International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 1 238 - 249

A Comprehensive Analysis of Literature A Comprehensive Analysis of Literature Reported Software Engineering Advancements Using AHP

Jogannagari Malla Reddy
Research Scholar, Dept. of CSE
Lingaya's University
Faridabad, Haryana State, India
Email: jmrsdpt06@gmail.com

S.V.A.V. Prasad
Prof & Dean (R & D)
Lingaya's University
Faridabad, Haryana State, India
Email: .prasad.svav@gmail.com

Samabasiva Rao Baragada
Head, Dept. of Computer Science
Government Degree College
Khairatabad, Hyderabad, India
Email: mrsambasivarao@yahoo.com

Abstract- The paper provides a various potential improvements in software engineering using analytic hierarchical processing (AHP). The presented work could support in assessing the selection of process, project, methods and tools depending on various situations encounter during software engineering. AHP belongs to Multi Criteria Decision making methods which seems to be a continuous research to solve critical and complex scientific and software engineering applications. This paper discusses existing key research contributions and their advancements in the areas of both software engineering and in combination of AHP with software engineering.

Keywords- *software engineering,AHP, metrics*

*****

I. HISTORY OF SOFTWARE

Software is viewed as a set of instructions which are stored in memory, produce required output when executed by the processor. Ada Lovelace is identified as the first software programmer during 19th century wrote a piece of software (algorithm) for the planned Analytical Engine. Her efforts have been supported by the theory of computation provided by Alan Turing [1]. This combination of computer science and software engineering gave rise to revolutionary developments in both the fields of software engineering and computer science. Computer science deals the problems theoretically and software works with problems in a practical manner. Prior to 1940's, since there is no stored program concept, electronic computing devices during this time were hardwired.

Claude Shannon has provided an outline of converting binary code into a software program. The process is a highly complicated, resulting in the computer programmers to work with a tedious mechanisms during loading the programs. After the developments of software slowly raised, software was sold to multiple customers as a free pack along with the hardware. This is because of no sophisticated provision of pre-installation of software with the hardware. This process gained popularity of software in commercial market. Based on the positive response, original equipment manufacturers like IBM etc, started selling software separately which began the age of commercial software. The paradigm shift also resulted in piracy [2]. Unix was an early operating system conceived by Ken Thompson which became widely popular and very influential, and still exists today. Mac OS is one of the most popular variant of Unix and Linux is a direct enhancement of Unix. BASIC is the early type-in program published in Dr Dobbs Journal in the year 1975. Young entrepreneurs like Bill Gates, Steve Jobs etc, capitalized their innovative ideas started capturing the market in a newer direction.

II. SOFTWARE ENGINEERING INCEPTION AND ITS SCOPE

The software engineering process has been started during 1940s and is still continuing in variety of newer directions. Presently software engineering is a profession concerned with the creation of a maximized quality of software. The term Quality is coupled with several parameters like stability, maintainability, testability, readability, size, cost, security, speed, usability. Measurable parameters like the total number of defects identified, user satisfaction and elegance are some other features which also impacts the quality of software product. Software engineering is also referred as the art of writing software is always a controversial problem among various experts which covering software design principles, so-called "best practices" for writing code. The management of team size, work culture, product in-time delivery procedure, out sourcing etc are some of the key practices included in the software design principles.

Software engineering became one of the bonafide profession by the early 1980s to stand beside computer science and traditional engineering [3]. During 1940's to 1960's men are recruited to work for conventional engineering and women are been delegated of writing software along with men. Grace Hopper, Jamie Fenton are the first decade women software engineers. The cost of software with respect to the hardware has been hiked substantially for the last 50 years. Every year almost one or two existing computers are becoming obsolete

and thus seed a provision to develop software for the new machines. The name software engineering is coined during the year 1965 by ACM president Anthony Oettinger [4][5] in his letter to address a conference at MIT.

The four key directions of Software engineering development include, aspects, agile, experimental and number of product lines. Aspects deal with the quality parameters to identify the bugs and removing them. Agile software engineering guidelines helps in development of software products which are comprised of drastic change in requirements. Another branch of software engineering is the experimental software engineering focused in conceptualization of theories, laws based on the experiments. Software product size is targeted in developing families of software systems.

Some of the early key contributors of software engineering are, Charles Bachman (Databases), Laszlo Belady (Operating systems), Fred Brooks (OS/360), Peter Chen (ER modeling), Edsger Dijkstra (frame work for proper programming), David Parnas (Information Hiding) and Michael A Jackson (JSP design).

III. KEY RESEARCH CONTRIBUTIONS IN SOFTWARE ENGINEERING

Software metrics are derived and estimated with the intent of evaluating certain characteristics of the software developed. Li and Cheung [6] have developed a Fortran static source code analyzer, (FORTRANAL), was developed to analyze 31 metrics available in the literature during the year 1987. They have also presented the analysis of a new hybrid metric presented in this work. A database comprised of 255 student assignments were taken as the sample data. The work has presented the inter-comparison of metrics and found the confirmation of internal consistency of some of these metrics which belong to the same class. The proposed hybrid metric has been developed which follows context sensitivity to the structural attributes obtained from flow graph to overcome the incompleteness limitation of existing metrics. The work revealed that many volume metrics have similar performance while some control metrics correlate well with typical volume metrics in the test samples which is unexpected by the authors. The proposed flexible class of hybrid metric has the ability to incorporate both volume and control attributes in assessing software complexity.

A technical report released by the William A Florac [7] of Software Engineering Institute (SEI) in the year 1992 has been taken as the prime framework for counting problems and defects in software products. The report has been prepared by huge number of software experts hailed from various prestigious organisations from all over the world. Authors proposed a framework that integrates and gives structure to the identification, reporting, and measurement of software bugs and defects found by the primary problem and defect finding activities. The framework identifies and organizes measurable parameters common to these activities. Authors show how to use the attributes with checklists and presented supporting forms for easier understanding of nomenclature.

The software Engineering Institute has developed a matured framework into Capability Maturity Model for software during the year 1997. A team of experts via. Paulk, Curtis, Chrissis, and Weber presented the SW-CMM [8] comprised of recommended practices in a number of key process areas that have been identified to enhance the software process quality. The CMM guides software organizations in gaining the control of their processes for developing and maintaining software. CMM also drives to toward a culture of software engineering and management excellence. Maturity level can be represented a plateau toward achieving a matured software process. The five levels of software maturity process are initial, repeatable, defined, managed and optimized. Level 1 being the capability is as a characteristic of the individual not of the organization. Establishing of appropriate management processes and policies reach level 2 for an organization. Level 3 is based on a common, organization-wide understanding of the activities, roles and responsibilities in a well defined software process. In level 4 the software process is measured and operates within pre-established quantitative limits. Level 5 eradicates the bugs in the software products and sometimes referred as the rework.

Software quality conforms with the degree of attainment of certain attributes to the developing software product. The nature of attributes and their combination in terms of assessment should be clearly defined. A methodology has been prepared by IEEE-SA standards board for establishing quality parameters and elicitation, implementation, analyzing, and validating the process and product software quality metrics is defined [9]. The methodology covers entire software life cycle.

Ron Burback [10] in the year 1998 presented a thesis to Stanford University, defining a novel software engineering methodology aka the WaterSluice methodology. The proposed WaterSluice borrowed both properties like the the iterative nature of the cyclical methodology and steady progressive nature of the sequential methodology. The tasks are prioritized in this methodology giving more advantageous by completing the non-conflicting tasks bit earlier. Several theorems are presented in the thesis to support the methodology for its strengths. The limitations are also verified by comparing this methodology with sequential and cyclical methodologies like paradigms (Noema), architecture (DADL), component composition (CHAIMS), and environments (DCE) available in

the literature. However, the methodology presented revised and enhanced software engineering practices..

Fenton and Neil [11] have carried a critical literature review of numerous software metrics and statistical models during their time. This review helps several software organizations in predicting the number of defects (faults) in software systems, prior to their deployment. Size and complexity metrics are the major parameters to estimate the bugs in the software product. Their review identified that the many empirical experimentations available in the literature are state-of-the-art. However, authors found that there are a number of serious theoretical and practical problems in many studies. This is because of the lack of proper correlation between defects and failures. Many prediction models tend to model only part of the underlying problem and ignore the prime components. Authors analyzed the Goldilock's conjecture to illustrate an optimum module size inherent in current defect prediction approaches, and found that conjecture lacks support and that some models are misleading. Finally the work recommended holistic models for software defect prediction, using Bayesian belief networks, as alternative approaches to the single-issue models available at the time. The theory of software decomposition is to be further studied and refined in order to test hypotheses about defect introduction and supports construction of a better science of software engineering.

Parallel to the survey conducted in the work [11], Kuilboer and Ashrafi [12] have attempted a survey in New England region to find out the impacts of using SPI techniques on product quality and productivity based on 67 collected responses. Authors have found an anecdotal evidence comprised of both supporting and opposing the benefits of SPI methodologies. The survey reveals that using SPI doesn't necessarily leads to quality software product at reduced cost or delivery time. The use of SPI methodologies just creates a perception of quality leading to end user satisfaction. The sample space is small and confined with a particular geographical extent cannot be drawn for generalization. However, such a kind of surveys are always needed to enhance the standards over the time.

Maria et al [13] have pursued a research to reevaluate the one of the first widely accepted software quality model presented by Boehm et al. [14]. The established hierarchy given by Boehm et al. [14] is organized into a framework of user priorities in the re-evaluatoin work of Maria et al. [13]. The main aim of [13] is to identify the important factors in software quality for different users. Authors have conducted a survey of software users comprised of technical and non-technical personnel working in corporate companies. The study identifies the mental or cognitive models of software quality held various professional groups.

Quality attributes can't be generic and can be seen as assumptions, constraints or goals of end users. Brito, Anna, and Araujo [15] presented a process to identify and specify quality attributes and to assemble them with functional requirements. Authors found that the crosscutting nature or ignoring of some of the quality attributes like reusability and traceability negatively influences with the overall quality. Authors proposed a template to specify quality attributes at the requirements stage in order to minimize the influence. The work presented use cases and sequence diagrams to specify the integration of those attributes with functional requirements to support the inferences.

Moody [16] presented the empirical evaluation of a set of proposed metrics for evaluating the quality of data models. The work comprised of a total of twenty nine candidate metrics which were originally proposed, measured each metric with a different aspect of quality of a data model. The research was leveraged to evaluate the usefulness of the metrics in 5 application development projects going on in two private sector companies. Three metrics survived the empirical validation process out of originally proposed ones, and the work discovered two new metrics. In total a set of 5 metrics are felt manageable by the participants in real-time use. Subjective ratings of quality and qualitative descriptions of quality issues seem to be much more useful than the metrics which is an unexpected finding of the work. The results of this study might indicate that it is not quite so useful in practice, but still the idea of using metrics to quantify the quality of data models perceive good in theory leaving a biased conclusion.

Souheil and Horgon [17] carried research work by identifying the metrics and measurement approaches that can be used in very large information systems which requires high degree of parallelism involving large and complex processing elements. The work investigated the factors which are relevant to the parallel class following the standard way used for sequential and parallel/distributed architectures. The approach presented in the work allows the specification of benchmarks against which achieved quality levels can be evaluated, and guides for building quality into software for parallel systems. With the use of Relationship chart and polarity profiles the feasibility of quality goals is controlled. Until unless the changes are not defined the system seems to be static but still, the approach is not static; if project stakeholder changes occur, or project requirements change, the Relationship Charts and Polarity Profiles can be updated in order to reflect these changes. Authors developed a set of formal guidelines for identifying the Essential Views, despite their importance to the proposed approach.

The process of improving the internal structure without changing its external structure in an object oriented system is said to be refactoring. Both restructuring and refactoring have the main intent of improving the quality interms of extensibility, modularity, reusability, complexity, maintainability, and efficiency. Mens and Tourwe [18] presented a paper which provides an extensive overview of existing research in the field of software refactoring in the year 2004. The work is compared and discussed with respect to a number of different criteria like, supporting refactoring activities, techniques and formalisms which are used to support these activities, software artifacts and their kinds that are being refactored, key issues which are considered while building refactoring tool support, and the impact of refactoring on the software process. A Document Class with three subclasses is taken as a running example to discuss and illustrate the proposed concepts. Authors indicated important open issues that remain to be solved in each of the 5 criteria. An urgent need for preparing formalisms, processes, methods and tools which address refactoring in a more consistent, generic, scalable and flexible way is identified by the authors. Research in the areas of software restructuring and refactoring continues to be very active, and is also to unleash and address the limitations of these tools.

Coupling and cohesion in an object-oriented system has a great scope in preserving an object-oriented system external quality. Aine Mitchell [19] presented a doctoral thesis titled *An empirical study of run-time coupling and Cohesion software metrics* to National University of Ireland Maynooth during the year 2005. The work is the largest empirical study that has been performed on the run-time analysis of Java programs. During the study she found that various proposed static coupling and cohesion metrics using empirical investigations available in the literature never considered the run-time properties of a program into account. Static metrics mostly fail to quantify all the underlying dimensions of coupling and cohesion, since the behavior of a program can be represented as a function of its operational environment and code complexity. Based on these influences, software designer need to acquire more comprehensive understanding of the quality of key components of a developing software system. Author believe that any measurement of such attributes should include changes that take place during run-time. Because of this reason, author using empirical evaluation of a selection of run-time measures for these properties has addressed the utility of run-time coupling and cohesion complexity. A comprehensive set of Java benchmark programs and some real-world programs are chosen for the study. Two case studies are included in the work. The first case study investigates the impact of instruction coverage on the relationship between static and run-time coupling metrics. The second case study establishes a new run-time coupling metric that can be used to study object behaviour and investigates the ability of measures of run-time cohesion to predict such behavioral nature. Author finally investigated the nature of run-time coupling metrics for being good predictors of software fault-proneness in comparison to standard coverage measures.

As mentioned in the work of [18], refactoring is to make software easier to understand and to improve software design. Still potential validations claiming the statement is not available in the literature. Bart Du Bois [20] presented a dissertation with an extensive contributions to such validation experiments. The work initially presented a validation of two existing reengineering patterns which are used in compressing the program. The work later discusses the results of formal analysis of the conditions in which known refactorings could improve coupling and cohesion as criteria for a good attempted object-oriented design. The results of the experiments confirm that the claimed benefits based on the selection of quality characteristics and improvement of internal design. The work finally opined that the integration of produced results in today's refactoring tools help a lot to the software maintainers in assessing which and where refactorings could be applied and ultimately reducing the human effort of transforming towards an optimal solution.

Girish, Jiang, and Clain [21] empirically studied the impact of the CMM on selected critical factors like software quality and project performance. Authors have concluded that a significant impact has been found on software quality and project performance during implementation. Implementation strategies like prototyping and developer commitment drives both software quality and project performance. Prior training to the developer effects only on software quality and the parameter simplicity effects on over project performance.

Radharaman and Juang [22] using defect density data have analyzed the quality of an ongoing software maintenance project before and after release changes. Their aim is to obtain the significance of certain factors like expertise of developer, change complexity and its proportional size with respect to the defect density of a particular change. Design and coding are the two major phases are studied in their work. Authors have developed regression equations to evaluate the impact of factors with respect to change complexity. Expertise of developer and change of requirements found to be significant with the project cost and schedule during design phase. However, no factor is found significant in coding phase. This could be because of high variability in software development. The work provides a conventional project manager to monitor the process performance and respond accordingly to any abnormalities.

Software development productivity management is a key component in software organizations. New strategies are always been investigated to cater the business demands for shorter time-to-market preserving high product quality force. Productivity modeling should focus on crucial and limited factors which has the most significant impact on productivity. Adam Trendowicz, Jürgen Münch [23] presented a comprehensive overview of productivity factors considered by software practitioners in recent times. The study has been carried by reviewing of 126 publications as well as international experiences of the Fraunhofer Institute, comprised of the most recent 13 industrial projects, 4 workshops, and 8 surveys on software productivity. The work opposes the traditional belief that software reuse is the key to productivity improvements. This situation can best be explained when a plethora of factors is considered in order to gain the expected benefits from reuse. A loss in productivity can be found if adhoc reuse is applied without any reasonable cost-benefit analysis and proper investments to create a reuse environment. The aggregated results of the work concluded that the productivity of software development processes still has a profound impact on the capabilities of software developers as well as on the tools and methods they use.

Forselius and Kakola [24] found that there is relatively little research on software Project Estimation and Measurement Systems (PEMS). The major limitation in the Commercial PEMS is that they vary in functionality and effectiveness. The users of these commercial PEMS users do not know what to expect from PEMS and how to evaluate them. Authors have presented a paper which creates an information system design product theory to overcome the limitations of PEMS by prescribing the meta-requirements, the meta-design, and applicable theories for all products within the class. Project estimation and measurement literature are been used to derive meta-requirements. The work also use past empirical experimental experiences carried by Finnish Software Measurement Association during last 10 years to derive the meta-requirements.

Understanding the attacker mindset is one the key requirement to preserve security. The identification of vulnerable code areas is always a challenging task and such a task should be thoroughly performed by the Security experts. Shin et. al [25] have investigated to find out the importance of software metrics obtained from source code and development history could be used as discriminative and predictive of vulnerable code locations. Authors categorized the investigating metrics into three categories: complexity, code churn, and developer activity metrics. Two empirical case studies were pursued using Mozilla Firefox web browser and the Red Hat Enterprise Linux kernel. Experimental results indicate that out of 28, 24 metrics are found to discriminative of vulnerabilities for both projects. The proposed models based on the three metrics classes have predicted over 80% known vulnerable files with false alarms less than 25% in both the projects. The work also proceeded by by considering a arbitrary selection of files for inspection and testing, these models found a reduced prediction but still with an accepted value of 71% with a false alarm rate of 28% in both the projects. The work majorly could be used by security experts as a prediction to prioritize security inspection and testing efforts.

The biggest challenge in software engineering is to deliver a customer satisfied product within the time deadline. Complex and large systems are now built within very short span of time and such systems need a good maintenance and to be improved according to the changing user requirements. Systems which are built in a short span of time tend to deteriorate in terms of quality. But the internal qualities (eg. Maintainability, etc.) are equally important as the others, and none of them should be ignored during the life-cycle of the software system. Short-term goals like quality should not be compromised for at the expense of long-term goals like maintainability. Istvan SiketIn [26] has presented a thesis comprised of a thorough study of applying software product metrics in software Maintenance to the University of Szeged in the year 2010. The work principally concerned with software maintenance, including testing and identifying a software system's bugs. The crucial part of the software development is to deliver a bug free product with the aid of testing and maintenance. Author have identified that there is a correlation between the defects and the metrics of an object-oriented system and the same is positively confirmed with several earlier researchers [27, 28, 29]. with a systematic and regular measurement and analysis of object-oriented metrics, an efficient system comprised of proper maintenance could be developed. This could lead to a best practice in software development. Experimental conclusions obtained with reference to the small and medium-sized system cannot be validated with a complex system. Deriving and estimating the metrics for a complex system is always a tedious and challenging role. These are two major difficulties addressed by the author. The proposed work has presented a solution for both these problems. A novel technology called Columbus Technology has been conceived to analyze and derive the metrics from a large and complex system. The Columbus framework has the ability to do automatically analyze and extract information from an arbitrary software system without no alterations in its source code. The validation of metrics has been done on seven version of Mozilla and the detailed hierarchical bug report is prepared. The work has extended the metrics presented by Chidamber and Kemerer [30] and are positively included in the proposed experimentation. This enhancement helped in examining metric categories rather than identification of metrics themselves. The work later constructed metric-based quality models to support the software

development. A opinion survey over the software developers has been carried to draw the practical aspect of metrics and about the relationship between four metrics and software comprehension & testing. This survey identified that a prior proper training in using metrics is mandatory before the practical use of metrics. The survey also supported in improving the metrics-based quality models.

Raed Shatnawi and Wei Li [31] found that the evidence available in the literature to support the belief software refactoring improves quality factors such as understandability, flexibility, reusability is not adequate and pursued an empirical experiment with an aim to confirm such beliefs using a hierarchal quality model. Authors observed the effect of software refactoring on software quality and presented the work in the form a heuristics for better understanding to software programmers. The work presented the validation of the proposed heuristics in an empirical setting on two open-source systems. The work concluded that the majority of refactoring heuristics do improve quality; still some heuristics do not have a positive impact on all software quality factors. Based on the impacts, the measures are classified into two kinds i.e high and low impacted measures. These categories drives in chosing the best measures that can be used to identify refactoring candidates. The experimental findings are validated on two open-source systems—Eclipse and Struts. For both systems, authors found consistency between the heuristics and the actual refactorings.

One of the major goal in performing software engineering is the prediction of software module complexity (a qualitative concept) using automatically generated software metrics (quantitative measurements). Nick John Pizzi [32] attempted to achieve the goal in a novel fashion by combining the problem with the science of pattern classification. Here the pattern could be a set of metrics for a software module, estimating the level of complexity to which the module belongs. Author presented a classification strategy to find the mapping between metrics and complexity, stochastic metric selection, to find the subset of software metrics which act as best predictors with respect to module complexity. The work empirically evaluated the publicly available medical imaging datasets with the proposed methodology and concluded that the proposed work is effective and strategic by comparing the prediction results against several classification system benchmarks.

An empirical assessment of metrics to predict the quality attributes is essential in order to deliver high software reliability. This problem is worked out in a new fashion by Ruchika Malhotra and Ankita Jain [33] by considering the task as a machine learning problem which is a similar paradigm presented in [32]. The work presented by Ruchika and Ankita [33] proposes a new model to estimate fault proneness using Object Oriented CK metrics and QMOOD metrics. Authors applied one statistical method and six machine learning methods to predict the proposed models. Datasets collected from Open Source Softwares are been chosen for validating the proposed models. Analysis of results is carried based on the two parameters via. Receiver Operating Characteristics (ROC) and Area Under the Curve (AUC). Based on the results it is observed that the proposed model predicted is superior with other models while using the random forest and bagging methods. Finally authors strongly supported that quality models have a significant impact with Object Oriented metrics and that machine learning methods have a comparable superior performance with statistical methods.

The extension of works presented in [32] and [33] has been pursued by many researchers with the intent of brining out a novel combination of using soft computing techniques with software engineering process. One such work is presented by Indu Sharma, ParveenBano [34] which proposes a fault prediction model using reliability relevant software metrics and fuzzy inference system. The work developed fuzzy profile of software metrics which are assumed to be more relevant for software fault prediction. The work predicts the density of faults at each phase of software process with the support of software metrics. This scenario is continued until the testing phase to estimate the total number of faults in developing product. Results of the proposed model are validated with the datasets collected using PROMISE Software Engineering Repository dataset and found that this proposed model will be suitable choice for both project managers and software programmers to optimally allocate resources and gain more reliable software within the time and cost constraints.

IV. RESERCH CONTRIBUTIONS IN SOFTWARE ENGINEERING USING AHP

The approach of selecting project management tool unlike other approaches is done in an adhoc manner. Such approach seems to be a non-rigorous which leads to erroneous result. Ahmad and Laplante [35] have introduced a rigorous model for selecting a software project management tool using the analytical hierarchy process (AHP). AHP is considered for its uniqueness in flexible, systematic, and repeatable evaluation procedure while selecting the appropriate software project management tool by decision maker. A list of factors and their significance scores has been provided by the commercial off-the-shelf solutions (COTS). The proposed work references these scores in opting factors as the selection criteria in ranking the software project management tools. The work also establishes a framework for cross comparison made across projects, project managers, organizational groups, and organizations.

The selection of appropriate reliability metrics in software engineering is very crucial and could never be ignored at any compromise. Haifeng Li et al. [36] proposed a framework for selecting software reliability metrics based on analytic hierarchy process (AHP) and expert opinion. The method of identifying appropriate criteria and their metrics is presented. Based on the experts guidance each criteria is graded with metrics qualitatively, and then analyzed synthetically to compute the weights of metrics using AHP. Metrics of high rank are analyzed critically. The proposed approach is further evaluated for its sensitivity and consistency by comparing the selection of criteria with other conventional methods, the method presented in this paper can be used to choose appropriate metrics correctly, stably and systemically. Finally, authors concluded that the work produced results are accordant with engineering experience, and using the metrics recommended will enhance software reliability evaluation more efficient and effective.

Reusability found to more beneficial in software development compared to other evolutionary concepts which reduces development time and cost, also improve quality, in developing a large, complex software system. To support this statement, the task of selection of potential software components, assigning them with appropriate score should be done during the process of requirements engineering process. Chung L et al. [37] have presented a technique for identifying requirements using a model which drives in evaluation of software components for their reuse. The evaluation process is comprised of verifying several models of software components. The proposed model-driven evaluation technique primarily intended to map the components of stakeholder with its corresponding matching component model. The process is also referred as component-aware requirements engineering (CARE). The proposed technique could be used in variety of search problems like keyword-based search, case-based reasoning (CBR) and analytic hierarchy process (AHP). The work has been experimented on a home appliance control system (HACS) example.

Value-neutral process is to be focused in the present software engineering practice. Several value-based architectural evaluation techniques and cost benefit analysis method (CBAM) are found in the literature and are widely used to increase return on investment (ROI). The limitations of the present available techniques are uncertainties from several subjective errors and the heavyweight process (complex), which involved in cumbersome steps and expert participation. Chang-Ki Kim et al. [38] with the intent of supporting a multi-criteria decision-making process, proposed a lightweight value-based architecture evaluation technique, called LiVASAE, using analytic hierarchy process (AHP). This proposed technique can help overcome the certain major limitations like uncertainities occurred because of subjective decision found in the existing conventional techniques. LiVASAE presents an effective way in three simple simplified evaluation steps to measure the uncertainty level using AHP consistency rate. Further, the LiVASAE presents a framework to assist decision makers to draw technical decisions connected with with business goals like cost, time-to-market, and integration with legacy system.

Formal modeling and quantitative modeling are the two base requirements in engineering. Metrics acts as the base to perform quantizing management in software management. The role of metrics is partial in quantification of software engineering. In order to make the process a complete, the factors that affect schedule, cost and quality of software development should be properly estimated. Yong Cao and Qing-xin Zhu [39] established a model to quantify the factors and introduced distance to compare the metric indicators. Their work is carried based on the maximum entropy principle presented by Jaynes [40][41]. Authors identified that the metric estimation tree comprising nodes as software attributes mapped with their corresponding evaluation values can aid in developing the proposed model. The method of dynamic feedback in the software processing is combined with AHP (analytic hierarchy process), for entire learning and analyzing the project and process of development.

The significance of Multi criteria decision making (MCDM) methods to decision is well discussed in the literature. MCDM assists decision makers to make preference decision over the variety of available alternatives. The software engineering tasks via. Evaluation and selection of the software packages categorized as a MCDM problem. AHP is well known for its selection and evaluation in several fields. Weighted scoring is also a widely used method for evaluation and selection of the software packages. Jadhav and Sonar [42] found a new approach called Hybrid knowledge based system (HKBS) approach for evaluation and selection of the software packages. Authors pursued an experimentation to analyze and the compare the performances of HKBS, AHP and WSM with respect to the task of evaluation and selection of software package. Authors observed that the comparison indicate HKBS approach comparatively better than AHP and WSM for evaluation and selection of the software packages in terms of computational efficiency, flexibility in problem solving, reuse of knowledge.

Requirements gathering should be done in the early stages of software engineering. Several cases exist where requirements keep on changing and the best completed requirements specification still misses the implicit requirements. Sadiq, M et al. [43] have presented an

algorithmic approach to identify the software requirements and their priority using analytic hierarchy process (AHP).

Software analysts mostly concentrate on delivering the functionality of the developing software product and postpone the quality concerns to the final stages of development. This practice would be expensive since the fixing of defects in later stages is tedious and cumbersome. That is, the concern of quality should be considered in the early configuration stages of requirement gathering. The identification of a feature for its quality is based on its impact on the overall product. Limitations like lack of quantitative measurements, valid products, expertise in assessing the human effort are the major barriers of existing approaches. Guoheng Zhang et al. [44] have presented a new approach of using Analytic Hierarchical Process (AHP) to estimate the relative importance of each functional variable feature on a quality attribute. The level of quality attributes of a product configuration could be assessed with reference to the impact of each functional variable feature of quality attribute. Authors have experimented their proposed approach on the Computer Aided Dispatch (CAD) software product line for clear demonstration.

The development of benchmarking methodology is quite a challenging role and is categorized as a continuous and consistency task. The assertion is getting slowly negated with the rapid development of software industry. One of the major benefits of benchmarking methodology is to evaluate domain-specific software product quality. Sun Haifang et al. [45] have proposed domain-specific software benchmarking methodology. Evaluation model is also combined with the work using fuzzy set theory and AHP (FAHP). FAHP helps in reducing uncertainness and vagueness, and also minimizes the role expert experience. The work is discussed with OA software as a case study.

Analytical Network Process seems to be another promising approach compared to approaches like AHP, decision support systems, fuzzy approach, AHP-GP etc. during locating the best software architecture style. Babu et al. [46] have presented a method which could do more precise and suitable decisions during selection of architecture styles with the help of Analytic Network Process (ANP) inference to assist the software architects while drawing decisions of in order to exploit implicit properties of styles in a simple and feasible way to optimize the design of software architecture. The frame work designed by authors is used to choose criteria within cluster criteria and alternatives from cluster of alternatives according to the requirements. During this process, the framework uses feedback, loops and also assign weights provided from various stake holders like users and domain experts to find the alternative in an efficient manner. Some of the key findings listed by authors are i) quality attributes which satisfy with the same software architecture interact each other ii) The identification of strengths and weaknesses with the support of criteria interaction during selection of architecture helps in making precise decisions.

Vijayalakshmi et al. [47] have devised a new architecture selection method based on multicriteria decision analysis called Multiplicative AHP (MAHP) combined with Weighted Product Mode (WPM). This methodology uses multiplication to rank software architectures rather than using addition. The proposed method is an efficient, simple and helps in performing accurate decision making for selection of software architectural style. The developed method is validated using stock market management system. Authors with the intent of improving the earlier methodology presented by Zayaraj [48][49] listed out the disadvantages. The validation of the proposed method has also been considered using a suitable case study. It is inferred that the proposed method more advantageous and gives more importance to the stakeholders' preferences and views. The completed methodology and evaluation procedure is explained using mathematical analysis.

Assessing the software security is a mandatory job to be done in all the phases of software life cycle, and to attain this task several factors should be considered like development environment, risks, and development documents. Zhuobing Han et al. [50] found that there is no solid methodology available in the literature in evaluating software security systematically. With the intent of establishing a solid methodology, Zhuobing Han et al. [50] have proposed a comprehensive model for evaluating the software security with three orthogonal and complementary concepts via. Technology, Management and Engineering. ISO/IEC15408 provides the guidelines for the technological dimension with 7 security levels based on Evaluation Assurance Levels (EALs). The management dimension primarily looks for the management of software infrastructures, development documents and risks. The third dimension (engineering) is sub categorized on 5 stages of software development lifecycle. Evidences drawn by experts from these three dimensions provide necessary assessments for software security. During the process of preparing assessments, advanced decision systems like Analytic Hierarchy Process (AHP) and Dempster-Shafer Evidence Theory are been leveraged. These assessments are again combined with the experts experience to obtain a score which presents the security degree of software. The work has been illustrated with a case study presenting the detailed discussion of the proposed approach to evaluate security of their system.

Sustainability and reusability have great significance in engineering learning domain. In this concern, metadata helps in supporting reusability and effective use of Learning Objects

(LOs). Sometimes searching in LO repository based on metadata consumes more time. Required LO could never be filtered if the chosen criteria do not exactly match the metadata values. This situation could be overcome with the multi-criteria decision making (MCDM) method. Yigit et al. [51] have carried a study and developed the SDUNESA software. With the support of AHP, this software allows for the selection of a suitable LO from the repository using MCDM method. AHP aids in selecting the necessary parameters prior to the process of searching. The software is developed using Web 2.0 technologies like AJAX, XML, SOA servies. Experimental results show that the combination of AHP to the MCDM method picks the most reliable learning object that meets the criteria.

Assessment and sensitivity are the two crucial analysis of software engineering reliability. Chen Qu et al. [52] have presented a research article which established the evaluation attributes of comprehensive categorical, and analysis assessment evidence based on the inferences of existing literature and computational logic by AHP. The work is experimented on a power plant information management system involved in analyzing the domain software reliability by the proposed method, and pursued the attribute sensitivity analysis based on relative data. Authors finally inferred that the work can analyze and evaluate the domain software reliability unambiguously, and efficient sensitivity analysis.

A complete requirements engineering can be used as a half job done and if decision support system is combined in requirements engineering then the resulting document is totally with all possibile conflicts. Vinay et al. [53] have proposed Goal Oriented Requirements Engineering (GORE) methods and have addressed improved aspects which are helpful in decision support. They have also proposed a combined method called IGAPE (GORE method – Integrating Goals after Prioritization and Evaluation). This method is designed as semi-formal to ensure active stakeholder participation. The knowledge derived by IGAPE is supplied as input to AHP supported decision system. A technique is developed for Order of Preference by Similarity to Ideal Solution (TOPSIS) is combined with decision system. Authors found that the integrated system (IGAPE with AHP and TOPSIS) will provide improved decision making compared to other works during requirements engineering phase. The proposed method is analyzed and illustrated with the help of an ecommerce application and is evaluated by expert analysis approach.

The concept of cloud computing drastically increased the computing services and such a system could be perceived as a dynamic allocation of information systems according to the cloud demands selecting the best mix of compute services and processing virtual machines. This situation strong requires evaluating whether existing web applications could work with the same quality at reduced cost when migrated to cloud machines. Also application engineers who develop these web applications should consider several crucial criteria like heterogeneous nature, complexity, etc. which are hard to solve manually. Menzel et al. [54] have previously developed a framework to support cloud system of single-component web applications. The migration process for web applications distributed over various locations is enhanced by the authors with intent of identifying the most important criteria relevant to the selection problem. This enhancement is categorized as a multi-criteria-based selection based on Analytic Hierarchy Process (AHP). A genetic algorithm is developed by the authors to cope with the exponential solution space in growing cloud market. The proposed system CloudGenius is evaluated for its applicability using adequate use case example proofs. The proposed work also developed a prototype of selection algorithm called CumulusGenius to conduct experimentation. Genetic algorithm is to deploy on hadoop clusters for selection. The work has presented the experiments with CumulusGenius and clearly discussed on the time complexities and the quality of the genetic algorithm.

V. SUMMARY

AHP is primarily leveraged as a multiple criteria decision-making tool. This process follows an eigen value approach to the pair-wise comparisons. The process also provides a methodology to calibrate the numeric scale for both quantitative and qualitative performances. An attempt has been made in this paper to analyze and present a comprehensive review of AHP in software engineering. Majority of the reviewed articles listed in this work belongs to one of the following combinations via. engineering and selection, social and selection, and personal and decision making. This identifies the nature of AHP to use as a decision making tool in engineering as well as in social sector. The positive side of AHP increased the confidence of the researchers began experimenting the combination of AHP with other techniques like linear programming, artificial neural network, fuzzy set theories, etc. Still AHP is considered in multi-criteria decision making in a stand-alone mode. Thomas L Saaty being identified as one of the early researchers worked in decision making. The modified versions of AHP show drastic improvement and during the process several authors have converted the Saaty 0's 9-point scale into convenient 5-point scale. Louis G Vargas presented critical analysis of using decision making in various fields like business, health, energy and transportation. Ian Sommervillie has made a textbook included with all the available research articles of software engineering. H F Li, John Pierrie, Daniel Moody, Aine Mitchell, Lionel and Raed Shatnawi some potential researchers

who employed software engineering using empirical approach. Babu, vijaya lakshmi and Vinay are some contemporary researchers who worked in software engineering using AHP and other decision making combinations.

REFERENCES

[1] Hally Mike, "Electronic brains/Stories from the dawn of the computer age", London: British Broadcasting Corporation and Granta Books. p. 79., 2005.

[2] "Tying Arrangements and the Computer Industry: Digidyne Corp. vs. Data General", JSTOR 1372482. (https://en.wikipedia.org/wiki/JSTOR) last accessed on 18.07.2015.

[3] Sommerville Ian, "Software engineering has recently emerged as a discipline in its own right", Software Engineering. Addison-Wesley, 1985. ISBN 0-201-14229-5.

[4] Meyer, Bertrand, "The origin of software engineering, 2013". (https://bertrandmeyer.com/2013/04/04/the-origin-of-software-engineering) last accessed on 18.07.2015.

[5] Tadre, Matti, "The Science of Computing", CRC Press. pp. 121. ISBN 9781482217704.

[6] H F Li, W K Cheung, "An Empirical Study of Software Metrics", IEEE Transactions on Software Engineering, Vol SE 13, No. 6, pp. 697 – 708, June 1987.

[7] William A Florac, "Software Quality Measurement: A Framework for Counting Problems and Defects", Technical Report, CMU/SEI-92-TR-022, ESC-TR-92-022, Software Engineering Institute (SEI), September 1992.

[8] M C Paulk, B Curtis, M B Chrissis, C V Weber, "The Capability Matured Model for Software", IEEE Journal of Software Engineering, pp. 427 – 438, 1997

[9] IEEE Standards Board, IEEE Standard for a Software Quality Metrics Methodology, December 1998

[10] Ron Burback, "Software Engineering Methodology: The Watersluice", Doctoral Thesis Submitted to the Stanford University, 1998.

[11] Fenton N E, Neil M,"A critique of software defect prediction models", IEEE Transactions on Software Engineering, Vol. 25, No. 5, pp. 675 – 689, 1999

[12] John Pierrie Kuilboer, Noushin Ahsrafi," Software Process Improvement Deployment – An empirical Perspective", Journal of Information Technology Management, Volume X, No: 3- 4, 1999

[13] Maria Sverstuk, June Verner, Jeffry Hand,"Software Quality: What is really important and Who Says So", International Conference NIMESTIC 2000, 11 – 13 September 2000

[14] Boehm B W, Brown J R, Lipow M, "Quantitative Evaluation of Software Quality", Proceedings of Second International Conference of Software Engineering, pp 592 – 605, 1976.

[15] Isabel Brito Ana , Ana Moreira , João Araújo, "A Requirements Model for Quality Attributes, Aspect-Oriented Requirements Engineering and Architecture Design", 2002.

[16] Daniel L Moody, "Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice", ECIS 2003: 1337-1352, 2003.

[17] Souheil Khaddaj, G Horgan, "The Evaluation of Software Quality Factors in Very Large Information Systems", Electronic Journal of Information Systems Evaluation, Vol. 7, No. 1, pp. 43 – 48, 2004.

[18] Tom Mens, Tom Tourwe, "A Survey of Software Refactoring", IEEE Transactions on Software Engineering, Vol. XX, No. Y, 2004.

[19] Aine Mitchell, "An empirical study of run-time coupling and Cohesion software metrics", Doctoral Thesis, Submitted to the Dept. of Computer Science, National University of Ireland Maynooth, 2005

[20] Bart Du Bois, "A Study of Quality Improvements By Refactoring", Doctoral Thesis submitted to the University of Antwerp, Belgium, 2006

[21] Girish H Subramanian, James J Jiang, Gary Klein, "Software quality and IS project performance improvements from software development process maturity and IS implementation strategies", The Journal of Systems and Software, pp. 616 – 627, Vol. 80, 2007

[22] R. Radharamanan, Jeng-Nan Juang, "Determining Significant Factors and their effects on Software Engineering Process Quality", ASEE publications, Fall-2009.

[23] Trendowicz, Jürgen Münch, "Factors Influencing Software Development Productivity - State of the Art and Industrial Experiences, Advances in Computers", Elsevier, Vol. 77, pp. 185 – 241, 2009.

[24] Forselius, P.; Kakola, T., "An Information Systems Design Product Theory for Software Project Estimation and Measurement Systems," System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on , vol., no., pp.1-10, 5-8 Jan. 2009. doi:10.1109/HICSS.2009.65.

[25] Yonghee Shin, Andrew Meneely Laurie Williams, "Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities", IEEE Transactions on Software Engineering, Vol. 37, No. 6, pp. 772 – 787, 2010.

[26] Istvan Siket, "Applying Software Product Metrics in Software Maintenance", Doctoral Thesis submitted to the Department of Software Engineering, University of Szeged, 2010

[27] Victor R. Basili, Lionel C. Briand, and Walc´elio L. Melo. "A Validation of Object-Oriented Design Metrics as Quality Indicators". In IEEE Transactions on Software Engineering, volume 22, pages 751–761, October 1996.

[28] Lionel C . Briand and Jurgen W¨ust . "Empirical Studies of Quality Models in Object- Oriented Systems. In Advances in Computers", volume 56, September 2002

[29] Ping Yu, Tarja, " System Predicting Fault-Proneness using OO Metrics: An Industrial Case Study. In Sixth European Conference on Software Maintenance and Reengineering", (CSMR 2002), pages 99–107, March 2002

[30] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object-Oriented Design". IEEE Transactions on Software Engineering 20,6(1994), pages 476–493, 1994

[31] Raed Shatnawi , Wei Li, "An Empirical Assessment of Refactoring Impact on Software Quality Using a Hierarchical Quality Model", International Journal of Software Engineering and Its Applications, Vol. 5 No. 4, October, 2011

[32] Nick John Pizzi, "Mapping Software Metrics to Module Complexity: A Pattern Classification Approach", Journal of Software Engineering and Applications, Vol. 4, pp. 426-432, 2011.

[33] Ruchika Malhotra and Ankita Jain, "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality", International Journal of Information Processing Systems, Vol.8, No.2, June 2012

[34] Indu Sharma, ParveenBano, "A Combined Approach of Software Metrics and Software Fault Analysis to Estimate Software Reliability" IOSR Journal of Computer Engineering (IOSR-JCE) Vol. 11, No. 6, pp. 01-14, 2013.

[35] Ahmad, N.; Laplante, P.A, "Software Project Management Tools: Making a Practical Decision Using AHP", 30th Annual IEEE/ Software Engineering Workshop, pp. 76 – 74, 2006.

[36] Haifeng Li; Minyan Lu Qiuying Li, "Software Reliability Metrics Selecting Method Based on Analytic Hierarchy Process", IEEE Sixth International Conference on Quality Software, (QSIC), pp. 337 – 346, 2006.

[37] Chung, L.; Weimin Ma; Cooper, " Requirements elicitation through model-driven evaluation of software components", IEEE Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, pp. 10, 2006.

[38] Chang-Ki Kim; Dan Hyung Lee; In-Young Ko; Jongmoon Baik, IEEE Eighth ACIS International Conference onSoftware Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, (SNPD), pp. 646 – 649, 2007

[39] Jaynes, E. T, "Information Theory and Statistical Mechanics", Physical Review. Series II 106 vol. 4, pp. 620–630, 1957.

[40] Jaynes, E. T,"Information Theory and Statistical Mechanics ", Physical Review. Series II 106 vol. 4, pp. 620–630, 1957.

[41] Jaynes, E. T "Information Theory and Statistical Mechanics II (PDF)", Physical Review. Series II 108 vol. 2, pp. 171–190, 1957.

[42] Jadhav, A. and Sonar, " A Comparative Study", 2nd IEEE International Conference on Emerging Trends in Engineering and Technology (ICETET), pp. 991-997, 2009.

[43] Sadiq, M.; Ghafir, S.; Shahid, M. , "An Approach for Eliciting Software Requirements and its Prioritization Using Analytic Hierarchy Process" , IEEE International Conference on Advances in Recent Technologies in Communication and Computing, pp. 790 – 795, 2009

[44] Guoheng Zhang; Huilin Ye; Yuqing Lin," Quality Attributes Assessment for Feature-Based Product Configuration in Software Product Line", IEEE 17th Asia Pacific Software Engineering Conference (APSEC), pp. 137 – 146, 2010

[45] Sun Haifang; Cai Lizhi; Liu Xiaoqiang; Song Hui; Yang Genxing; Liu Zhenyu; Meng Zhiming, "Domain-Specific Software Benchmarking Methodology Based on Fuzzy Set Theory and AHP", IEEE International Conference on Computational Intelligence and Software Engineering (CiSE), pp. 1-4, 2010

[46] Babu, D., K., Rajulu, G., P., Reddy, R., A., Kumari, A., A., N., "Selection of Architecture Styles using Analytic Network Process for the Optimization of Software Architecture", International Journal of Computer Science and Information Security, Vol. 8, No. 1, April 2010.

[47] Vijayalakshmi, S., Zayaraz. G., Vijayalakshmi. V., 2010. "Multicriteria Decision Analysis Method for Evaluation of Software Architecture. 2010", International Journal of Computer Applications (0975 - 8887) Vol.1, No. 25, 2010

[48] Zyaraz . G, Dr. P. Thambidurai, "Quantitative Model for the Evaluation of Software Architectures", Journal of Software Quality Professional, American Society for Quality, Vol.9, no.3, pp. 28-40, June 2007.

[49] Zayaraz .G and Dr. P. Thambidurai, "Software Architecture Selection Framework Based on Quality Attributes", Proceedings of the IEEE Conference INDICON, pp. 67-170, Dec. 2005

[50] Zhuobing Han; Xiaohong Li; Ruitao Feng; Jing Hu, "A Three-Dimensional Model for Software Security Evaluation", IEEE Theoretical Aspects of Software Engineering Conference (TASE), pp. 34 – 41, 2014

[51] Yigit, T.; Isik, A.H.; Ince, M, "Web-based learning object selection software using analytical hierarchy process Software", IEEE IET, Volume: 8, Issue: 4 pp. 174 - 183, 2014.

[52] Chen Qu; Bao Tie; Zheng Wanbo; Lian Wei, "IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)", pp. 502 – 505, 2014.

[53] Vinay S, Shridhar Aitha and Sudhakara Adiga, "Integrating Goals after Prioritization and Evaluation – A Goal Oriented Requirements Engineering Method", International Journal of Software Engineering & Applications (IJSEA), Vol. 5, No. 6, 2014.

[54] Menzel, M.; Ranjan, R.; Lizhe Wang; Khan, S.U., "A Hybrid Decision Support Method for Automating the Migration of Web Application Clusters to Public Clouds", IEEE Transactions on Computers, Vol. 64, No. 5, pp. 1336 – 1348, 2015