International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015
ISSN 2091-2730

Qualitative Approach For Estimating the Influence Of Refactoring And Scrum In Software Development

Rida Ghafoor Hussain, Ali Javed

Department of Software Engineering

University Of Engineering And Technology, Taxila, Pakistan.

rida_ghafoor@yahoo.com , ali.javed@uettaxila.edu.pk

Abstract— Software development is intellectually a complex chore. The swift progress of software currently requires the high rate software product release by development teams. Different software development techniques and quality assurance methods are used in order to achieve high worth, unfailing, and error free software. In order to deliver the product earlier, the development teams make an alteration to their conservative software development lifecycle to agile development method which can allow them towards prompt release of software management with the requirements-change experience. Refactoring has been rising in magnitude with modern software engineering advances, predominantly agile methodologies, which endorse uninterrupted progress of an application's code and blueprint. Refactoring is the practice of analyzing and facilitating the plan of offered code, without altering its performance. Another trendy techniques in Agile development is the Scrum methodology. It involves regular release and the client receives an absolutely prepared application that includes more and more features every time In this paper Qualitative Approach For Estimating the Influence Of Refactoring And Scrum In Software Development is utilized. In this model scrum methodology is utilized in enhanced form to overcome scrum issues along with refactoring project at both design and implementation level.

Keywords— AOSD, CBO,KPI, SDMs, MDA,XP,agile,refactor

INTRODUCTION

The development of the Internet and the electronic frugality has tainted the policy of software engineering. Conventional software development methodologies (SDMs) are being recouped by agile SDMs. Agile SDMs are outlined by incremental development, incessant code assimilation, and the capability to switch altering production requirements. Agile technology is used to generate advanced quality software in a briefer epoch .Agile procedures were refined to rationalize the growth practices and remove boundaries to accommodating production requirement changes through the growth process. Agile methodologies do not need that business requirements and plan particulars be protected in for the period of expansion [1]. Agile SDMs contribute to numerous features including prototyping, incremental development, and negligible citations [1]. Extreme Programming (XP) is an agile (lightweight) soft-ware development methodology and it becomes more and more well-liked. Extreme programming (XP) is one of the mainly and extensively used agile practices for software development. It tries to look up software quality and receptiveness to varying client requirements. Software refactoring is an XP procedure to augment the maintainability of software, improve reusability and understandability of the software. Refactoring is basically the object-oriented variant of restructuring:"the process of changing an object-oriented software system in such a mode that it does not change the peripheral performance of the code, however enhances its inner organization" [3]. In the refactoring process, modifications were done to the scheme parting its performance unaffected, but upgrade some non-functional quality like integrity, flexibility, understandability, etc., [4].Un-refactored code contribute to decompose. Rot takes several forms: insanitary interdependence between classes or packages, poor distribution of class errands, too many responsibilities per method or class, replica code, and many other variations of uncertainty and litter. This is because each time code is modified without refactoring, rot aggravates. Code rot disappoints users, overheads time, and excessively reduces the lifetime of practical systems. In an agile perspective, it can signify the difference between fulfilling or not fulfilling an iteration target. Refactoring code callously prevents rot, keeping the code trouble-free to sustain and expand. This extensibility is the reason to refactor and the degree of its success. The Scrum procedure was also considered to switch speedily altering business requirements. The practice's name is a consequent of an approach used in the game of Rugby. In a Rugby scrum, the ball is passed reverse and forward between players to move the ball onward. The Scrum method promotes a project by enhancing connection between group members and splitting the task into a series of “sprints” that last thirty days or less [Schatz]. Scrum focuses additional on organization of the growth process than coding phenomenon [5]. Scrum is a practica that can be utilized on little and huge projects. Individual teams can use the Scrum methodology on their projects while huge projects can be splitted into modules and a Scrum team ascribed to each subproject. The connection and main concern administration between the subproject teams can be regulated with Scrum techniques. The key purpose of the investigation is to enhance software development model using Refactoring and Scrum practices. The software development model is planned in such a way that refactoring activity accompanies all design and implementation phases of software development. Similarly scrum methodology is proposed considering certain issues like usability, understanding, security etc. This effort also shows the estimation of influence of projected strategy.

LITERATURE REVIEW

Quality cannot be attained rapidly in any software development. Perfection comes over the epoch. Adjustment in the software development form is required in managing quality product deployment.[11] Agile software development procedures are used to yield high aspect software in briefer duration of time.[14] "Refreshing requirements yet belatedly in development" is fundamental of agile development methodology.[6] A few prominent agile development techniques are Extreme Programming (XP), Scrum, Crystal Methods, Feature Driven Development (FDD) and Test Driven Development. These procedures behave in a different way from conventional Software Development Methods and facilitate systems meet up the demands of the digital cost-cutting measure .[15] Refactoring is a foremost technique used to handle changes. Freezing the external performance it facilitates to restructure code .[6] Similarly Scrum is a nimble way to run a development, typically software development. Scrum software development team deals with project development in the agile development based on scrum methodology. It doesn't need detailed imagery of implementation, because the team knows best optimization of project to solve the problem.[16] Karim M. Zaki, Ramadan Moawad in [9] "A Hybrid Disciplined Agile Software Process Model" provides integration between customary techniques, configuration, constancy and swift understanding of populace, capability, ease. The proposed model will provide a platform that regulates aims and objectives of both management and implementation team. The past vacancies and issues could be easily resolved and a follow up will be provided to track system status. Manjunath K N, Jagadeesh J, Yogeesh M in [11] "Achieving quality product in a long term software product development in healthcare application using Lean and Agile principles" have proposed V-model.By comparison of the outcome obtained through implementation of Agile principles with previous results, it is observed that this model is an excellent approach for lengthy projects in healthcare departments. Completion of V-model will be possible through agile and lean approaches in every iteration of requirements change. A. Ahmed, et al in [10] "Agile Software Development: Impact on Productivity and Quality" recommended a model that accentuate on code refactoring .There should be a chapter of high rank blueprint of the project after the early requirements gathering and reasoning of the project. The suggested model recommends that the design must be stretchy as much as necessary to put up changes afterward in the execution part. Code refactoring guarantee simple development and expandability of the project making it more comprehensible and advances the quality of the code. K.Ush, N.Poonguzhali ,E.Kavitha in [7] "A Quantitative Approach for Evaluating the Effectiveness of Refactoring in Software Development Process" conducted various experiments based on refactoring in the developed model .Research in this proposed method extend refactoring in three phases, Identification, Proposal and Application. The Evaluation criteria's are Reusability, Understandability, Maintainability. In evaluation phase, the effect of refactoring based on evaluation parameters is quantified using software metrics. From the execution results, it is concluded that, refactoring phase in the software development increases reusability and understandability of the code, thus boosting maintainability of the code. M. Kleyman, S.Tyszberowicz, A.Yehudai in [6] "Refactoring Aspects into Java Code" declared transformation through refactoring. This opposite transformation is done by ACME i-e generation of a object oriented structure through refactoring. It can be when a structural change that makes aspect inappropriate is required. In several cases, aspects are not utilized just as some organizations avoid in production code. But now ACME allows use of aspect is possible and object oriented code can be obtained according to requirement. Without changes in already existing classes ACME allows implementation modifications. Automatic refactorings can be done. However, it will be an addition to the core functionality of ACME. It depends on programmer, if he wants to apply conversions on aspects. The case study presents that without intentions of using ACME, it is possible to convert aspects. S.A.M.Rizvi, Zeba Khanam in [8] "A Methodology for Refactoring Legacy Code" organized methodology which is embraced based on refactoring. First of all, the purpose to make the intact process is defined. Selection of refactoring patterns from the existing and new catalog is done after creating the goal. Next step is to determine the application of refactorings. The development team can then explore refactoring opportunities and assess the effects and results of refactoring. Systematically, refactoring can be applied to move in the correct path instead of applying on different inventories. Bart Du Bois, et al in [12] "A Discussion of Refactoring in Research and Practice" in their research paper demonstrates that software refactoring is revolving efficiently, different business tools for refactoring are increasing swiftly, but there are a large number of issues that still require consideration and solution. In refactoring research, they deal with action-preservation, assembling responses and feedback on refactoring implementation and assimilation with MDA and AOSD for future research work. There is requirement for procedures, techniques and tools that tackle refactoring in a more constant, aimed, flexible manner. Raimund Moser, et al in [13] "Does Refactoring Improve Reusability?" examines reusability of adhoc in refactoring is either supported or not. In software development and implementation, reusability should be supported at maximum. Refactoring have improved object oriented classes by promoting internal metrics like reusability especially in adhoc. The first choice for development code could be refactoring as it enhances many factors like reusability and maintainability.

PROPOSED FRAMEWORK

In this model scrum methodology is utilized in enhanced form to overcome scrum issues along with refactoring project at both design and implementation level. The start of development is a simple design and any flexibility problem if discovered later through the process, the design is refactored. Refactoring can be applied to all phases of software development and artifacts (design, test cases, use cases, sequence diagrams etc).

1. Refactoring at design and implementation level:

In web development ,refactoring can be applied at both design and implementation phase. Implementation phase refactoring is similar to code-level refactoring both by convention and structure but in code refactoring we work with object oriented programming but in this type all other codetypes like HTML, javascript ,XML etc are also refactored.

1.1. Design level

In this phase we will focus on navigational refactoring model where navigational class diagram are transformed that preserve operational semantics and navigability. It means that existing nodes may not become unreachable though the set may be augmented (e.g. by splitting a node).Following steps must be considered while doing navigation model refactoring:

1. Add operations, content and links to the node, already present.
2. Add a new node
3. Remove a node with no link i-e unreachable.
4. If the node does not become unreachable by removing a link, remove it.

Analysis of application's usage have shown that users repeatedly use forward and backward links when navigating a web application. This is because the target link is not the user expected.Too much false link activations will lead to frustration and confusion.User will ultimately leave the site.This model provides a solution to this problem .

1.1.1. Anticipate target:

By adding a script to the link anchor, a mouse can hover over link which will give a small version of the target page. This may be also called interface refactoring and is utilized in advance scripting languages like AJAX.

Figure 1:small version of target information[23]

*1.1.2.*Introduce link destination announcement:

By adding a script to particular widget or index, mouse can hover over link which will open a pop-up menu consisting of all possible operations and functionalities related to that link. The drawback of applying this solution is that pop-ups may be blocked or may be annoying for some users.

1.1.3. **Introduce scrolling:**

Use vertical and horizontal scroll bars.

Figure 2:scroll bars[24]

1.1.4. **Split list:**

Divide entries of index in several pages. This will make it user friendly. There are plenty of examples of this, like Google search results. Example: B-commerce applications usually provide recommendations for their products as an effective way of advertising. It has become a trend that emerging website has a starting page with list of products and titles etc. All operations related to it must be shifted to next linked pages.

1.1.5. **Add operation:**

To remove repetitive operations, add operations in main class from where it can be accessed when required. For example, if a user want to buy a product then if he is already a customer of that site, all user information will be retrieved. User will not have to write details on next product purchase.

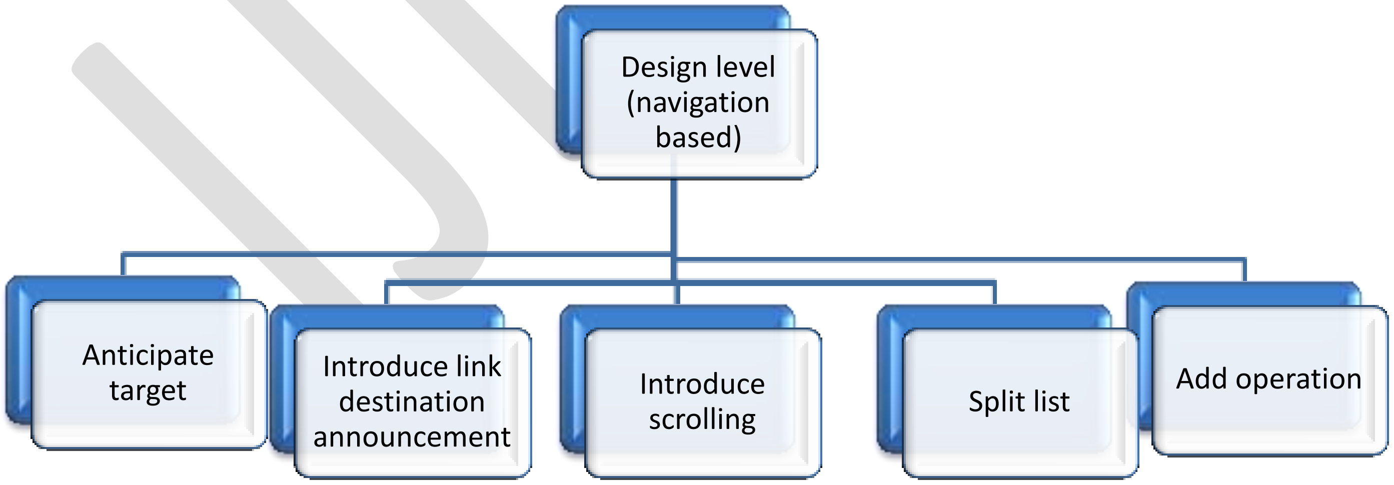
Design level (navigation based)
Anticipate target
Introduce link destination announcement
Introduce scrolling
Split list
Add operation

Figure 3: phases of design level refactoring for web based application

1.2. Implementation level:

Improvement of code clarity and enhancing maintainability and usability are the main objectives of refactoring. Therefore it is reasonable to evaluate the refactoring effect in terms of:

1. Maintainability
2. Reusability
3. Understandability

1.2.1.Refactoring through code smell:

This is identification phase where source code area to be refactored is identified. This is done by means of code smell detection. Normally there are two types of bad smell:

1. Duplicated code : By unifying these parts code is enhanced
2. Lazy class (class that does not play significant role): They may be excluded by adjusting their functionality somewhere else in the source code.

1.2.2.Selection of metrics:

Refactoring can be implemented in two modules:

1. Selection of appropriate quantification metrics for maintainability, reusability and understandability.
2. Measuring and comparing metrics before and after refactoring.

Popular metrics suites are Halstead's Complexity Measures [17], McCabe's. Cyclomatic Complexity [9] and Maintainability Index[18].

1.2.3.Effect of Refactoring based on Complexity Measure:

***NOA*:** Number of attribute metrics is used to count the average number of class and instance variable.A class with large number of variables indicates cohesion. Class requires further decomposition to manage complexity. Number of attributes > 10 indicates poor design.

***NOM*:** number of methods in a class. A class must not have excessive number of methods in it.

TLOC: Total lines of code in class metric will count the non-blank and non-comment lines in a class. Size of the system can be easily measured through it.

NOC: Number of classes in package.The overall size of the system can be estimated by finding the number of classes it contains. system with more classes become complex because object interaction is higher.

CC (m): Cyclomatic Complexity in a method . It measures the number of linearly independent paths through a system program module. The amount of decision logic in each software module is measured.

1.2.4.Effect of Refactoring based on Modularity Measure:

Coupling between Objects (CBO):Coupling measurement is done . when one class uses functions or variables of another class then coupling is said to occur. Understandability and maintainability become difficult. C&K suggest CBO as an indicator for evaluation of effort required for maintenance. CBO was found to be helpful in detecting highly coupled classes [19]. In [20], presents that maximum value for CBO is 30 and min value to be o.

Coupling Factor (CF): computes the ratio of number of pairs of classes that are coupled with each other to total number of possible couplings in a given system of classes [20].

Lack of cohesion among Methods (LCOM):The metric counts the number of methods ‘pairs in the class using no attributes in common, minus the number of pairs of methods that do.

1.2.5.Applying and comparing results:

- Average Number of Attributes will make the code simple and less
- Decrease in Average Number of Methods makes the code simple. Class is not overloaded with more functionality.
- System with few classes will be understandable and easier.
- Decrease in the average Cyclomatic complexity reduces The system complexity
- Decrease in the value of CBO improves reusability which decreases the dependencies exits between classes.
- Maintains Moderate values for CF enhances maintainability and reusability without any side effects
- Decrease value of LCOM increases cohesion values, thereby improving reusability and Understandability of the code

Scrum artifacts that is proposed to change to work better in Network Organizations.

1. Task-feasibility instead of time-estimation:

Instead of using formal time estimates ,it is focused to commit only those user stories which are realistic to implement before next session.Through this change, commitment is limited, which is unable to obtain.

2. Report Meeting instead of Sprint Review Meeting:

We propose to limit participants only to representatives of the customer and the team because sprint review meetings require lot of resources (i-e participants) .This type of meeting should be held more frequently in order to improve performance and requirements gathering between customer.

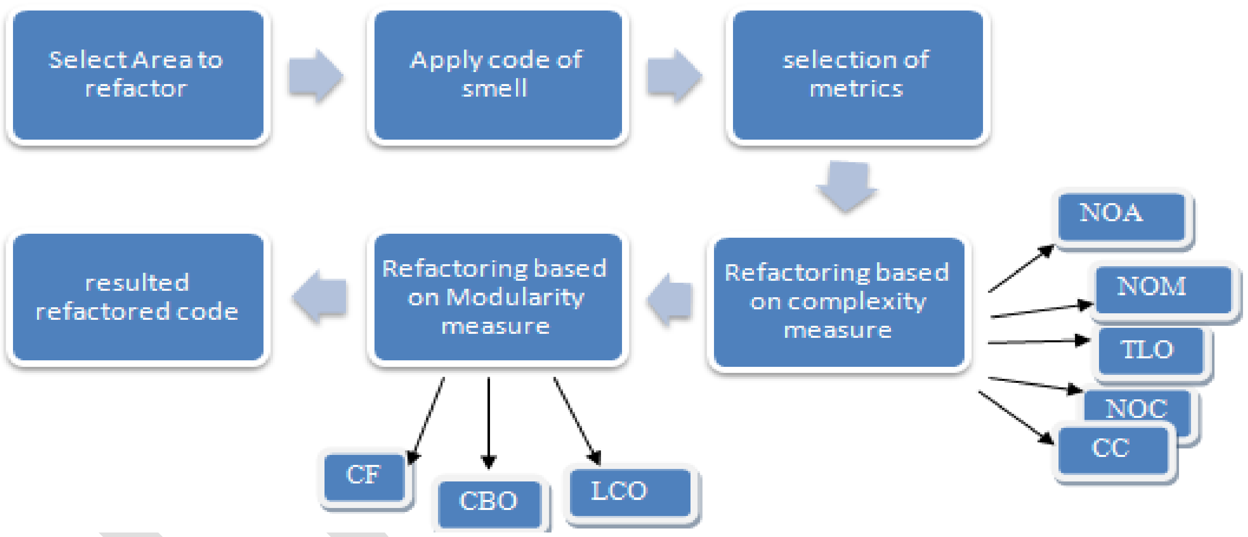

Figure 4: phases of implementation level refactoring

3. Key Performance Indicators:

An item of information collected to track the performance of a system is known as performance in indicator [22].In scrum, indicators are used as time-estimate of the remaining work amount that needs to be done versus amount of User stories that are considered done in Sprint Backlog [21].It is pro-posed to use the following KPI's (i.e. Key Performance Indicators) that help better control software development in Net-work Organization:

• ***Reliability***: to measure if the team is successful in achieving the desired. The difference between the amount of committed Story Points (ci) and delivered Story Points is represented as percentage of reliability calculated per Sprint (Ri) can be calculated as.

$$Ri= ci/di *100\% \qquad (1)$$

• ***Productivity***: to measure project velocity.The amount of fixed bugs (bi) and newly implemented re-quirements (si) are represented as the value of produc-tivity (Pi) after delivery of each sprint should be calculated as :

$$Pi= bi + Si \qquad (2)$$

• ***Effectiveness*:** to evaluate effectiveness of testing service by measuring the amount of defects delivered to the customer. The ratio between all found defects (ai) and those found by external S (ei) providing complementary testing are measured to calculate the effectiveness of internal testing service Based on this KPI. This shows effectiveness (Ei) of soft-ware development team and testing services:

$$Ei =ai-ei/ai*100\% \qquad (3)$$

2. Scrum Enhancement:

2.1. Security Backlog:

By analyzing the phases in scrum it is analyzed that in first phase i-e planning the approved product is used .User/client approve this product and the users/clients do not know much about security risks. An extra backlog can be managed to overcome security risks well without affecting the agility of this method. For this, an additional role, called "Security Master" is introduced . The Security Backlog follows existing security principle so that security issues can be reduced. the features in Product Backlog are made security-pruned by the addition of security backlog. No feature should miss its security concern. The features in Product Backlog will go through Security Backlog. Only the certain features in the product backlog are figured out by the security master that require the security attention. The security requirements for the selected features in security backlog are marked. The testing part will be conducted and the marked security concerns will be carried forward to sprint backlog for developers' attention. The features selected by the Scrum Master are processed as usual like other processes. It is also helpful to highlight the development team and client wish to discuss about the security requirement or anything related to security to the features. The Security Backlog is related to technical skill to identify and overcome the security risks, so it is also recommend to make a security education first through training to developer and stakeholder to make sure their security awareness are satisfied.

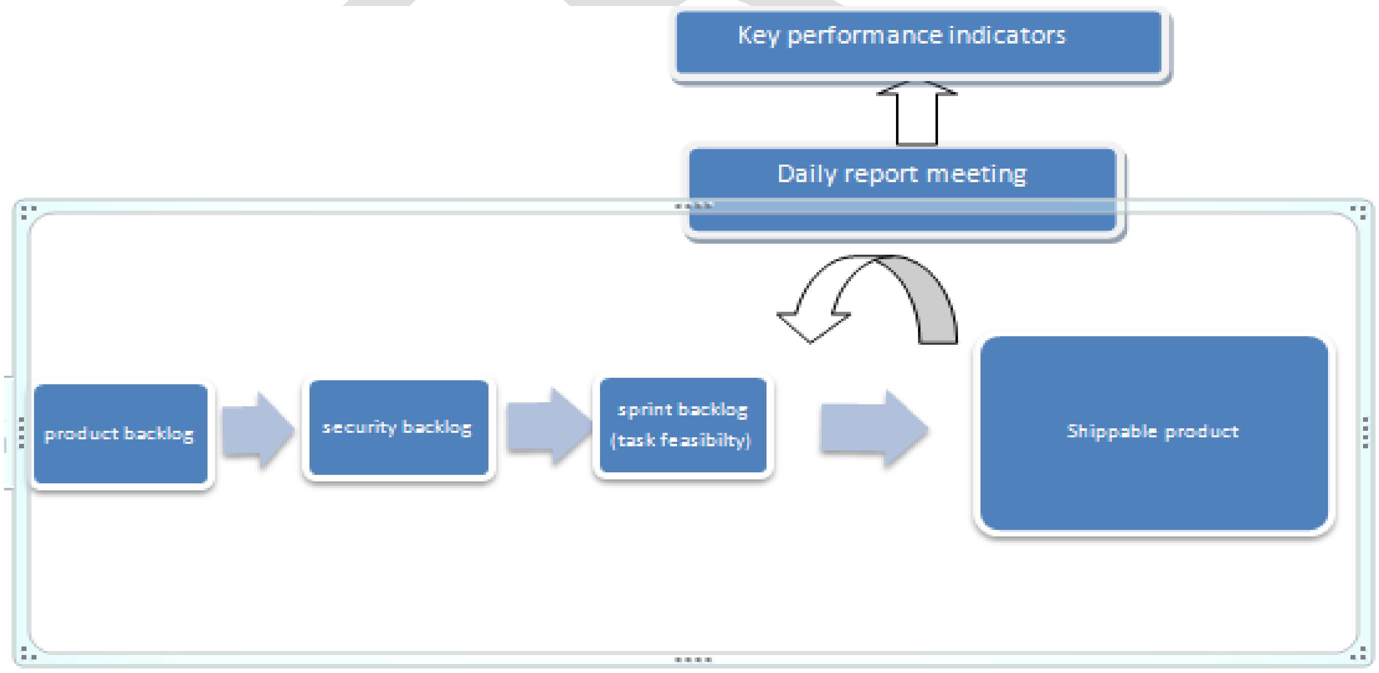

Figure 5: phases of implementation level refactoring

CONCLUSION

In this research work initially a refactoring methodology at design and implementation level is proposed. An approach for design refactoring in Web applications is presented specifically. It is based on the view of modern Web engineering methods and it considers refactorings to the navigation and presentation design models. It is demonstrated how refactorings can help Web applications evolve by applying well-known Web patterns into their design, in order to improve quality in use properties, such as usability. At implementation level, metrics are selected and their effects are described .The Evaluation parameters are taken as Maintainability, Reusability and Understandability. It is conclude that, refactoring activity in the software development process leads to improvement in reusability and Understandability of the code, thereby enhancing maintainability of the code. Further, scrum method is enhanced by introducing security backlog and monitoring of indicators in report meetings. this will be cost-effective and secure. Usability level and user understanding is also improved.

FUTURE WORK

Methodology based on design refactorings for all applications in addition to web applications. The proposed methodology can be practically implemented in software development processes .

ACKNOWLEDGMENT

We would like to thank our supervisor Assistant Prof. Ali Javed for being very encouraging and supportive throughout which helped me to complete the research paper very smoothly.We would like to thank our family and friends for being supportive with us all through our work

REFERENCES:

[1] Lindstrom, L. & Jeffries, R. “Extreme programming and agile software development methodologies”. Information Systems Management. 21(13), 41-53. 2005.

[2] Holmstrom, H., Fitzgerald, B., Agerfalk, P., & Conchuir, E. “Agile practices reduce distance in global software development .Information Systems Development”. 23(3), 7-18. 2006.

[3] Tom Mens, Tom Tourw , "A Survey of Software Refactoring", IEEE Transactions On Software Engineering, Vol. Xx,No. Y,Month 2004 Frank Simon, Frank Steinbruckner, Claus Lewerentz,"Metrics Based Refactoring", 2001 IEEE

[4] Frank Simon, Frank Steinbruckner, Claus Lewerentz, "Metrics Based Refactoring", 2001 IEEE

[5] Mann, C. & Maurer, F.” A case study on the impact of scrum on overtime and customer satisfaction.”, Proceedings of the Agile development Conference (ADC’05). Denver, CO. 70-79. 2005.

[6] M. Kleyman, S.Tyszberowicz, A.Yehudai, “Refactoring Aspects into Java Code” in 2007 IEEE International Conference on Software – Science, Technology and Engineering

[7] K.Ush,N.Poonguzhali ,E.Kavitha, “A Quantitative Approach for Evaluating the Effectiveness of Refactoring in Software Development Process” in International Conference on Methods and Models in Computer Science, 2009

[8] S.A.M.Rizvi, Zeba Khanam ,“A Methodology for Refactoring Legacy Code” , 978-1-4244-8679-3/11/$26.00 ©2011 IEEE

[9] Karim M. Zaki, Ramadan Moawad, “A Hybrid Disciplined Agile Software Process Model”

[10] A. Ahmed, S. Ahmad, Dr. N. Ehsan, E. Mirza, S. Z. Sarwar, “Agile Software Development: Impact on Productivity and Quality”

[11] Manjunath K N, Jagadeesh J, Yogeesh M, “Achieving quality product in a long term software product development in healthcare application using Lean and Agile principles” , 978-1-4673-5090-7/13/$31.00 ©2013 IEEE

[12] Bart Du Bois, Pieter Van Gorp, Alon Amsel, Niels Van Eetvelde, Hans Stenten, and Serge Demeyer, Tom Mens, “A Discussion of Refactoring in Research and Practice”

[13] Raimund Moser, Alberto Sillitti, Pekka Abrahamsson, and Giancarlo Succi ,“Does Refactoring Improve Reusability?” in M. Morisio (Ed.): ICSR 2006, LNCS 4039, pp. 287– 297, 2006. © Springer-Verlag Berlin Heidelberg 2006

[14] Jeffery A. Livermore, “Factors that Impact Implementing an Agile Software Development Methodology”

[15] Boehm, B. & Turner, “R. Management challenges to implement agile processes in traditional development organizations” IEEE Software. 22(5), 30-40. 2005.

[16] http://www.mountaingoatsoftware.com/agile

[17] M. H. Halstead, "Elements of software science", Operating and Programming Systems Series, 7,1977.

[18] K. D. Welker and P. W. Oman, "Software maintainability metrics models in practice. Crosstalk" - The Journal of Defense Software Engineering, 8(11): 19-23, 1995.

[19] M. Lorenz and 1. Kidd, "Object-oriented Software Metrics", Prentice Hall Object-Oriented Series, 1994

[20] F. Brito eAbreu, M. Goulao, and R. Estevers, "Toward the Design Quality Evaluation of OO Software Systems," Proc. Fifth Int'l Conf. Software Quality, 1995.

[21] N. Zabkar, V. Mahnic, "Using COBIT indicators for measuring Scrum-based software development", WSEAS Transactions on Computers, vol. 7, no. 10, pp. 1605-1617, 10 2008.

[22] C. T. Fitz-Gibbon, Bera Dialogues: 2, Performance Indicators, Clevedon, England, Multilingual Matters, 1990, pp. 111.

[23]http://uxdesign.smashingmagazine.com/2011/10/20/comprehensive-review-usability-user-experience-testing-tools/

[24]http://stackoverflow.com/questions/16651364/nexus-4-android-tap-to-scroll-issue