International Journal of Engineering Research & Management Technology ISSN:2348-4039

Email:editor@ijermt.org November-2017 Volume 4, Issue-6 www.ijermt.org

Agile Software Quality of Design Risk Assessment Using Fuzzy logic

Anand Kumar Rai,
Dept. of Computer Science
Mumtaz Post Graduate College,
Lucknow

Karuna Shankar Awasthi, Santosh Kumar and Laxmi Shankar Awasthi
Dept. of Computer Science Lucknow
Public College of Professional Studies
Lucknow

ABSTRACT:

Agile software development advocates less documentation and less design but the software design cannot be ignored completely. Software Design is the model that a development team builds and maintains. High quality design makes for an application that is easy to understand and change as new requirements are discovered. Traditionally, the team has one shot to get the design right and then it degrades over time as it is patched over time. Agile practices, however, give an alternative; using practices like test driven development and refactoring teams are now able to continuously improve the design of their system. Many organizations are shifting towards agile software development techniques as compare to heavy weight practices. But in this scenario, still there are issues related with the quality.

KEYWORDS: Agile Software, Fuzzy Logic, Risk Assessment, Quality, MATLAB.

1. INTRODUCTION

Agile methodologies have played a crucial role in software development success in recent years as compared to heavy weight methodologies [6][7]. Because many organizations report the inability of traditional development methods to handle the scaling as well as increased software complexity during software delivery and it affects the quality aspects so there is an urgent need to respond quickly to these functions and problems so that projects can achieve their goals in terms of cost, time and quality. As our research aim is to look for quality factors of agile software development, this paper has put more stress on quality metrics and to analyze them in order to provide better outcomes [9][10]. In traditional development techniques like waterfall model, spiral model, incremental model etc., at the end of every phase, there were reviews like design review, code review and inspections so that errors can be identified as early as possible and can be corrected to improve the quality of product. Now, many organizations have shifted from traditional software development to modern development, in this case traditional quality assurance techniques are not suitable for modern development as modern development have goals which are iteration oriented and traditional development was phase oriented.

Agile software development gives more stress on collocation of teams as well as face to face communication for iterations & daily team activities. So, there is an urgent need to collaborate the quality assurance activities or techniques with daily team activities so that product quality can be improved. Many large organizations are adopting agile methodologies because these methodologies support flexibility, welcome changes at any stage, light documentation and the challenge here is for large organizations to achieve these features as well as to follow quality assurance practices so that product quality will not be at stake.

2. BACKGROUND

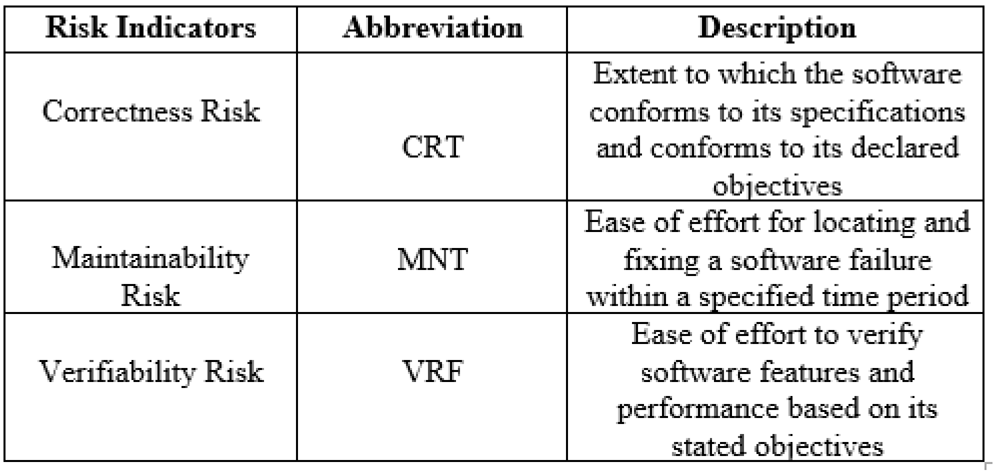
Quality defined as high levels of user satisfaction and low defect levels, often associated with low complexity. The quality of software is assessed by a number of variables. These variables can be divided into external and internal quality criteria. External quality is what a user experiences when running the software in its operational mode. Internal quality refers to aspects that are code-dependent, and that are not visible to the end-user. External quality is critical to the user; while internal quality is meaningful to the developer only [1][2]. Table 1 shows a version of the software quality model [3]. This model categorizes 14 quality factors in three steps of the development life cycle: Quality of design, Quality of performance, Quality of adaptation. This model provides a superior structure of reference to recognize software quality. We used this model throughout the study. In this paper we have shown that how to enhance the criteria of quality factors using agile techniques.

3. DISCUSSION

Quality of design for the agile software risk indicators are identified and shown in given Table 1.0 with description. These three risk indicators need to be managing for the good quality of design. On the basis of literature review and on basis of the agile expert's opinion eight rule bases are created. These three risk indicators hold the qualitative values in nature like low, medium and high. For the precise risk evaluation we need to quantification of these values, for this purpose we use the MATLAB simulator Fuzzy Inference System to resolve this problem. In this study qualitative values of risk indicators are quantified through the fuzzification process which is shown in Figure 2. The qualitative value low ranges between 0.0-0.4, medium ranges between 0.1-0.9 and high ranges between 0.6-1.0 with its membership value between 0-1, which is well elaborated in given figure.

Figure 3, shows the relationship between input indicators value, fuzzy rule base and output value. Table 2.0 shows the rule base which is further used in fuzzy inference system for the fuzzy rule base creation. Figure 4, shows the fuzzy inference rules, on the basis of this rule the rule view is generated as an output which is shown in Figure 5. In this figure vertical lines are appearing with each input factor. The input value can be fixed by this line. This line gives the crisp value of the input factors; finally accordingly the rule base output crisp value is evaluated. We collected the 10 input/output samples from the rule view, which is shown in Table 3.0. Figure 6, elaborated the impact on quality of design on the basis of risk indicators crisp value.

Table1.0 Risk Indicators Quality of Design Agile software



Risk Indicators	Abbreviation	Description
Correctness Risk	CRT	Extent to which the software conforms to its specifications and conforms to its declared objectives
Maintainability Risk	MNT	Ease of effort for locating and fixing a software failure within a specified time period
Verifiability Risk	VRF	Ease of effort to verify software features and performance based on its stated objectives

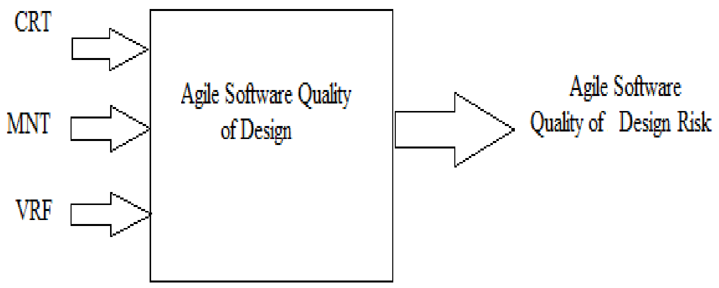
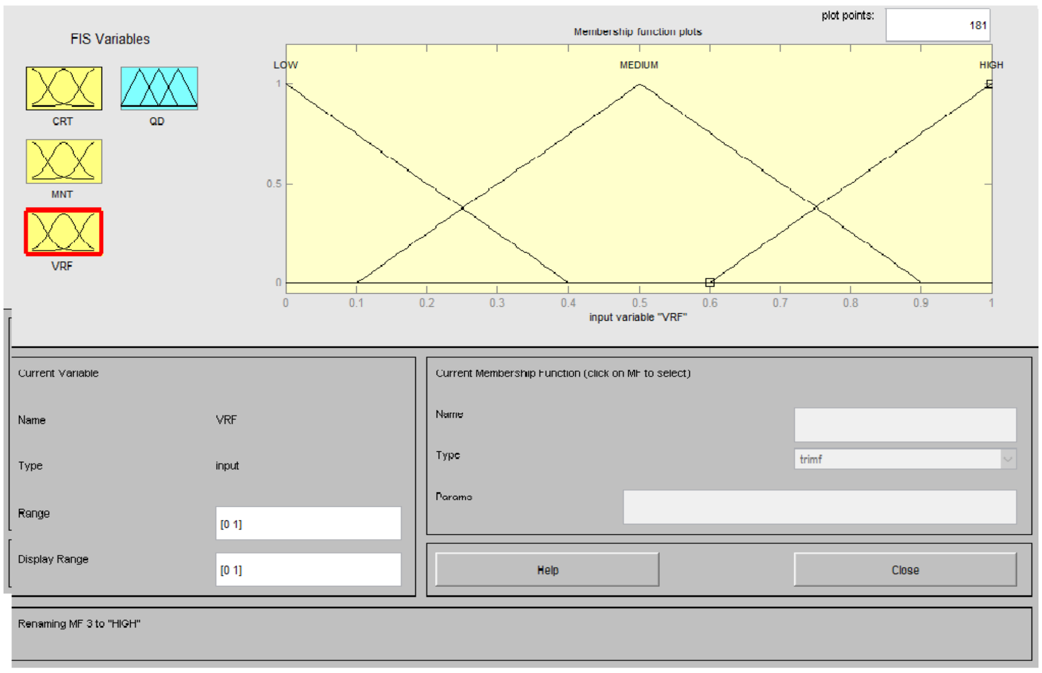

CRT
MNT
VRF
Agile Software Quality of Design
Agile Software Quality of Design Risk

Figure 1. Agile Software Quality of Design Risk evaluation

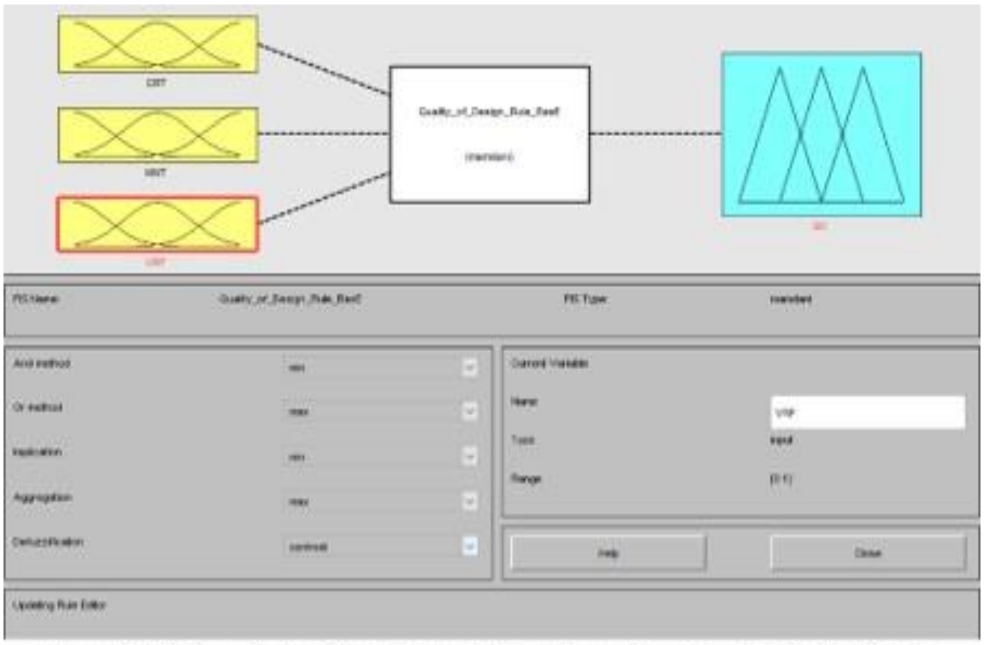


Current Variable
Name VRF
Type input
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name
Type trimf
Params

Renaming MF 3 to "HIGH"

Figure 2 Agile Software Quality of Design Factors Fuzzification



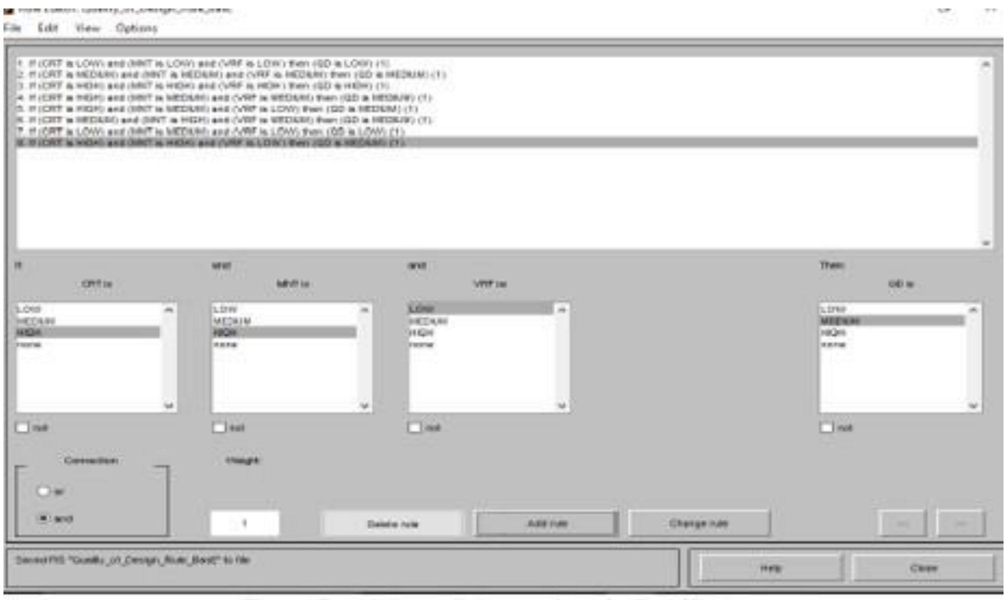
FIS Name: Quality_of_Design_Rule_Base FIS Type: mamdani

And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

Current Variable
Name VRF
Type input
Range [0 1]

Updating Rule Editor

Figure 3 Agile Software Quality of Design Factors and Fuzzy Inference System to evaluate Quality of Design



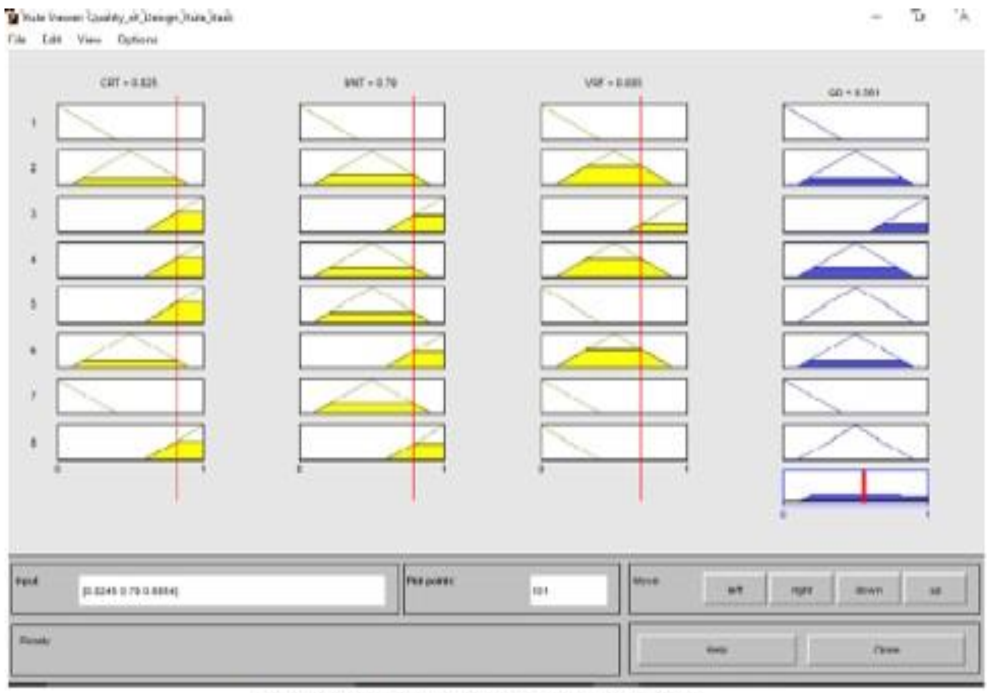
1. If (CRT is LOW) and (MNT is LOW) and (VRF is LOW) then (QD is LOW) (1)
2. If (CRT is MEDIUM) and (MNT is MEDIUM) and (VRF is MEDIUM) then (QD is MEDIUM) (1)
3. If (CRT is HIGH) and (MNT is HIGH) and (VRF is HIGH) then (QD is HIGH) (1)
4. If (CRT is HIGH) and (MNT is MEDIUM) and (VRF is MEDIUM) then (QD is MEDIUM) (1)
5. If (CRT is HIGH) and (MNT is MEDIUM) and (VRF is LOW) then (QD is MEDIUM) (1)
6. If (CRT is MEDIUM) and (MNT is HIGH) and (VRF is MEDIUM) then (QD is MEDIUM) (1)
7. If (CRT is LOW) and (MNT is MEDIUM) and (VRF is LOW) then (QD is LOW) (1)
8. If (CRT is HIGH) and (MNT is HIGH) and (VRF is LOW) then (QD is MEDIUM) (1)

Saved FIS "Quality_of_Design_Rule_Base" to file

Figure 4 Fuzzy Inference Rules to evaluate Quality of Design

Table 3.0 Quality of Design and its Factors crisp value

CRT	MNT	VRF	QD
.109	.103	.142	.155
.109	.503	.507	.500
.242	.637	.301	.440
.381	.797	.646	.500
.679	.657	.652	.522
.738	.710	.745	.563
.818	.770	.838	.672
.911	.850	.904	.856
.957	.930	.977	.866
1.00	1.00	1.00	.870



Input: [0.8245 0.79 0.8854]

Ready

Figure 5 Agile Software Quality of Design Rule View

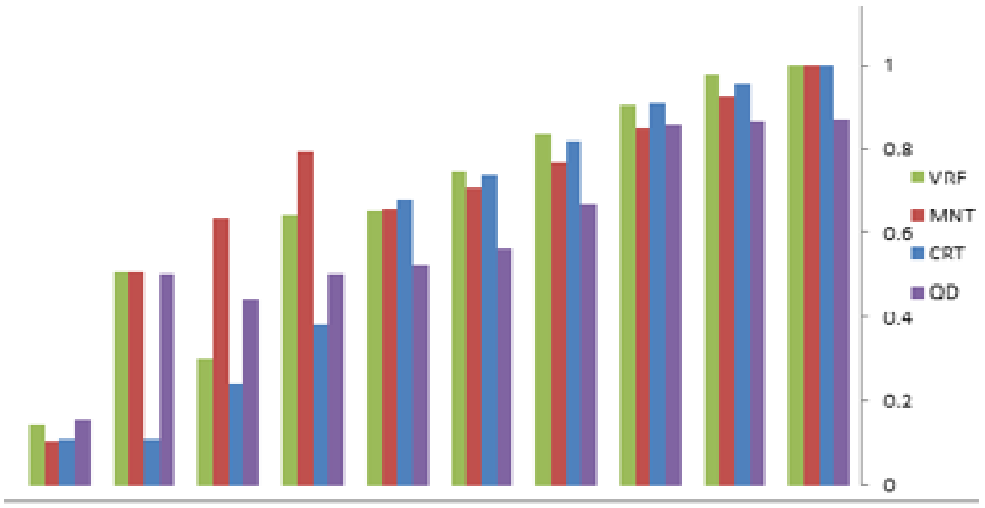


Figure 6 Agile Software Quality of Design Risk Analysis

4. CONCLUSION & FUTURE WORK

In this study the agile software quality of design risk evaluated based on the 3 risk indicators identified and used as input to fuzzy logic Matlab simulator based model to assess the quality of design risk. In this research qualitative data of the agile software design risk converted into the quantitative form using the Fuzzy Inference System. Quantitative data helps in assessing the risk indicators precise values to assess the agile software design risk. We found that Correctness risk is highly sensitive.
In future few more risk indicators can be identified and more rule base can be created for more precise risk evaluation. Further AI based tools can be applied to automate the agile software design risk assessment.

5. REFERENCE

[1] Abrahamsson, P., Conboy, K., Wang, X., 2009. 'Lots done, more to do': the currentstate of agile systems development research. European Journal of InformationSystems 18, 281–284.
[2] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., 2002. Agile software developmentmethods: review and analysis. VTT Technical report, p. 107.
[3] Acuna, S.T., Gomez, M., Juristo, N., 2009. How do personality, team processes andtask characteristics relate to job satisfaction and software quality? Informationand Software Technology 51, 627–639.
[4] Agarwal, A., Shankar, R., Tiwari, M.K., 2006. Modeling the metrics of lean, agile andleagile supply chain: an ANP-based approach. European Journal of OperationalResearch 173, 211–225.
[5] Ågerfalk, P., Fitzgerald, B., Slaughter, S., 2009. Introduction to the special issue:flexible and distributed information systems development: state of the art andresearch challenges. Information Systems Research 20, 317.
[6] Arisholm, E., Gallis, H., Dyba, T., Sjoberg, D.I.K., 2007. Evaluating pair programmingwith respect to system complexity and programmer expertise. IEEE Transactionson Software Engineering 33, 65–86.
[7] Balijepally, V., Mahapatra, R., Nerur, S., Price, K.H., 2009. Are two heads better thanone for software development? The productivity paradox of pair programming.MIS Quarterly 33, 91–118.
[8] Bellini, E., Canfora, G., Garcia, F., Piattini, M., Visaggio, C.A., 2005. Pair designingas practice for enforcing and diffusing design knowledge. Journal of SoftwareMaintenance and Evolution-Research and Practice 17, 401–423.
[9] Boehm, B., 2002. Get ready for agile methods with care. IEEE Computer 35, 64–69.
[10] Chan, F.K.Y., Thong, J.Y.L., 2009. Acceptance of agile methodologies: a critical reviewand conceptual framework. Decision Support Systems 46, 803–814.
[11] Choi, K.S., Deek, F.P., Im, I., 2008. Exploring the underlying aspects of pair pro-gramming: the impact of personality. Information and Software Technology 50,1114–1126.