid (2011) VALUE BASED INTELLIGENT REQUIREMENT PRIORITIZATION (VIRP): EXPERT DRIVEN FUZZY LOGIC BASED PRIORITIZATION TECHNIQUE, International Journal of Innovative Computing, Information and Control, Volume 7, Number 3, March 2011.

[23] Mohd. Sadiq, Jawed Ahmed, Mohammad Asim, Aslam Qureshi , R. Suman (2010) More on Elicitation of Software Requirements and Prioritization using AHP, 2010 International Conference on Data Storage and Data Engineering

[24] M. Waseem Asghar, Alessandro Marchetto, and Angelo Susi Fondazione Bruno Kessler , Giuseppe Scanniello (2013) Maintainability-based Requirements Prioritization by using Artifacts Traceability and Code Metrics, 2013 17th European Conference on Software Maintenance and Reengineering

[25] Richard Berntsson Svensson, Tony Gorschek, Björn Regnell, Richard Torkar, Ali Shahrokn, Robert Feldt, Aybuke Aurum (2011) Prioritization of Quality Requirements: State of Practice in Eleven Companies, 2011 IEEE 19th International Requirements Engineering Conference

[26] Mukhtar A. Abo Elsood, Hesham A. Hefny , Eman S. Nasr (2014) A Goal-Based Technique for Requirements Prioritization, The 9th International Conference on INFOrmatics and Systems (INFOS2014) - 15-17 December Software Engineering - Challenges of Openness Track

[27] Nikita Garg , Dr. Pankaj Agarwal , Shadab Khan (2015) Recent Advancements in Requirement Elicitation and Prioritization Techniques, 2015 International Conference on Advances in Computer Engineering and Applications (ICACEA) IMS Engineering College, Ghaziabad, India

[28] Mr. Seyed Ali Marjaie , Mrs. Vasundhara Kulkarni, 'Recognition of Hidden Factors In Requirements Prioritization Using Factor Analysis', IEEE

[29] Muhammad Imran Babar, Muhammad Rarnzan, Shahbaz A. K. Ghayyur (2007) Challenges and Future Trends in Software Requirements Prioritization, IEEE

[30] Muhammad Aasem, Muhammad Ramzan and Arfan Jaffar, 'Analysis and optimization of software requirements prioritization techniques', IEEE