Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Improving the Requirement Elicitation Process in Extreme Programming: The QFD Approach

Ms. Nandini Nayar , Ms. Tanu Sharma Angra, Mr. Sushil Kumar Bansal
Compute Science & Engineering,
Chitkara University, India

Abstract— Developing software that completely satisfies the customers' expectations is the main goal of Software Development Methodology. To achieve customer satisfaction, the user requirements must be elicited carefully. There are many problems associated with requirement gathering as the customers cannot explain their expectations, also confusions may arise which may lead to misinterpretation. Therefore, there is a need to analyse their expectations into more specific customer requirements. This paper has proposed a solution in form of an approach which Integrates Extreme Programming and Quality Function Deployment to improve the requirement elicitation process that helps to prioritize the user stories and categorize the risk involved with the user stories.

Keywords— Extreme Programming, User Stories, QFD, House of Quality, Risk

I. INTRODUCTION

Extreme programming is an agile methodology designed to simplify and speed up the software development process in order to satisfy the customer. Five key principles of Extreme Programming (XP) are communication, simplicity, feedback, courage, and quality work [2]. The first framework activity of Extreme Programming is "Planning", in which the customer writes the user stories. The user stories are written by customer, which define the functionality desired for the software and what the developer must provide to the customer. Traditional XP story card framework or template is not well defined for the requirements elicitation. It supports to write requirements or user needs in two to three sentences and it does not discover any information rather than user functionality. Different stakeholders have different needs. End user has rarely a picture of a clear system to write down the user stories. This will lead to problems like requirements conflicts, missing requirements, and ambiguous requirements etc, and not address non-functional requirements from exploration phase. Due to this reason they hardly make a decision or predict wrong priority of the requirements or story cards. Two third of the projects fail because of ambiguous and incomplete user requirements and poor quality of the requirements. For small to medium organizations, proper requirements prioritization and selection can make the difference in not only project success or failure but also overall company survivability [1]. Therefore, it becomes essential to prioritize the requirements carefully so that proper assessment of customer needs can be done which will lead to best possible results. If vital user requirements remain anonymous, this could lead to more development cost, more testing cost and the project may run out of schedule. Five of the eight main factors for project failure deal with incomplete requirements, low customer involvement, unrealistic expectations, and useless requirements [3]. The aim of software development methodology is to build a software that fulfils customer's expectations. To achieve customer satisfaction, the user requirements must be elicited carefully. Sometimes, the customer's expectations may be unclear or vague. There is a need to analyse their expectations into more precise customer requirements which requires a disciplined approach and that approach is Quality Function Deployment.

II. QUALITY FUNCTION DEPLOYMENT

Quality Function Deployment (QFD) is a management approach that deals with identifying needs of customers and transforming those needs into plans that would lead to improved customer satisfaction. During requirement gathering, conflicting requirements may exist, which must be identified and must be resolved, thereby, preventing the defects during early stages which would lead to reduced development time. Because QFD concentrates on customer expectations and needs, a considerable amount of effort is put into research to determine customer expectations. This process increases the initial planning stage of the project definition phase in the development cycle. But the result is a total reduction of the overall cycle time in bringing to the market a product that satisfies the customer [4]. The discerning and stringent customer of today is no more satisfied with low-cost products; s(he) demands high quality, high variety/more options, and rapid delivery of a product – all at the lowest possible cost. Since, all four of these objectives are mutually conflicting – i.e. each one drains resources at the cost of the other – there is a dire need for an optimizing tool that helps guide the planning and budgeting process with an eye on customer needs. By virtue of the way it is designed, the QFD methodology assures, with high a degree of confidence, that a company will design and develop its new products or product improvement programs exactly the way that maximizes customer satisfaction using the least resources [5]. QFD provides proper documentation which may be used for future reference. A data base for future design or process

improvements is created. Data that are historically scattered within operations, frequently lost and often referenced out of context, are now saved in an orderly manner to serve future needs. This data base also serves as a training tool for new engineers. Quality function deployment is also very flexible when new information is introduced or things have to be changed on the QFD matrix.[4] QFD is helpful in better capturing and ranking of customer needs and to perform competitive analysis i.e. it will help us to evaluate our software application with the competitors. This would help us to make our software better than the competitors' software. Quality Function Deployment is critical to software development because it provides a methodology for handling the important characteristics of software quality. Quality Function Deployment gives the software development effort a solid foundation for embedding quality into the product. Software quality can take on many different forms (e.g. functionality, efficiency, reliability, usability, maintainability, and portability) [6].

A. Voice of Customer (VoC)

The words used by customer to describe his/her requirements are known as voice of customer. There are various ways to capture VOC, this may include direct discussion with customer or interviewing the customer or conducting surveys. Sometimes, the user expectations can be unclear or vague. Customer requirements are often stated in nontechnical or non-measurable terms. With QFD, these nontechnical terms could be analysed and converted into technical specifications. In QFD, the process requires a multi disciplinary team. With a multi-disciplinary team, design defects that will result in costly prototyping and time re-design can be found and solved in the early stages of design. [7] Therefore, this is the task of QFD team to examine the inconsistent needs and convert them into more precise customer requirements. By using QFD as a planning tool, the inconsistent user stories can be identified, and resolved before development stage, which leads to reduced implementation time and timely completion of software.

B. House of Quality

The House of Quality (HOQ) is the prime planning tool used in QFD that is used to interpret the voice of customer into design requirements that focus on achieving the target values.

III. THE PROPOSED APPROACH

This paper has proposed an approach which integrates Extreme Programming and Quality Function Deployment for improving the process of requirement elicitation. This approach has been adopted in the planning phase of "*WeighStack*", which is a Software Application designed for the company "CW Instruments, Haryana". It is a small scale project. The scope of project is to provide an efficient way of managing all the weighing information by sending commands from the weigh balance to the PC, allowing the user to save the information and to generate customized reports. The planning phase of "*WeighStack*" commenced in April, 2013, which included interview with stakeholders. The end-user needs were identified and were documented.

The various steps were followed to create House of Quality Matrix, which are discussed below:

STEP 1: The requirements were gathered and the user stories were written and were placed on story cards. Requirement elicitation process was carried out by discussion with the customer.

STEP 2: The customer requirements (WHATs) written on story cards were placed on the left side of the HOQ matrix.

STEP 3: The technical descriptors i.e. the software features and engineering characteristics (HOWs) were listed at the top of HOQ matrix.

STEP 4: The relationship matrix was created, forming the body of HOQ that describe the relationship between WHATs and HOWs. The relationship can be strong (●), medium (O) or weak (∇). A blank field indicates that no relationship exists.

Then these symbols were substituted with numbers

● = 9
O = 3
∇ = 1

STEP 5: The Correlation matrix (interrelationship matrix) that is a diagonal matrix at the roof of HOQ was developed to recognize the inter-relationship between HOWs.

The strength of relationship can be positive (+) or negative(-)Correlation matrix is beneficial to analyze which technical descriptors support each other and which contradict each other. Contradicting HOWs signify points where there is a need to make trade-offs. It is essential to recognize these trade-offs and resolve them, such that the problems (inferior quality software, increased costs of software and unfulfilled requirements) can be avoided.

STEP 6: Competitive assessment of customer requirements and technical descriptors was done to identify where the software stands in terms of the competitor company's software.

In this step ranking from 1 to5 was provided (1 for worst and 5 for best).

STEP 7: For prioritizing the User Stories, following values were determined:
1) Importance to customer: signify ranking from 1 to 10 (1 for least important and 10 for most important customer requirement), which was helpful for prioritizing and decision-making process. Consider Figure 1. Here rating has been provided which means that user friendly software is very important to customer so it has been assigned a value of 10 and exporting files in various formats is not so important, therefore it is assigned a value of 5.
2) Target Value: A ranking was provided from 1 to 5 (1 for worst and 5 for best) to determine whether it is required to keep the software as it is, or the software needs any up gradation or the software needs improvement so as to surpass the competitors' software.
3) Scale-up Factor: The scale-up factor was calculated by calculating ratio of target value to software rating (which is provided in customer competitive assessment). It helped in reviewing the current rating of software, target rating and the difference between these two. It helped in analyzing whether this difference was reasonable and is it possible to achieve that target.
4) Sales Point: Relating the customer requirements to sales point of view, the sales point was determined by assigning values from 1(lowest) to 2(highest) that would help in the sale of the software.
*5) Absolute Weight***:** The absolute weight was calculated by multiplying the importance to customer, scale-up factor, and sales point:

Absolute Weight = (Importance to Customer)(Scale-up Factor)(Sales Point) [4]

STEP 8: Risk associated with each user story was analyzed.
Risk Analysis is the conversion of risk data into risk decision-making information. Analysis provides the basis for the project manager to work on the ―right and most critical risks. [8]
In this step, risk was analyzed based on 3 factors: volatility, complexity and completeness.
1) Volatility (deals with the probability whether the user story will change or not). An index from 0 to 2 (0 for low, 1 for medium and 2 for high) was assigned.
2) Completeness factor was used to identify whether the customer provided entire story details or incomplete story details. An index from 0 to 2 was assigned (0 for complete story details, 1 for incomplete story details and 2 for unidentified user stories).
3) Complexity factor was used to identify the degree of difficulty associated with each user story. An index from 0 to 2 was provided (0 for easy, 1 for normal and 2 for difficult).
Based on these 3 factors, Risk Index was calculated by adding these indexes for each row. Risk index value from 0 to 1 indicates low risk, values from 2 to 4 indicate medium risk and values from 5 to 6 indicate that high risk is associated with that particular user story. The risk index has been represented in the House of Quality matrix as L, M and H (which represent Low, medium and high respectively).
Consider the HOQ matrix in Figure 1, it signifies that "adding custom fields" and "easily upgradable" customer requirements have high risks associated with them, which implies that they may take more effort and time to finish and thus they must be done early in the development phase.

STEP 9: Technical Descriptors were prioritized so as to conclude which technical descriptors are most required to meet the needs of the customers and where changes are to be made regarding these technical descriptors. For this purpose, the following values were determined:
1) Degree of Difficulty: A rating was given from 1(for the technical descriptor which is least difficult to implement) to 10 (for the one which is most difficult to implement) so that ability to implement these descriptors can be estimated.
2) Target Value: A ranking was provided from 1 to 5(1 for worst and 5 for best) to determine the value that needs to be achieved to fulfill that particular technical descriptor.
*3) Absolute Weight :*For calculating Absolute weight, numerical values were assigned to symbols in the relationship matrix symbols. The absolute weight for the *j*th technical descriptor was then given by

$$a_j = \sum_{i=1}^{n} R_{ij} c_i$$

where
a_j=row vector of absolute weights for the technical descriptors (j = 1,..., m)
R_{ij}=weights assigned to the relationship matrix (i = 1 ,..., n, j = 1,..., m)
c_i=column vector of importance to customer for the customer requirements (i = 1,..., n)
m=number of technical descriptors
n=number of customer requirements [4]

4)Relative Weight: Then the relative weight for the jth technical descriptor was calculated by replacing the degree of importance for the customer requirements with the absolute weight for customer requirements, using the formula

$$b_j = \sum_{i=1}^{n} R_{ij} d_i$$

where b_j= row vector of relative weights for the technical descriptors (j = 1,..., m)
d_i= column vector of absolute weights for the customer requirements (i = 1,..., n) [4]

The higher values of absolute and relative weights means that more engineering efforts must be made for that descriptor and thus there is a need to focus that where the resources must be allocated for providing quality software. Consider the HOQ matrix in Figure 1, implies that it is very essential that the software must have efficient user interface, implying that this technical descriptor requires higher engineering efforts and may need more resources for designing good quality software.

STEP 10: The House of Quality matrix was created and was discussed with customer. According to customer's feedback, necessary changes were made in the matrix. Then the matrix was verified by the developer to confirm whether all the values and correlations were reasonable. The final HOQ matrix is shown in Figure 1.

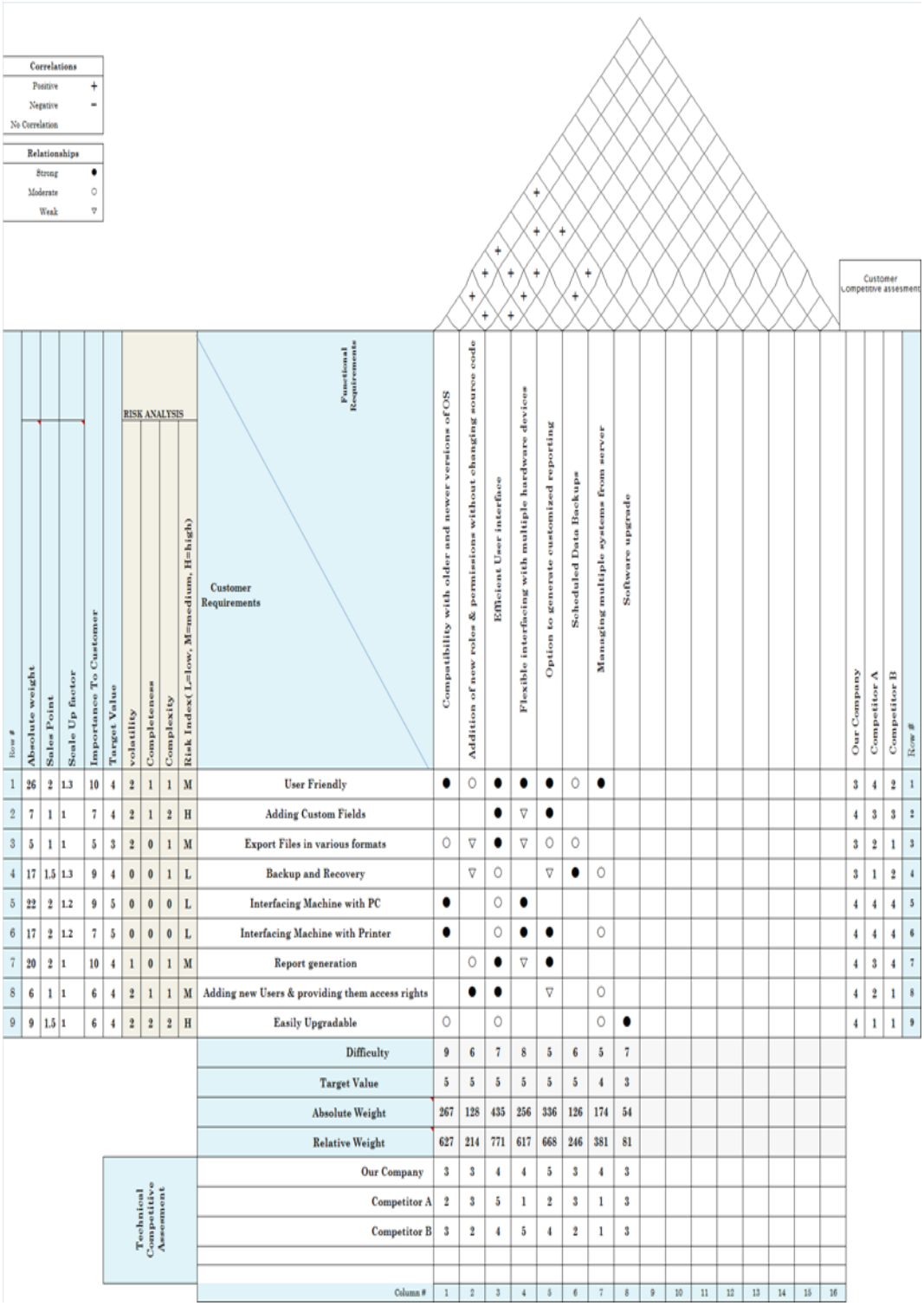

Fig. 1 House of Quality Matrix for WeighStack

IV CONCLUSION

The success of a software project depends on the degree of satisfaction it provides to the customer and how well it meets the customer needs and expectations. This paper has discussed problems associated with user stories and QFD has been used as a tool in planning phase of "WeighStack" Software Application which has been helpful in managing the user stories and to identify the areas where more effort and concentration is required. Also, the addition of "Risk Analysis" column in the House of Quality Matrix is beneficial for analysing the risk associated with each user story, which has helped in well-timed identification of risks. The results of planning stage were discussed with stakeholders of project to make sure that voice of customer was heard. Thus, in this project, the House of Quality has served as an excellent tool for decision making. The use of QFD has also provided sufficient level of documentation to Extreme Programming which can be helpful for reusing the gathered information.

REFERENCES

[1] Chetankumar Patel, and Muthu Ramachandran,"*Story Card Based Agile Software Development*", International Journal of Hybrid Information Technology,Vol.2, No.2, 2009,pp.125-140

[2] Naresh Kumar Nagwani, and Pradeep Singh, "*An Agile Methodology Based Model for Change-Oriented Software Engineering*", International Journal of Recent Trends in Engineering, Vol.1,No.1,2009, pp.128-132

[3] Veerapaneni Esther Jyothi, and K. Nageswara Rao "*Effective Implementation of Agile Practices*", International Journal of Advanced Computer Science and Applications,Vol. 2,No. 3, 2011, pp.41-48

[4] Besterfield Dale H., Dale H. Besterfield, Carol Besterfield-Michna, Glen H. Besterfield, Mary Besterfield-Sacre, Hermant Urdhwareshe and, Rashmi Urdhwareshe, *Total Quality Management*, India: Pearson Education, 2011

[5] Ashok Kumar, Jiju Antony and Tej S. Dhakar, "*Integrating Quality Function Deployment and Benchmarking to Achieve Greater Profitability*", Benchmarking : An International Journal, Vol. 13, No. 3, 2006, pp 290-310

[6] Okanta, Okechukwu Emmanuel, Ojugo, Adimabua Arnold, Wemembu Uchenna Raphael an Ajani Dele, "*Embedding Quality Function Deployment In Software Development: A Novel Approach*", West African Journal of Industrial & Academic Research, Vol.6, No.1 March 2013, pp 50-64

[7] Yonghui CAO, "*Theoretical Model of Software Process Improvement for CMM and CMMI based on QFD*", International Journal of Computer Science Issues, Vol. 10, Issue 1, No. 3 January 2013, pp 614-620

[8] Vasundhara Rathod, Monali Chim, and Pramila Chawan," *An Overview of Software Risk Management Principles*" International Journal of Advanced Research in Computer Engineering & Technology Vol. 1, Issue 3, May 2012, pp51-54

