

Implementation of “XP-QFD” in a Small Scale Project

Nandini Nayar
Chitkara University
Himachal Pradesh

Tanu Sharma
Chitkara University
Himachal Pradesh

Sushil Kr.Bansal
Chitkara University
Himachal Pradesh

Sapna Saxena
Chitkara University
Himachal Pradesh

ABSTRACT

Eliciting requirements is the most crucial part of software development. Deficient, ambiguous and worthless requirements are among the major causes of failure of any project. The process of requirement elicitation in Extreme Programming must include some systematic approach to simplify the process such that requirements elicitation may be carried out efficiently. As a part of this research, an algorithm has been developed for creating House of Quality matrix and an approach named “XP-QFD” which integrates Extreme Programming (XP) and Quality Function Deployment (QFD), has been implemented in a small scale project that helps in delivering quality software by efficient prioritization of user stories and saving data in orderly manner for better decision making.

Keywords

Disaggregation, Extreme Programming, House of Quality, Quality Function Deployment, Risk, User Story, XP-QFD.

1. INTRODUCTION

The term “requirement” defines a bounded characterization of the system scope that can be generated by different stakeholders. It includes the information essential to communicate an understanding of the problem and to support relevant stakeholders in its resolution.[1]

Five of the eight main factors for project failure deal with incomplete requirements, low customer involvement, unrealistic expectations, changes in the requirements, and useless requirements.[2]

Table 1. Main causes of project failure

Problem	%
Incomplete requirements	13.1
Low customer involvement	12.4
Lack of resources	10.6
Unrealistic expectations	9.9
Lack of management support	9.3
Changes in the requirements	8.7
Lack of planning	8.1
Useless requirements	7.5

[2]

A successful project requires active participation of the stakeholders, such that they can share their views, and this would result in more stable requirements. Lack of communication may lead to missing or inconsistent requirements. Table 1 clearly depicts that deficient, ambiguous and worthless requirements are among the major causes of failure of any project. So it can be concluded that

one of the most critical phase of a project is to gather requirements effectively.

Successful companies in today’s dynamic global economy are those that are able to efficiently design, develop, and manufacture products that will be preferred by customers over those offered by competitors [3]

2. EXTREME PROGRAMMING

Extreme Programming (XP) is a methodology for software development that focuses on high customer integration, extensive testing, code-centered development and documentation, refactoring and paired programming [4] It is a set of twelve practices and four principles, which makes XP successful and well known among all the agile software development methods. [5]

XP has attracted attention because of its fierce denial of many well-accepted software engineering practices considered as a sound approach to the development of intensive software systems. In the core of XP practices are programming activities and strong emphasis on oral communications, automated tests, pair programming, storytelling culture and collective code-ownership at any time in the XP project. [6]

2.1 User Story

User story is a brief description of a need, a feature or a desire from the point of view of a specific user role in the software project. [7]

User stories are written by customer to signify what should be accomplished by the desired software, which doesn’t provide any thorough or formal specifications of the system. The foremost task of developer is to understand these stories, then the stories are assessed and an estimate is provided to the customer. Then the user stories are prioritized. The user stories are placed on Story cards

2.2 Problems in XP

Customers rarely have an apparent idea for the software to be developed, which may lead to missing requirements. Customer’s requirements may vary daily. There may be many reasons: the client may come up with more prospects of the new system to be developed; or due to changes in client’s organization; and most of the times, when customer sees the first part of the running project, he/she may have some new suggestions.

Mostly, the customers define their requirements in natural language, but it doesn't provide an adequate structure to make a good set of requirements. It is very difficult to manage the requirements that are unstructured, which makes decision-making process difficult or may lead to wrong prioritization of user stories. It is very crucial to select suitable requirements from the set of inconsistent requirements. Therefore, the process of requirement elicitation in Extreme Programming process must include some systematic approach to simplify the process such that requirements elicitation may be carried out efficiently and that approach is Quality Function Deployment.

3. QFD

Quality function deployment (QFD) is a powerful customer-oriented tool to facilitate the customer needs. QFD approach is helpful to prioritize and implement the new layout solutions at the onset of the design stage before the actual layout implementation. [8]

Quality Function Deployment can help to quell many of the fears executives face when discussing a move to an Agile development methodology. It helps them to know that there is a plan in place for development, and that their development teams are not going to aimlessly add features to their applications as they see fit. Additionally, it helps stakeholders to manage their feature pools from a high-level vantage point, without devoting endless hours to reviewing and re-reviewing individual features. [9] QFD performs evaluation and comparisons of competitor's products, also considering all precedent response of the customer which helps to define the most essential needs. The inconsistent requirements can be recognized before coding begins. This decreases the time required for retooling and operator training.

Quality Function Deployment (QFD) can be used as an effective tool for capturing and refining the requirements. When applied to a project, QFD will enable its success. More specifically, QFD will: [10]

- (1) Help to improve quality of product by focusing on customer requirements up-front and throughout the development cycle;
- (2) Improve the communication and help in efficient decision-making.
- (3) Reduce costs of projects, by enabling concurrent engineering and by reducing costs associated with late-in-the life-cycle rework, [10]
- (4) Enable cataloguing of key performance requirements, for parameter-based modeling of the target application and systematic reuse of requirements across projects [10]

QFD approach has been used as a tool for defining new products, as well as for improving existing products. In substance, QFD builds the relationship between "target" (what to do) and "measure" (how to do) [11]

The House of Quality (HOQ) is the prime planning tool used in QFD to interpret voice of customer into design requirements that focus on achieving the target values [12]

4. THE PROPOSED APPROACH:

An approach was proposed in [12] which integrates Extreme Programming and Quality Function Deployment for improvising the requirements gathering in Extreme Programming and analyzing risk associated with each user story. This approach has been implemented in elicitation phase of "WeighStack" software application. In order to enhance the capability of the above approach, an algorithm has been proposed, which also helps to prioritize user stories, and analyze risk, along with a new column in HOQ which helps to analyze whether the user story needs to be disaggregated or not. This approach has been named as "XP-QFD".


Sometimes it is difficult to implement a story in a single iteration, and then these stories are split into smaller set of user stories. In this research paper, a new column, i.e. "disaggregation" has been added to the HOQ matrix, along with user stories, which indicates whether there is a need to disaggregate that particular user story or not. If a user story can be completed in a single iteration, then "0" is assigned, and if it needs to be split into more than 1 story, then a "1" is assigned. The HOQ matrix has been shown in Fig.1.

4.1 Algorithm:

A set of user stories U and Software characteristics (technical Descriptors) S is given. Put U at left side of H.O.Q and S at top of H.O.Q. The variable C denotes Competitive assessment of U and S, I denotes Importance to customer. The target value for U is denoted by T1. For rating of software, SF i.e. Scale up factor is determined. Sales Point is denoted by SP. The algorithm analyses the risk associated with U based on 3 factors: Volatility (V), Completeness(C), Complexity (Cx), based on which, Risk Index (RI) is determined. The variables D and T2 denote degree of difficulty and target value for S respectively.

1. [Find Relationship R between U and S.]


If R=strong, then:

Set R:= 

Else if R=medium, then:

Set R:=O

Else if R=weak, then:

Set R:= 

Else:

Set R:=Blank

2. [Find Strength of Relationship SR.]

If SR= positive, then:

Set SR:= +

Else If SR=negative, then:

Set SR:= -

Else:

Set SR:=Blank

3. Read: C.

4. Read: I.

5. Read: T1.

6. Set SF:= T1/ C.

7. Read: SP.

8. Set ABSOLUTE WEIGHT:= I*SF*SP.

9. Set INDEX:= V+C+Cx.

If INDEX=0 to1, then:

Set RI:=L

Else If INDEX=2 to 4, then:

Set RI:=M

Else If INDEX=5 to 6, then:

Set RI:=H.

10. If Iteration > 1, then:

Set DISAGGREGATION:=1

Else:

Set DISAGGREGATION:=0

11. Read: D.

12. Read: T2.

13. Set a_j : = row vector of absolute weight for S(j=1 to m)

Set R_{ij} : = weight of R(i=1 to n and j=1 to m)

C_i = column vector of I(i=1 to n)

m= number of S

n=number of U [13]

14. Set Absolute Weight:=
$$a_j = \sum_{i=1}^n R_{ij} C_i$$
 [13]

15. Set b_j : = row vector of relative weights for S (j = 1 to m)

Set d_i : = column vector of absolute weights for U (i = 1 to n)

Set Relative Weight:=
$$b_j = \sum_{i=1}^n R_{ij} d_i$$
 [13]

Based on the above algorithm, House of Quality Matrix has been created for *WeighStack* Software Application, which has an additional column i.e. Disaggregation as shown in Fig. 1. The values of Absolute Weight and Relative Weight help in prioritization of user stories and corresponding technical descriptors.

4.2 Acceptance testing:

After each release, Acceptance tests were created to test whether the user stories were correctly implemented, and customers verified their correctness. In this case, the acceptance tests indicated that the product delivered is satisfactory to customers. Although, according to customer's feedback, some changes were also suggested, but overall acceptance level of customer was positive

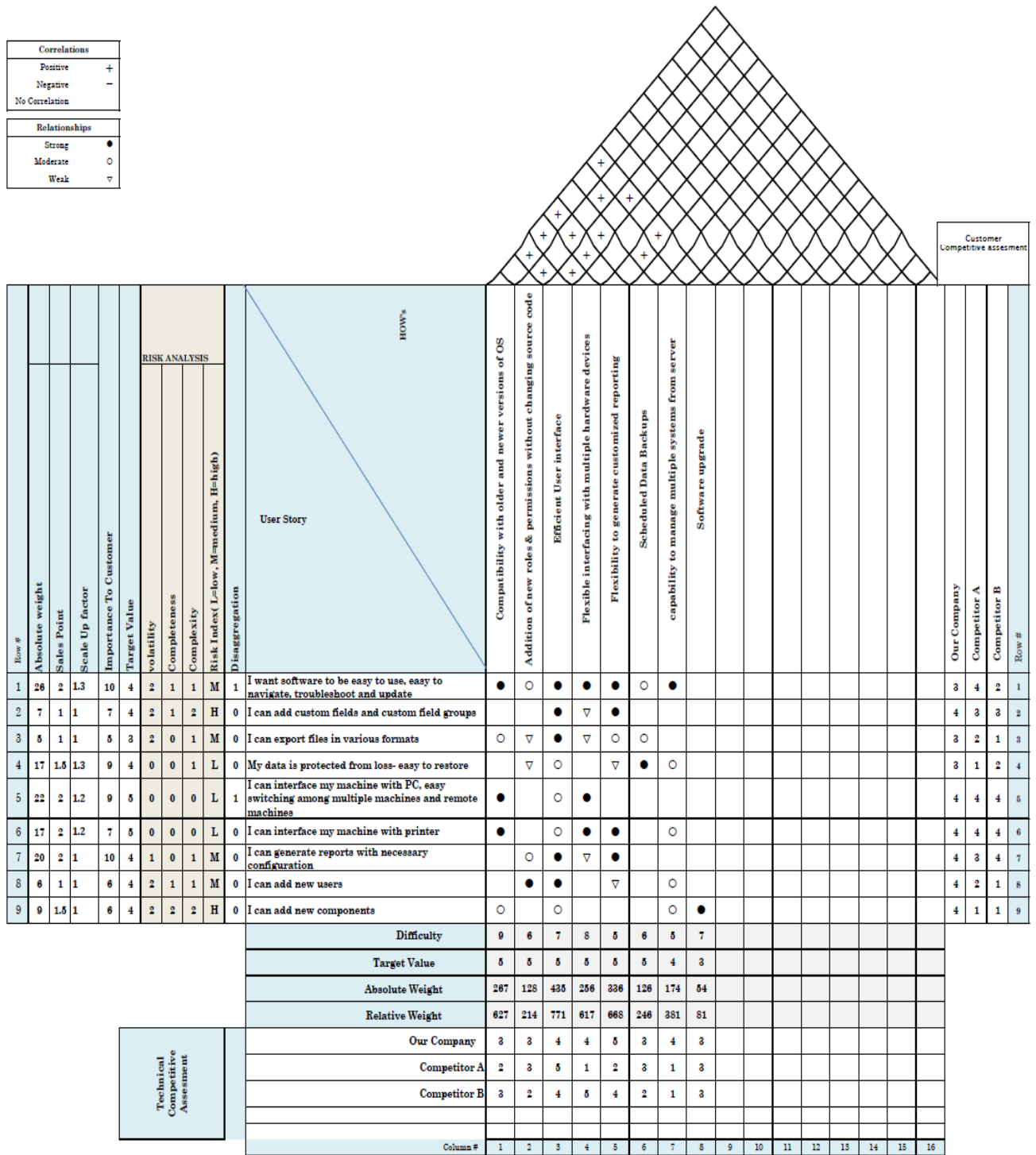
5. QUESTIONNAIRE

Subjective evaluation technique was followed, in which a questionnaire was prepared. All the stakeholders of the project were asked to fill the questionnaire and they had to rate how strongly they agreed or disagreed with the corresponding statements.

1. Was the requirements management effective?
2. Were the requirements elicited using a precise method?
3. Adequate customer involvement was there?
4. The requirements related to the software were scoped to fit within the methodology?
5. Were the requirements complete enough to recognize the sub-problem types within the project?
6. Was there any repository for organizing the requirements properly?
7. Were the suggestions for enhancement or improvement accepted by the customers or the RE team?
8. If any change occurred in customer requirements, was it feasible to accommodate change in HOQ matrix?
9. Were the customers able to identify the most important requirements?
10. Were the important quality attributes considered?
11. Was an analytical approach followed to rate the risks?

Each question had 4 options associated with it, namely: Strongly agree/ Agree/ Disagree/ strongly disagree and No Comments. The evaluators were asked to choose 1 option for each question. Each option had a score associated with it. The scores associated are stated as: Strongly agree: score= 4, agree: score=3, disagree: score= 2 , strongly disagree :score=1 and no comments: score=0.

Adding scores from each item, a "Final score" was calculated. Based on evaluation of each stakeholder, an average score was calculated, which is the ratio of adding final score of each questionnaire to the total number of evaluators. The average score obtained was 37.4.



QF

Fig. 1. House of Quality Matrix for WeighStack

6. COMPARATIVE ANALYSIS

Table 2 shows the comparison of *XP-QFD* with the *3C approach*

Table 2: Comparative Analysis

Feature	3C	XP-QFD
Prioritize User Stories	Limited	✓
Valuable small releases	✓	✓
Categorization of Risk involved with each user story		✓
Provides baseline for Competitive Analysis		✓
Acceptance Tests	✓	✓
Provides basis for Cost Estimation	✓	
Resolve conflicting requirements		✓

7. CONCLUSION

XP-QFD has been implemented in WeighStack software application. This approach has been helpful to improve the decision making process and for prioritizing user stories, and identifying risk associated with each user story. The “disaggregation” column helps to identify which user stories may take more than one iteration to be completed. The conflicting requirements can be identified and resolved. The data can be saved in orderly and structured manner. The results of acceptance testing and questionnaire indicate that this approach has been able to satisfy the customer to a great extent.

8. REFERENCES

- [1] Orlando Lopez, “Requirements Management”, Journal of Validation Technology, pp. 78-86, Spring2011
- [2] Veerapaneni Esther Jyothi and K. Nageswara Rao, ”Effective Implementation of Agile Practices”, International Journal of Advanced Computer Science and Applications, Vol. 2, No.3, pp. 41-48, 2011
- [3] John J. Cristiano, Jeffrey K. Liker, and Chelsea C. White,” Key Factors in the Successful Application of Quality Function Deployment (QFD)” IEEE Trans. Engineering Management, Vol. 48, No. 1, pp. 81-95, , February 2001
- [4] J. Kivi, D. Haydon, J. Hayes, R. Schneider, and G. Succi, "Extreme Programming: a University Team Design Experience," Canadian Conference Electrical and Computer Engineering, pp. 816-820, March 2000
- [5] Chetankumar Patel, and Muthu Ramachandran,”Story Card Based Agile Software Development”, International Journal of Hybrid Information Technology, Vol.2, No.2, pp.125-140, 2009
- [6] R. Juric. "Extreme programming and its development practices", Proc. 22nd International Conf. Information Technology Interfaces, pp. 97-104, 2000
- [7] Victor Schetinger, Cesar Souza, Lisandra Manzoni, Cesar Tadeu, “User Stories as activities for game development”, Proc. SB Games, pp. 1-4, November 2011
- [8] Ali, Mas Alina Mohd,”Assimilating Quality Function Deployment (QFD) with QUEST® analysis for facility layout redesign of Handwork Section”, Science and Social Research (CSSR) International Conference , pp. 985 – 990, Dec 2010
- [9] John Livingston,”QFD and Agile Feature Prioritization”, <http://www.qfdweb.com/?p=23>. 2007
- [10] T.-L. Tran and J. S. Sherif, "Quality Function Deployment (QFD): an effective technique for requirements acquisition and reuse", Second IEEE Software Engineering Standards Symposium. IEEE, pp. 191-200, August 1995
- [11] Yu Zhao, Xiu Li and Wenhua Liu , “The application of extend QFD in BPR”, IEEE International Conference, pp. 4742 - 4747 , Vol. 5 , October, 2003
- [12] Nandini Nayar, Tanu Sharma Angra, Sushil Kumar Bansal,”Improving the requirement Elicitation process in Extreme Programming: The QFD Approach”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, pp. 873-877, June 2013
- [13] 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, pp. 259-287, 2011