Role of Requirements Elicitation & Prioritization to Optimize Quality in Scrum Agile Development

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

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

Shahid Nazir Bhatti
Senior Assistant Professor,
Department of Software Engineering,
Bahria University Islamabad, Pakistan

Zainab Sultan, Rabiya Abbas
Department of Software Engineering,
Bahria University Islamabad, Pakistan

***Abstract*—One of most common aspect with traditional software development is managing requirements. As requirements emerge throughout the software development process and thus are needed to be addressed through proper communication and integration between stakeholders, developers and documentation. Agile methodology is an innovative and iterative process that supports changing requirements and helps in addressing changes throughout the development process.**

Requirements are elicited at the beginning of every software development process and project (product) and latter are prioritized according to their importance to the market and to the product itself. One of the most important and influencing steps while making a software product is requirements prioritization. Prioritizing requirements 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. There are many requirements prioritization techniques with their relative strength and weaknesses. Otherwise 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.

There are several requirements prioritization methodologies that aid in decision making but importantly many lacks to account the important factors that have significant influence in prioritizing requirements. A requirement prioritization methodology based on several factors such as time to market, cost, risk etc has been proposed. The proposed model is expected to overcome this lack. 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. 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.

***Keywords*—Agile Software Engineering (ASE); Agile Software Development (ASD); Scrum Software Development Process; SCRUM; Product Owner (PO)**

I. Introduction

Traditional requirement engineering does not support changing requirements and continuous communication with stakeholders therefore problems arise when new requirements are evolved due to change in business needs and time to market [4] [3] [1]. Thus software market is moving towards agile software development as it supports changing requirements and speedy process development. Agile practices are being acknowledged and are becoming popular day by day in the field of requirement engineering. 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.

There are many factors involved in the success or failure of a product, one of them is collecting and prioritizing requirements [2]. Requirements elicitation and prioritization is one the most challenging task during product development and it is very unlikely to be able to write down all the requirements at early stage, they evolve continuously throughout the development process and are needed to be addressed properly to meet the changing needs of market and time. As scrum is an agile methodology, therefore it allows 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. Organizing requirements into Priorities 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 improves its quality.

A requirement prioritization technique based on several factors such as time to market, cost, risk etc. has been proposed that will improve the quality of software product when it is included in the development process of scrum. The proposed model is expected to overcome the lack of quality of the prior models. 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. Requirements will not only be prioritized based on human decision but by critically analysing the factors that can cause the product to fail/success repeatedly thus ensuring the consistency in right requirements for a particular sprint at a time.

II. MATERIALS AND METHODS

A. *Agile Requirement Engineering*

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.

B. *Requirements Prioritization Techniques*

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 bound 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.

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. Framework for Requirements Prioritization

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].

IV. 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.

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.

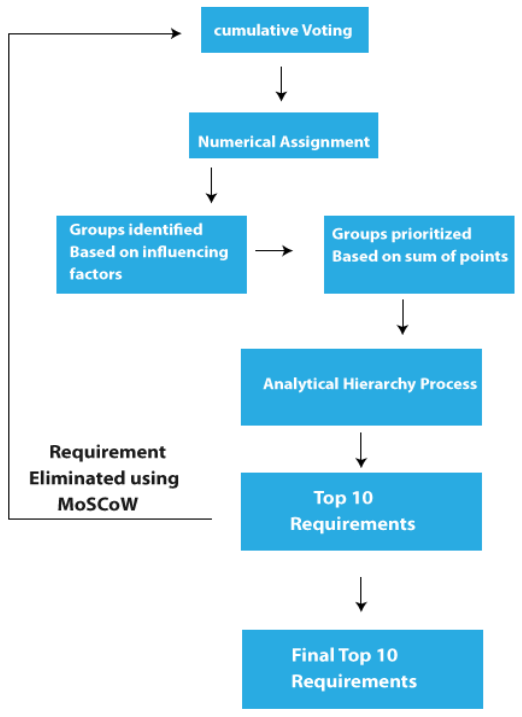
cumulative Voting
Numerical Assignment
Groups identified Based on influencing factors
Groups prioritized Based on sum of points
Analytical Hierarchy Process
Requirement Eliminated using MoSCoW
Top 10 Requirements
Final Top 10 Requirements

Fig. 1. Hybrid model for Requirements Prioritization

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.

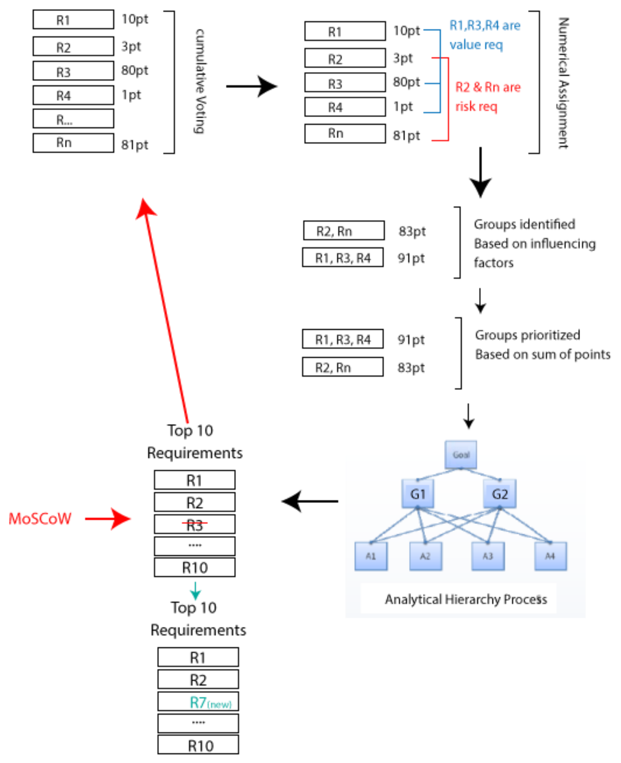

Fig. 2. Detailed presentation of the proposed model

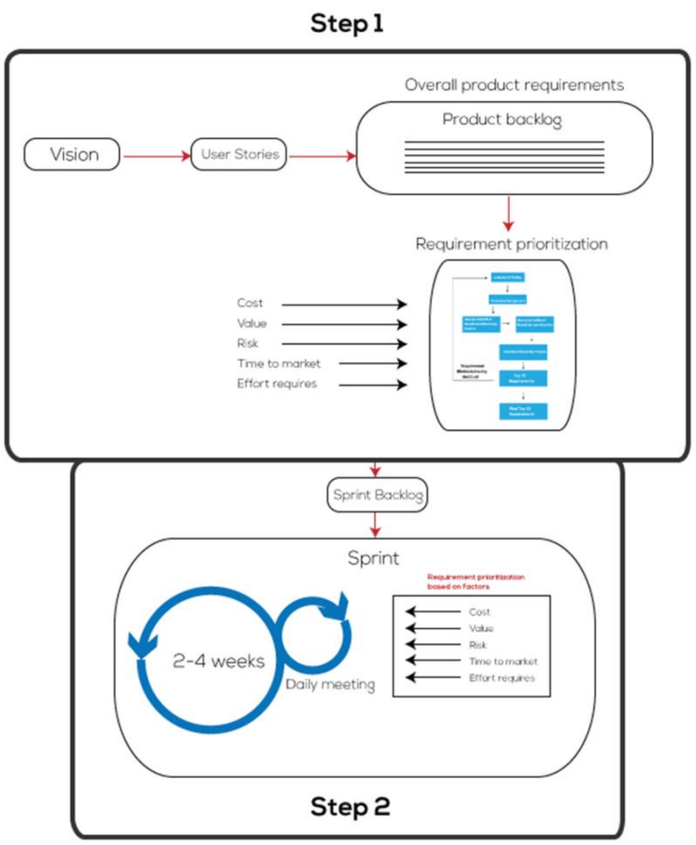

Fig. 3. Detail implementation (staging) in scrum

The validation of the proposed model is made via detail simulations using iThink software (see fig 4 below). The model is expected to increase the quality of the requirements being selected and prioritized during the sprints in SCRUM agile development. The simulation shows that as the number of requirements increases during the development process (see chart 1) and new requirements are added after filtering out requirements that were contradicting others, the priority of the requirements changes as new requirements were added after applying the selected techniques.(see chart 2) The results of these changes in requirements and prioritization shows that the quality of the selected requirements and prioritization increases (see chart 3) and is expected to give better results while implementing in SCRUM agile development.

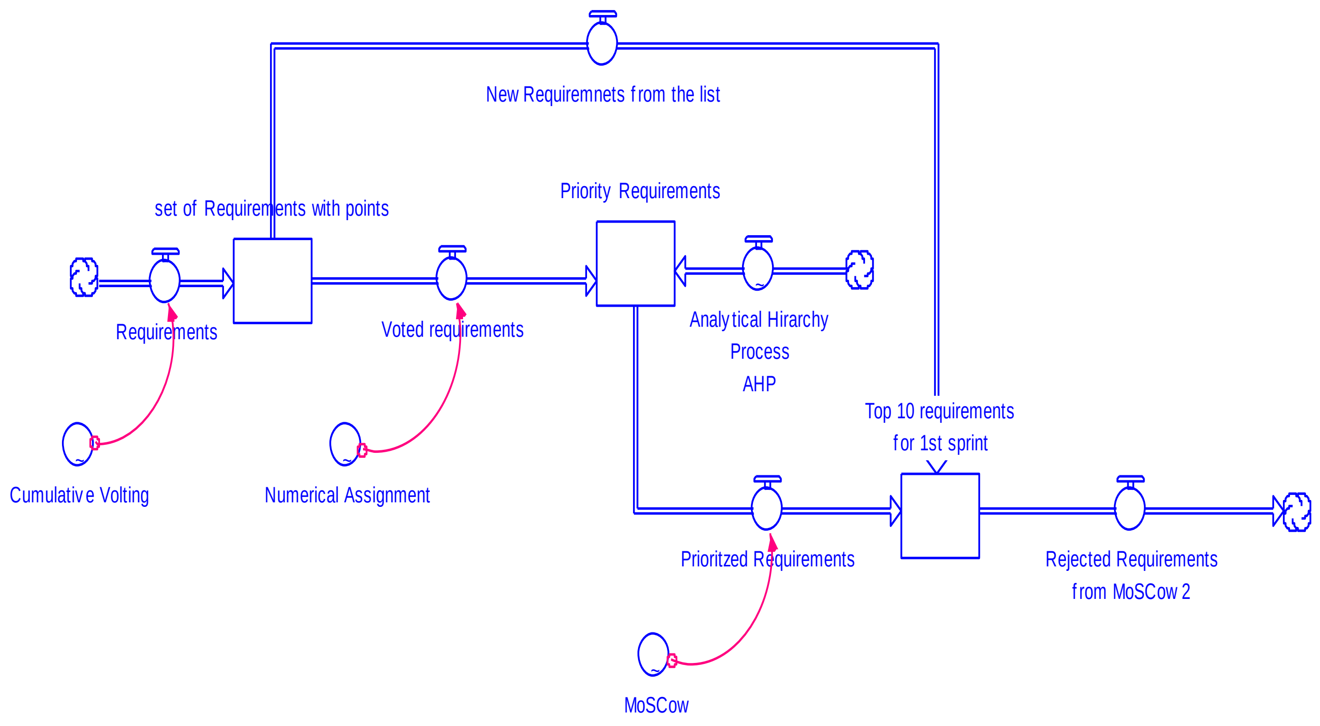
New Requiremnets from the list
set of Requirements with points
Priority Requirements
Requirements
Voted requirements
Analytical Hirarchy Process AHP
Top 10 requirements for 1st sprint
Cumulative Volting
Numerical Assignment
Prioritzed Requirements
Rejected Requirements from MoSCow 2
MoSCow

Fig. 4. Simulations of the proposed model in iThink

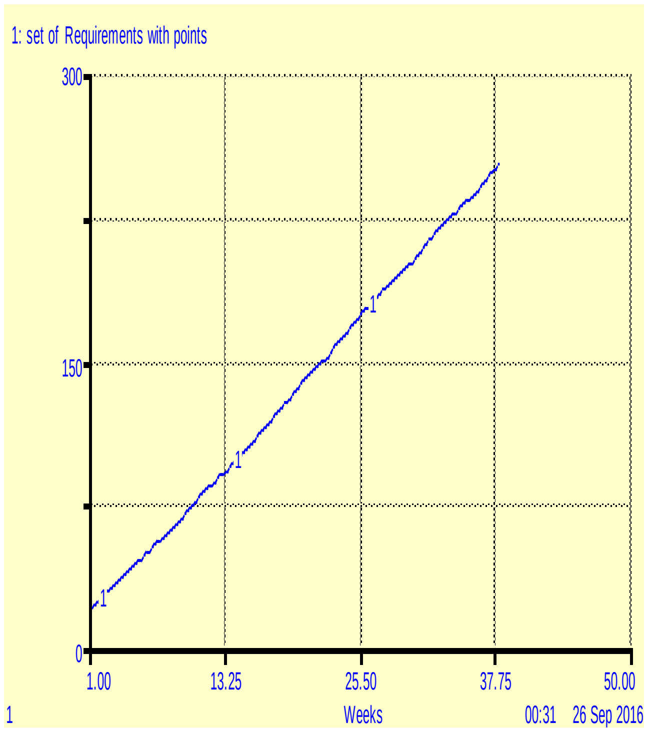

Fig. 5. Chart showing requirements upsurge during SDLC

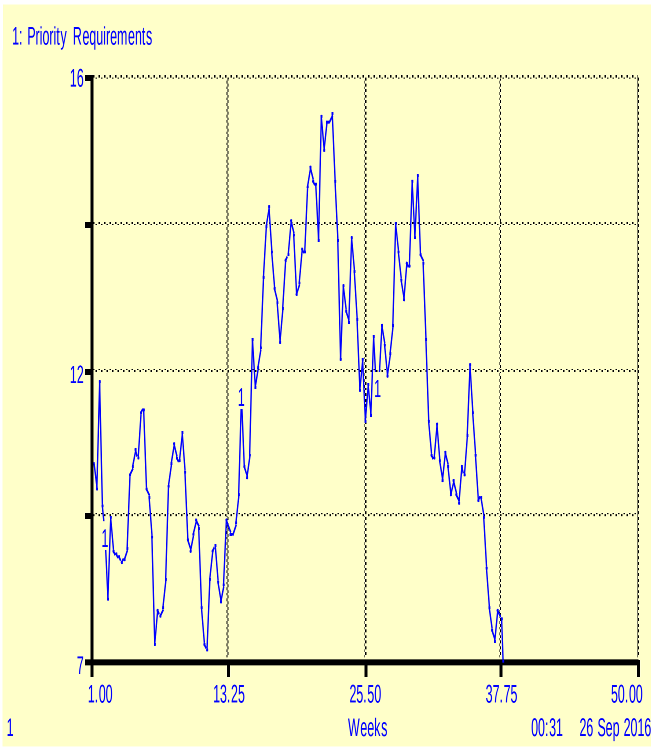

Fig. 6. Chart showing the priority of requirements change's as new requirements arrive

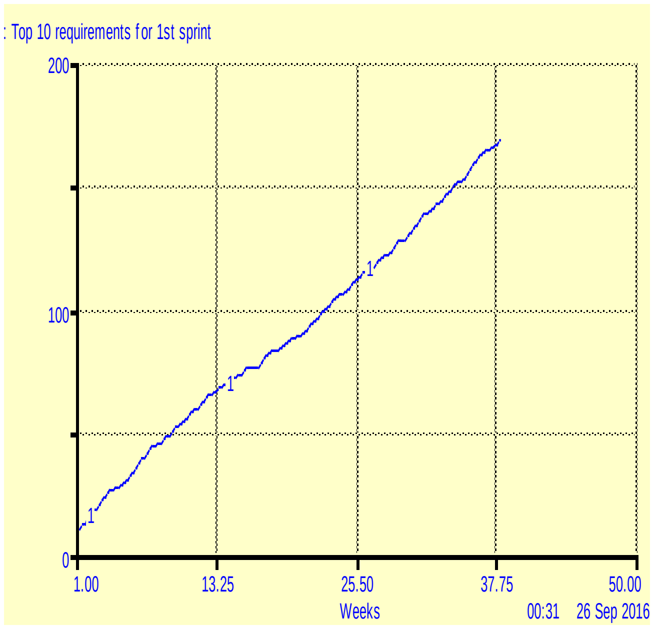

Fig. 7. Chart shows that the quality of requriements increases when applied to proposed model for a peculiar set of requirements

V. Conclusion

As requirements emerge throughout the software development process and requirements are needed to be prioritized and hence managed with highest priority, especially when the scenario is that 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 account those classical factors (metrics) that influence the overall quality of software product being developed for example ISO (9126, 25000) external metrics. In the following research work a methodology has been proposed in which we have taken account of the mentioned ISO/ IEC external metrics (i.e. 25000, 9126) which affect the quality of process as well as product. Further as it can be seen clearly that these mentioned metrics (attributes) increase the quality of requirement's being selected for the development of the product by considering all those aspects that has influence in prioritizing requirements, especially in case of ISO 25000. The proposed model here is a hybrid of other requirements prioritization techniques, it has the advantages of all the positive features already available of those ascribed techniques as mentioned and also have a consistency check that ensure that right requirements are being selected at the right time for the sprint under process in case of ASE (scrum).

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.

DOI=http://dl.acm.org/citation.cfm?doid=1050849.1050860

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

[7] 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.

[8] S. N. Bhatti, Deducing the complexity to quality of a system using UML. ACM SIGSOFT Software Engineering Notes 34(3): 1-7 (2009). DOI= http://dl.acm.org/citation.cfm?doid=1527202.1527207

[9] 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.

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

[11] 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.

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

[13] 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

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

[15] 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

[16] 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

[17] 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

[18] 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= http://dl.acm.org/citation.cfm?doid=2815021.2815031

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

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