EIJEST

*The Egyptian Int. J. of Eng. Sci. and Technology*
*Vol.14, No. 3 (Sept. 2011)*

# A NEW MEHODOLOGY USING QFD FOR TRACING REQUIREMENTS IN OBJECT-ORIENTED SOFTWARE DESIGN PROCESS*

Ahmed A. Attia[(1)] and Mohamed A-k. Soliman[(2)]**

[(1)] National Authority for Remote Sensing and Space Sciences, Egypt
[(2)] Computer Eng. Dept., Faculty of Engineering, Zagazig University, Egypt

## ABSTRACT

It has been widely acknowledged that software products should be developed based on customer requirements in order to achieve a high level of software quality and customer satisfaction. Tracing customer requirements and their impacts through the software development life cycle is not a well-explored area. In this paper, a framework is presented that uses quality function deployment (QFD) to trace customer requirements explicitly through various phases, such as requirements elicitation, analysis, and design in object-oriented software development, by assessing their impact on software artifacts of the next stages. QFD helps visualize the complete tracing from customer requirements to class designs. Degrees of impact are clearly calculated and presented in QFD automatically using a simple software (an excel sheet). The Analytical Hierarchy Process (AHP) is used to prioritize and calculate the importance index of customer requirements and their impact on design stages. In traditional QFD, the correlation between customer requirements and technical requirements is determined by the members of a design team using linguistic expressions (e.g. weak, average, and strong). These linguistic terms are then scaled into crisp values (e.g. 1-3-9) for the ranking of each alternative. This crisp assessment for correlation evaluation in QFD analysis has difficulty coping with uncertainty among design team members. Therefore, fuzzy sets are adapted in this paper. TRIZ methodology (Theory of Inventive Problem Solving) is used to solve for contradicting technical requirements in Object-Oriented design process. An ATM machine object-oriented software design example is developed to illustrate and validate the framework.

**KEY WORDS**: Analytical hierarchy, quality function, house of quality, software quality, fuzzy sets, object-oriented software design, traceability, customer requirements, subsystem, Unified Modeling Language (UML), Class Diagram.

## UNE NOUVELLE METHODOLOGIE QFD UTILISER POUR TRAÇAGE EXIGENCES ORIENTEE OBJET PROCESSUS DE CONCEPTION DE LOGICIELS

## RÉSUMÉ

Il a été largement reconnu que les produits logiciels doivent être développés en fonction des besoins des clients afin d'atteindre un haut niveau de qualité des logiciels et la satisfaction du client. traçage exigences des clients et de leurs impacts à travers le cycle de vie du développement logiciel n'est pas une zone bien explorée. Dans ce papier, un cadre est présenté qui utilise Qualité Fonction Déploiement (QFD) pour tracer les exigences des clients explicitement par différentes phases, telles que l'élicitassions des exigences, d'analyse et de conception dans le développement logiciel orienté objet, en évaluant leur impact sur les artefacts logiciels de prochaines étapes. QFD permet de visualiser la traçabilité complète des besoins du client à des conceptions de classe. Degrés d'impact sont clairement calculés et présentées dans QFD automatiquement en utilisant un logiciel simple (une feuille Excel). Le processus de hiérarchie analytique (AHP) est utilisée pour hiérarchiser et calculer l'indice de l'importance des besoins des clients et leur impact sur les étapes de conception. En QFD traditionnelle, la corrélation entre les exigences des clients et des exigences techniques est déterminée par les membres d'une équipe de conception en utilisant des expressions linguistiques (par exemple, faible, moyenne et forte). Ces termes linguistiques sont ensuite mis à l'échelle en valeurs nettes (par exemple 1-3-9) pour le classement de chaque solution. Cette évaluation nette pour l'évaluation dans l'analyse de corrélation QFD a de la difficulté face à l'incertitude parmi les membres de l'équipe de conception. Par conséquent, les ensembles flous sont adaptés dans ce papier. TRIZ méthodologie (Théorie de résolution des problèmes inventifs) est utilisé pour résoudre pour contredire les exigences techniques dans les processus de conception orientée objet. Un guichet automatique orientée objet par exemple un logiciel de conception est développée pour illustrer et de valider le cadre
.

**MOTS CLÉS:** processus de hiérarchie analytique (AHP), Qualité Fonction Déploiement (QFD), maison de qualité (HOQ), déploiement de la fonction des logiciels de qualité (SQFD), ensembles flous, la conception de logiciels orientés objet, de la traçabilité, les exigences des clients, les exigences du système, sous-système exigences, les exigences de la classe, Langage Modelées Unifie (LMU), Diagramme de classe, cas d'utilisation.

* Received: 23/ 1 / 2011, Accepted: 4 / 8 / 2011 (Original Paper)
** Contact author ( mamas2000@gmail.com)

# 1. INTRODUCTION

Traditionally, traceability analysis provides linkages between requirements and design items. Although the linkage is necessary, it is not enough to develop software products with high customer satisfaction.

A tradeoff analysis that can be done to select a suitable requirement prioritization method and the results of trying one method, AHP is described by Nancy ([1], [2]). AHP was developed by Thomas Saaty [3] and applied to software engineering by Karlsson [4] and Karlson and Ryan [5]. AHP is a method for decision making in situations where multiple objectives are present. This method uses a pair-wise comparison matrix to calculate the relative importance of software requirements. By using AHP, the requirements engineer can also confirm the consistency of the result. AHP can prevent subjective judgment errors and increase the likelihood that the results are reliable.

The limitations of QFD house of quality in its original form and also the advantages of automating it are identified [6]. It simplifies the construction of the house of quality by creating it on Microsoft Excel. The standard format of the automated house of quality (AHOQ) created has been tested to be reusable and extendable for multiple applications. It saves time and effort and ensures accurate calculations of absolute and relative values.

A method for mapping and prioritizing customer requirements into functional features and technical modules to optimize market performance is described [7]. Although the quality of a product can be dramatically improved through a QFD exercise, the traditional crisp scoring approach has a major drawback. To overcome this problem, fuzzy scoring for linguistic terms is proposed. The implementation case of a low-end digital camera design shows that the result of the proposed fuzzy QFD model can reflect the certainty level of an evaluation term, which is designated for each correlation of customer requirements and technical requirements considered in design.

How different requirements have different impacts on design items is analyzed [8]. A design item that is impacted by more important requirements deserves more attention than a design item that is impacted by fewer important requirements. Otherwise, if more resources are given to design items with small impacts on the requirements, it is a waste of limited resources. The issue of requirements traceability is addressed by assessing the degrees of impact with the help of quality function deployment (QFD). QFD, which was developed more than 30 years ago in Japan, is a methodology that incorporates the voice of the customer into a product, and it is an excellent method for assuring that customers receive high quality products [9]. QFD is a process that transforms the desires of the customer at all levels into the implementation of a product. Software quality function deployment (SQFD) is the application of QFD to software production, which focuses on improving the quality of both the software development process and the product [10]. The ultimate goal is no longer zero-defect software, but rather good software that provides very high customer satisfaction. SQFD has been applied to the improvement of software quality focusing on three phases of the software development life cycle. It uses a set of house of quality (HoQ) matrices to translate customer requirements into system, subsystem, and class requirements.

This paper addresses the issue of requirements traceability by assessing the degrees of impact with the help of quality function deployment (QFD), House of Quality (HoQ). Analytical Hierarchy Process (AHP) is used for the purpose of prioritizing the customer requirements during implementing the House of Quality. The house of quality has been automated using an excel sheets thereby saving effort and time by using automated calculations. Besides, it gives the possibility of adding more customer or technical requirements to the HoQ matrix. Fuzzy sets and the concept of linguistic variables are adapted in this research. This model uses a three-phase set of house of quality (HoQ) matrices to translate customer requirements into system, subsystem, and class requirements. An application example about developing ATM machine object oriented software design process is used. The priorities resulting from the above has been used to prioritize the methods of design in the application program. The structure of classes corresponding to each phase of development has been shown.

## 2. QFD METHODOLOGY FOR OBJECT-ORIENTED SOFTWARE DEVELOPMENT; A NEW INTEGRATED FRAMEWORK

### 2.1 Phases of development

During the design and development phases, it is helpful to know what the most important design items are in terms of their correlation with the requirements. Thus, a priority assessment framework is provided to help find the important design items phase by phase. In this framework, HoQ incorporates customer requirements into multiple phases of the object-oriented software development life cycle, including system, subsystem, and class designs. There has been little research, however, on the traceability of customer requirements through object-oriented software developments. QFD seems to be a natural solution to this problem because it was developed to transform the voice of customer into designs. The advantage of using HoQ (from QFD) in this methodology is that it traces customer requirements from the very beginning to object class design. As a result, it is easier for both customers and developers to visualize which component is designed to reflect which set of requirements and to what extent these requirements are implemented. Based on the assessment result, limited resources can be allocated to more important design items and the resultant software product will achieve a higher level of customer satisfaction.

A new integrated framework (see Fig. 1) for the application of QFD to object-oriented software development is developed. There are three phases in this development life cycle that this framework covers. They are:

***Phase 1***: Customer requirements are deployed to both the product functions and the quality factors of the whole system. The fuzzy sets are used for correlation in this phase instead of crisp numbers.

***Phase 2***: The system characteristics, which reflect the voice of customers, obtained from the previous phase are deployed into the important subsystem functions and subsystem constraints.

***Phase 3***: The subsystem characteristics from the previous phase are deployed to the important class functions and class constraints.

Quality and functionalities are the two major issues affecting the degree of customer satisfaction. Thus, it is needed to relate customer requirements with each one of the two using HoQs. The HoQ relating customer requirements with the quality factors is given the name Q-HoQ; similarly, the HoQ relating customer requirements with the functionalities is given the name F-HoQ. The design point analysis matrix is then used to combine the quality factors and functionalities, both of which now have weight values reflecting the impacts from the customer requirements.

In Fig. 1, the matrices R2, S2 and C2 are Q-HoQs; the matrices R1, S1 and C1 constitute F-HoQs; and the matrices R3 and S3 are of the type of design point analysis matrix. The customer requirements serve as an input into R1 (F-HoQ) and R2 (Q-HoQ) requirement elicitation matrices. The results of these two requirements elicitation matrices serve as inputs for the R3 matrix. Results of the R3 matrix are used to combine the product functions and quality factors into one set of subsystem-level requirements, which are carried over to Phase 2 of the development life cycle where similar steps are taken

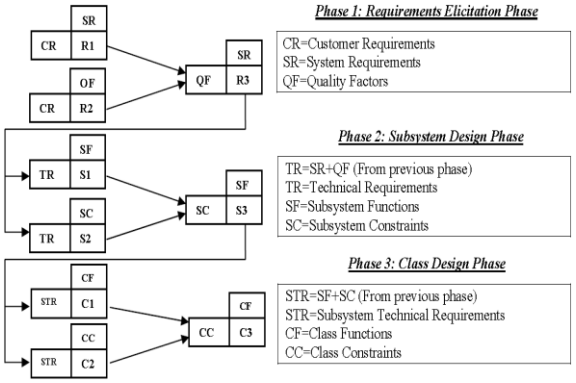



**Fig. 1** Integrated framework for object-oriented software development [8].

### 2.2 Types of Matrices of QFD

#### 2.2.1 The Q-HoQ Matrix (R2/S2/C2):

The structure of the Q-HoQ matrix is shown in Fig. 2. The most important components of the Q-HoQ are:

**Requirements**: They are identified from customer statements or are obtained from the previous phase.

**Importance:** Traditionally the importance column in the matrix accommodates a list of importance ratings (real values between 1 and 9) for the requirements entered. Importance ratings can be better achieved using the Analytical Hierarchy Process (AHP) Technique (Tables 1,2).

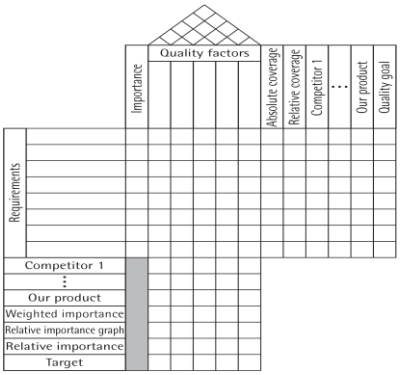



**Fig. 2** The Q-HoQ Matrix [8].

Table 1: A questionnaire form for deciding on the importance index of the customer requirements [3]

| With respect to " " | Importance (or Preference) of one Sub-Criterion over another | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Questions | Criteria | Absolute | Intermediate | Very Strong | Intermediate | Strong | Intermediate | Weak | Intermediate | Equal | Intermediate | Weak | Intermediate | Strong | Intermediate | Very Strong | Intermediate | Absolute | Criteria |
| | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |

Table 2: Analytical hierarchy process for prioritizing customer requirements [3]

| | CR-1 | CR-2 | CR-3 | CR-4 | CR-5 | CR-6 | CR-7 | CR-1 | CR-2 | CR-3 | CR-4 | CR-5 | CR-6 | CR-7 | Score | Product | Ratio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CR-1 | | | | | | | | | | | | | | | | | |
| CR-2 | | | | | | | | | | | | | | | | | |
| CR-3 | | | | | | | | | | | | | | | | | |
| CR-4 | | | | | | | | | | | | | | | | | |
| CR-5 | | | | | | | | | | | | | | | | | |
| CR-6 | | | | | | | | | | | | | | | | | |
| CR-7 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | CI | |
| | | | | | | | | | | | | | | | | CI/RI | |

**Quality factors:** The quality factors columns in the matrix accommodate a list of quality factors that contribute to the satisfaction of the requirements. Quality factors specify the desired quality attributes that need to be considered during the development of a particular software product, such as reliability, understandability, and so on.

**Correlation:** The degree of impact of a quality factor on the satisfaction of a requirement is entered in a correlation matrix cell (the intersection of the quality factor and the requirement). Seven levels of impact are used to fill these cells. The fuzzy set is used to implement this correlation (Fig. 3). Most researchers use special fuzzy numbers, such as triangular fuzzy numbers, trapezoidal fuzzy numbers, and R-L fuzzy numbers, to satisfy the need of modeling fuzzy problems. For simplicity, the most commonly used trapezoidal fuzzy numbers are used for necessary illustrations in this paper (Fig. 4). The proposed fuzzy QFD model provides the ability for changing the level of linguistic certainty for the problem by altering the proposed linguistic certainty index. That is, selecting different spreads of fuzzy numbers will reveal different levels of linguistic certainty (Fig. 5). A fuzzy number with a wider spread possesses a more ambiguous decision-making condition where the design team is uncertain with the evaluation. Conversely, a fuzzy number with a shorter spread represents a more clear and confident decision-making environment.

$$\mu_{\tilde{A}}(x)=\begin{cases} l(x) & for \quad x\in(\alpha,\beta) \\ 1 & for \quad x\in[\beta,\gamma] \\ r(x) & for \quad x\in(\gamma,\delta) \\ 0 & otherwise \end{cases}$$

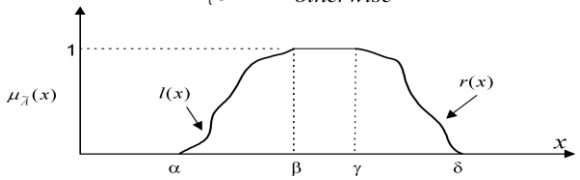



**Fig. 3** A typical graph of a fuzzy number described by the equation above, [7].

The membership function of a trapezoidal fuzzy number will be:

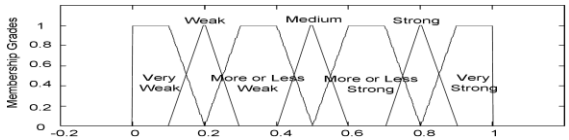



**Fig. 4** Linguistic terms for Correlation [7].

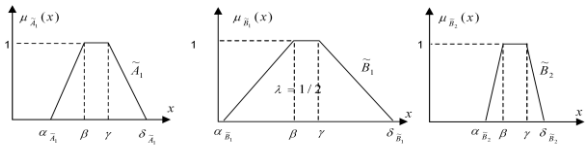



**Fig. 5** Different fuzzy numbers revealing different linguistic certainty levels [7].

**Absolute coverage**: The absolute coverage of a requirement is examined against its corresponding quality factors in the matrix. For each requirement X, across all quality factors, Y is calculated as:

$$ABS\_Coverage(X_i)=\sum_{k=1}^{\#\_Quality\_Factors} Correlation(X_i,Y_k) \qquad (1)$$

**Relative coverage:** The relative coverage of a requirement is examined against those of all requirements. For each requirement X, the relative coverage is calculated as:

$$REL\_Coverage(X_i) = \frac{ABS\_Coverage(X_i)}{\sum_{k=1}^{\#\_Requirements} ABS\_Coverage(X_k)} \times 10 \quad (2)$$

The relative coverage ensures that a high-priority customer requirement receives coverage proportional to its priority.

**Weighted and relative importance**: For each quality factor X, across all requirements Y, the weighted importance value can be calculated from the importance values of the requirements and the correlation values between this quality factor and all requirements as follows:

$$Weighted\_IMP(X_i) = \sum_{k=1}^{\#\_Requirements} Correlation(Y_k, X_i) \times IMP(Y_k)) \quad (3)$$

With all the weighted importance values calculated, the relative importance value of a quality factor X can be obtained as follows:

$$REL\_IMP(X_i) = \frac{Weighted\_IMP(X_i)}{\sum_{k=1}^{\#\_Quality\_Factors} Weighted\_IMP(X_k)} \times 10 \quad (4)$$

**Target**: The development targets are set for one's product.

**Roof**: The roof contains the tradeoffs between the quality elements. A plus sign (+) is used to indicate a positive relation and a minus sign (-) to indicate a negative relation. If improving the satisfaction of one quality factor will harm another, a negative relation exists between the two. For instance, if the fault tolerance requires more safety checking and recovering calculation, it will very likely sacrifice the efficiency of the system. Thus, fault tolerance and efficiency are negatively related. Conversely, if one quality factor improves another, there is a positive relation.

### 2.2.2 F-HoQ Matrix (R1/S1/C1):

The structure of the F-HoQ matrix is shown in Fig. 6.

It differs from the Q-HoQ by not having the roof, because the functions are implementation independent. Hence, negative correlations among them are rare. In addition, the F-HoQ deploys requirements to functions instead of quality factors. The calculations of the absolute and relative coverages for the requirements and the weighted and relative importance values for the functions are similar to Equations 1 to 4 used in the Q-HoQ.

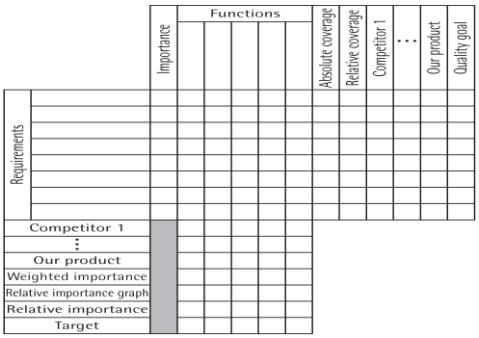



**Fig. 6** F-HoQ matrix [8].

It differs from the Q-HoQ by not having the roof, because the functions are implementation independent. Hence, negative correlations among them are rare. In addition, the F-HoQ deploys requirements to functions instead of quality factors. The calculations of the absolute and relative coverages for the requirements and the weighted and relative importance values for the functions are similar to Equations 1 to 4 used in the Q-HoQ.

### 2.2.3 Design Point Analysis Matrix (R3/S3):

The structure of the design point analysis matrix is shown in Fig. 7.

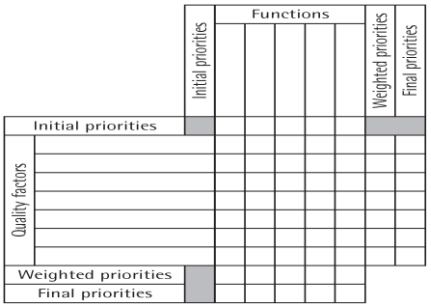



**Fig. 7** Design point analysis matrix [8].

It is used to integrate functions and quality factors by examining their impacts on each other. The aim is to produce technical requirements for the next phase so that the original customer requirements are traced along the design of the system components. Following is the list of components in the design point analysis matrix:

**Quality factors and functions**: These are obtained from the Q-HoQ and F-HoQ matrices.

**Initial priorities**: These are obtained from the relative importance values calculated in the Q-HoQ and F-HoQ matrices.

**Correlation**: The degree of importance of a quality factor on a function is entered in a correlation matrix cell (the intersection of the quality factor and the function) using crisp values (Fuzzy sets are used instead of crisp values in phase 1, requirements elicitation phase). Three levels of impact are used for crisp values to fill these cells (as shown in Fig. 8).

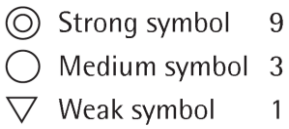



**Fig. 8**: Crisp correlation values [8].

**Weighted priorities**: For each quality factor X and each function Y, the weighted priority can be calculated from the initial priority values and the correlation values as follows:

$$Weighted\_P(X_i) = \sum_{k=1}^{\#\_functions} (Init\_P(X_i) \times Correlation(X_i Y_k) \times Init\_P(Y_k) \quad (5)$$

$$Weighted\_P(Y_i) = \sum_{k=1}^{\#\_Quality\_Factors} (Init\_P(Y_i) \times Correlation(Y_i X_k) \times Init\_P(X_k) \quad (6)$$

**Final priorities:** For each quality factor X and each function Y:

$$Final\_P(X_i) = \frac{Weighted\_P(X_i)}{\sum_{k=1}^{\#\_Quality\_Factors} Weighted\_P(X_k)} \times 10 \quad (7)$$

$$Final\_P(Y_i) = \frac{Weighted\_P(Y_i)}{\sum_{k=1}^{\#\_Functions} Weighted\_P(Y_k)} \times 10 \quad (8)$$

These final priorities are calculated for traceability purpose. They reflect the level of satisfaction of the original set of customer requirements.

## 3. APPLICATION EXAMPLE

The design of ATM machine software, through which bank customers can perform several of the most common financial transactions, was chosen as an example to illustrate the QFD methodology for object-oriented software development. The machine consists of a display screen, a bankcard reader, numeric and special input keys, a money dispenser slot, and a receipt printer. The methodology consists of three phases of object oriented software design. In Requirements Elicitation Phase, a number of requirements were elicited. For instance, the software should have an easy-to-use interface, real-time updating capability of the account information, and so on. From these requirements, the system design starts with a number of major system functionalities and system constraints. In the subsystem design phase, the integrated system-level functionalities and constraints become the subsystem requirements from which the subsystem constraints and functionalities are obtained. Finally, In Class Design Phase, the integrated subsystem-level constraints and functionalities are used to develop class-level functionalities and constraints. After the relative importance values of the class constraints and class functions are calculated, the original set of customer requirements are successfully transformed into the object class design in the object oriented software development process.

The major contribution of the new methodology is that it coordinates efficiently the following tools and functions:

- Customer Requirements are prioritized using the Analytical Hierarch Process (AHP).
- Fuzzy set theory is used for the correlation of requirements in the QFD, House of Quality, in requirements elicitation phase, for the purpose of increasing the discriminating ability of QFD analysis, removing the possible stakeholder's bias and increasing the accuracy of calculations.
- The House of Quality is automated on an excel sheet in order to save time and effort and ensure accuracy of calculating the absolute and relative weightings of technical requirements.
- TRIZ methodology is used in the Q_HoQ for resolving the technical contradictions between the feature parameters of the object oriented design process.
- Requirements traceability diagrams are demonstrated.
- Class diagrams and use cases are implemented to depict graphically object oriented software design process

### 3.1 Analytical Hierarchy Process

Categories of requirements are shown in Table 3.

**Table 3: Requirements categories of ATM machine [8]**

| No | Customer Requirements | | System Requirements | | Subsystem Requirements | | Class Requirements | |
|---|---|---|---|---|---|---|---|---|
| | ID | Title | ID | Title | ID | Title | ID | Title |
| 1 | CR1 | Ease of use | QF1 | Time to learn | SQF1 | Accuracy | CQF1 | Reusability |
| 2 | CR2 | Readable screen | QF2 | Data integrity | SQF2 | Efficiency | CQF2 | Security |
| 3 | CR3 | Easy to correct | QF3 | Level of security | SQF3 | Operability | CQF3 | Fault tolerance |
| 4 | CR4 | Access any account | SR1 | Access accounts | SF1 | Dispense cash | CQF4 | Recoverability |
| 5 | CR5 | Real-time update | SR2 | Updates account | SF2 | Screen display | CF1 | Control ATM |
| 6 | CR6 | Security | SR3 | Transfer funds | SF3 | Cash deposit | CF2 | Handle ATM card |
| 7 | CR7 | Fast response | SR4 | Display account status | SF4 | Read card | CF3 | Handle customer data |
| 8 | CR8 | Always available | SR5 | Validate access | SF5 | Access customer data | CF4 | Interface |
| 9 | CR9 | Accuracy | SR6 | Provide receipt | SF6 | Account handler | CF5 | Manage account |

The Analytical Hierarchy Process for prioritizing customer requirements (CR1, CR2… CR9) is shown in Tables 4, 5.

**Table 4: Questionnaire for CRs pair wise comparisons [3].**

| With respect to | | Importance (or Preference) of one Sub-Criterion over another | | | | | | | | | | | | | | | With respect to |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Questions | Criteria | Absolute | Intermediate | Very Strong | Intermediate | Strong | Intermediate | Weak | Intermediate | Equal | Intermediate | Weak | Intermediate | Strong | Intermediate | Very Strong | Intermediate | Absolute | Criteria |
| | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 1 | Ease of use | x | | | | | | | | | | | | | | | | | Readable screen |
| 2 | Ease of use | | | | | | | | x | | | | | | | | | | Easy to correct |
| 3 | Ease of use | | | | | | | | x | | | | | | | | | | Access any account |
| 4 | Ease of use | | | | | | | | x | | | | | | | | | | Real-time update |
| 5 | Ease of use | | | | | | | | x | | | | | | | | | | Security |
| 6 | Ease of use | | | | | | | x | | | | | | | | | | | Fast response |
| 7 | Ease of use | | | | | | | | | | x | | | | | | | | Always available |
| 8 | Ease of use | | | | | | | | | | x | | | | | | | | Accuracy |
| 9 | Readable screen | | | | | | | | | | | | | x | | | | | Easy to correct |
| 10 | Readable screen | | | | | | | | | | | | | | x | | | | Access any account |
| 11 | Readable screen | | | | | | | | | | | | | x | | | | | Real-time update |
| 12 | Readable screen | | | | | | | | | | | | x | | | | | | Security |
| 13 | Readable screen | | | | | | | | | | | x | | | | | | | Fast response |
| 14 | Readable screen | | | | | | | | | | | | | | | | | x | Always available |
| 15 | Readable screen | | | | | | | | | | | | | | | | | x | Accuracy |
| 16 | Easy to correct | | | | | | | | | x | | | | | | | | | Access any account |
| 17 | Easy to correct | | | | | | | | | x | | | | | | | | | Real-time update |
| 18 | Easy to correct | | | | | | | | | x | | | | | | | | | Security |
| 19 | Easy to correct | | | | | | | | x | | | | | | | | | | Fast response |
| 20 | Easy to correct | | | | | | | | | | | x | | | | | | | Always available |
| 21 | Easy to correct | | | | | | | | | | | x | | | | | | | Accuracy |
| 22 | Access any account | | | | | | | | | x | | | | | | | | | Real-time update |
| 23 | Access any account | | | | | | | | | x | | | | | | | | | Security |
| 24 | Access any account | | | | | | | | x | | | | | | | | | | Fast response |
| 25 | Access any account | | | | | | | | | | x | | | | | | | | Always available |
| 26 | Access any account | | | | | | | | | | | x | | | | | | | Accuracy |
| 27 | Real-time update | | | | | | | | | x | | | | | | | | | Security |
| 28 | Real-time update | | | | | | | | x | | | | | | | | | | Fast response |
| 29 | Real-time update | | | | | | | | | | | x | | | | | | | Always available |
| 30 | Real-time update | | | | | | | | | | | x | | | | | | | Accuracy |
| 31 | Security | | | | | | | | | x | | | | | | | | | Fast response |
| 32 | Security | | | | | | | | | | | x | | | | | | | Always available |
| 33 | Security | | | | | | | | | | | | x | | | | | | Accuracy |
| 34 | Fast response | | | | | | | | | | | | | x | | | | | Always available |
| 35 | Fast response | | | | | | | | | | | | | x | | | | | Accuracy |
| 36 | Always available | | | | | | | | | x | | | | | | | | | Accuracy |

**Table 5: AHP for prioritizing customer requirements of ATM machine software [3].**

**AHP-ATM**

| | | CR1 | CR2 | CR3 | CR4 | CR5 | CR6 | CR7 | CR8 | CR9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CR1 | 1 | 9 | 2 | 2 | 2 | 2 | 3 | 1/2 | 1/2 |
| 2 | CR2 | | 1 | 1/5 | 1/6 | 1/5 | 1/4 | 1/3 | 1/9 | 1/9 |
| 3 | CR3 | | | 1 | 1 | 1 | 1 | 2 | 1/3 | 1/3 |
| 4 | CR4 | | | | 1 | 1 | 1 | 2 | 1/2 | 1/3 |
| 5 | CR5 | | | | | 1 | 1 | 2 | 1/3 | 1/3 |
| 6 | CR6 | | | | | | 1 | 1 | 1/3 | 1/4 |
| 7 | CR7 | | | | | | | 1 | 1/5 | 1/5 |
| 8 | CR8 | | | | | | | | 1 | 1 |
| 9 | CR9 | | | | | | | | | 1 |

Consistency Check - Eigenvalue / Eigenvector Method

| | | CR1 | CR2 | CR3 | CR4 | CR5 | CR6 | CR7 | CR8 | CR9 | CR1 | CR2 | CR3 | CR4 | CR5 | CR6 | CR7 | CR8 | CR9 | Score | Product | Ratio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CR1 | 1.00 | 9.00 | 2.00 | 2.00 | 2.00 | 2.00 | 3.00 | 0.50 | 0.50 | 0.134328 | 0.176471 | 0.15748 | 0.171429 | 0.15748 | 0.140351 | 0.140625 | 0.115979 | 0.123119 | 0.15402343 | 1.3176 | 8.5543 |
| 2 | CR2 | 0.11 | 1.00 | 0.20 | 0.17 | 0.20 | 0.25 | 0.33 | 0.11 | 0.11 | 0.014925 | 0.019608 | 0.015748 | 0.014286 | 0.015748 | 0.017544 | 0.015625 | 0.025773 | 0.02736 | 0.01621198 | 0.1657 | 10.2181 |
| 3 | CR3 | 0.50 | 5.00 | 1.00 | 1.00 | 1.00 | 1.00 | 2.00 | 0.33 | 0.33 | 0.067164 | 0.098039 | 0.07874 | 0.085714 | 0.07874 | 0.070175 | 0.09375 | 0.07732 | 0.082079 | 0.08176049 | 0.7297 | 8.9250 |
| 4 | CR4 | 0.50 | 6.00 | 1.00 | 1.00 | 1.00 | 1.00 | 2.00 | 0.50 | 0.33 | 0.067164 | 0.117647 | 0.07874 | 0.085714 | 0.07874 | 0.070175 | 0.09375 | 0.115979 | 0.082079 | 0.08456161 | 0.7824 | 9.2529 |
| 5 | CR5 | 0.50 | 5.00 | 1.00 | 1.00 | 1.00 | 1.00 | 2.00 | 0.33 | 0.33 | 0.067164 | 0.098039 | 0.07874 | 0.085714 | 0.07874 | 0.070175 | 0.09375 | 0.07732 | 0.082079 | 0.08176049 | 0.7297 | 8.9250 |
| 6 | CR6 | 0.50 | 4.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.33 | 0.25 | 0.067164 | 0.078431 | 0.07874 | 0.085714 | 0.07874 | 0.070175 | 0.046875 | 0.07732 | 0.06156 | 0.07226294 | 0.6445 | 8.9187 |
| 7 | CR7 | 0.33 | 3.00 | 0.50 | 0.50 | 0.50 | 1.00 | 1.00 | 0.20 | 0.20 | 0.044776 | 0.058824 | 0.03937 | 0.042857 | 0.03937 | 0.070175 | 0.046875 | 0.046392 | 0.049248 | 0.04889248 | 0.4373 | 8.9437 |
| 8 | CR8 | 2.00 | 9.00 | 3.00 | 2.00 | 3.00 | 3.00 | 5.00 | 1.00 | 1.00 | 0.268657 | 0.176471 | 0.23622 | 0.171429 | 0.23622 | 0.210526 | 0.234375 | 0.231959 | 0.246238 | 0.21912831 | 2.0354 | 9.2887 |
| 9 | CR9 | 2.00 | 9.00 | 3.00 | 3.00 | 3.00 | 4.00 | 5.00 | 1.00 | 1.00 | 0.268657 | 0.176471 | 0.23622 | 0.257143 | 0.23622 | 0.280702 | 0.234375 | 0.231959 | 0.246238 | 0.24139827 | 2.1922 | 9.0814 |
| | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | CI | 0.015387 |
| | | | | | | | | | | | | | | | | | | | | | CI/RI | 0.010612 |

It is obvious from the consistency check of the AHP process that the inconsistency ratio (CI/RI) is about 1%. This is a very small ratio (acceptable ratio is up to 10%) and indicates that the inconsistency in judgment of the pair-wise comparison of customer requirements is negligible.

## 3.2 Design Phases

### 2.2.4 3.2.1 Phase 1: Customer requirements – system requirements

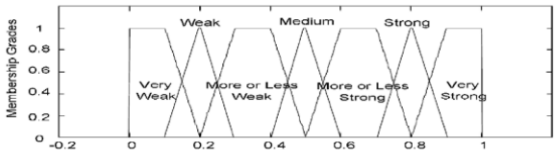
The Quality, House of Quality (Q_HoQ), in requirements elicitation phase is shown in **Fig. 9.**

The Functional, House of Quality (F_HoQ), in requirements elicitation phase is shown in **Fig. 10**

The Design Point Analysis Matrix in Requirements Elicitation Phase is shown in **Fig. 11.**

| | VW | W | MW | M | MS | S | VS |
|---|---|---|---|---|---|---|---|
| α | 0 | 0.1 | 0.2 | 0.4 | 0.5 | 0.7 | 0.8 |
| β | 0 | 0.2 | 0.3 | 0.5 | 0.6 | 0.8 | 0.9 |
| γ | 0.1 | 0.2 | 0.4 | 0.5 | 0.7 | 0.8 | 1 |
| δ | 0.2 | 0.3 | 0.5 | 0.6 | 0.8 | 0.9 | 1 |

| Correlation | |
|---|---|
| Very Weak | VW |
| Weak | W |
| More-Or-Less Weak | MW |
| Mediam | M |
| More-Or-Less Strong | MS |
| Strong | S |
| Very Strong | VS |




| Q-HoQ - In Requirements Elicitation Phase | | Reqs Importance | Time to Learn | Data Integrity | Level of Security | Absolute Coverage |
|---|---|---|---|---|---|---|
| | | | QF1 | QF2 | QF3 | |
| Ease of use | | | S | S | | |
| | α | 0.154023 | 0.7 | 0.7 | | |
| | β | 0.154023 | 0.8 | 0.8 | | |
| | γ | 0.154023 | 0.8 | 0.8 | | 1.6 |
| | δ | 0.154023 | 0.9 | 0.9 | | 1.8 |
| Readable screen | | | S | | | |
| | α | 0.016212 | 0.7 | | | |
| | β | 0.016212 | 0.8 | | | |
| | γ | 0.016212 | 0.8 | | | 0.8 |
| | δ | 0.016212 | 0.9 | | | 0.9 |
| Easy to correct | | | M | | | |
| | α | 0.08176 | 0.4 | | | |
| | β | 0.08176 | 0.5 | | | |
| | γ | 0.08176 | 0.5 | | | 0.5 |
| | δ | 0.08176 | 0.6 | | | 0.6 |
| Access any account | | | W | M | S | |
| | α | 0.084562 | 0.1 | 0.4 | 0.7 | |
| | β | 0.084562 | 0.2 | 0.5 | 0.8 | |
| | γ | 0.084562 | 0.2 | 0.5 | 0.8 | 1.5 |
| | δ | 0.084562 | 0.3 | 0.6 | 0.9 | 1.8 |
| Real-time update | | | | M | | |
| | α | 0.08176 | | 0.4 | | |
| | β | 0.08176 | | 0.5 | | |
| | γ | 0.08176 | | 0.5 | | 0.5 |
| | δ | 0.08176 | | 0.6 | | 0.6 |
| Security | | | | M | S | |
| | α | 0.072263 | | 0.4 | 0.7 | |
| | β | 0.072263 | | 0.5 | 0.8 | |
| | γ | 0.072263 | | 0.5 | 0.8 | 1.3 |
| | δ | 0.072263 | | 0.6 | 0.9 | 1.5 |
| Fast response | | | M | | | |
| | α | 0.048892 | 0.4 | | | |
| | β | 0.048892 | 0.5 | | | |
| | γ | 0.048892 | 0.5 | | | 0.5 |
| | δ | 0.048892 | 0.6 | | | 0.6 |
| Always available | | | M | | | |
| | α | 0.219128 | 0.4 | | | |
| | β | 0.219128 | 0.5 | | | |
| | γ | 0.219128 | 0.5 | | | 0.5 |
| | δ | 0.219128 | 0.6 | | | 0.6 |
| Accuracy | | | | S | M | |
| | α | 0.241398 | | 0.7 | 0.4 | |
| | β | 0.241398 | | 0.8 | 0.5 | |
| | γ | 0.241398 | | 0.8 | 0.5 | 1.3 |
| | δ | 0.241398 | | 0.9 | 0.6 | 1.5 |
| Competitor 1 | | | | | | |
| Competitor 2 | | | | | | |
| Our Product | | | W1 | W2 | W3 | |
| Weighted Importance | γ | | 0.328 | 0.436 | 0.246 | |
| Weighted Importance | δ | | 0.388 | 0.499 | 0.286 | |
| Integral Priorities | (γ + δ)/2 | | 0.3582 | 0.4673 | 0.2661 | |
| Relative Importance | | | 2 | 1 | 3 | |
| Target | | | | | | |

**Fig. 9 Q-HoQ in requirements elicitation phase**

| F-HoQ - In Requirements Elicitation Phase | | Requirements Importance | Access Accounts | Updates Account | Transfer Funds | Display Account status | Validate Access | Provide Receipt | Absolute Coverage |
|---|---|---|---|---|---|---|---|---|---|
| | | | SR1 | SR2 | SR3 | SR4 | SR5 | SR6 | |
| Ease of use | | | M | | M | M | | S | |
| | α | 0.154023 | 0.4 | | 0.4 | 0.4 | | 0.7 | |
| | β | 0.154023 | 0.5 | | 0.5 | 0.5 | | 0.8 | |
| | γ | 0.154023 | 0.5 | | 0.5 | 0.5 | | 0.8 | 2.3 |
| | δ | 0.154023 | 0.6 | | 0.6 | 0.6 | | 0.9 | 2.7 |
| Readable screen | | | M | | | | | | |
| | α | 0.016212 | 0.4 | | | | | | |
| | β | 0.016212 | 0.5 | | | | | | |
| | γ | 0.016212 | 0.5 | | | | | | 0.5 |
| | δ | 0.016212 | 0.6 | | | | | | 0.6 |
| Easy to correct | | | M | | M | M | | | |
| | α | 0.08176 | 0.4 | | 0.4 | 0.4 | | | |
| | β | 0.08176 | 0.5 | | 0.5 | 0.5 | | | |
| | γ | 0.08176 | 0.5 | | 0.5 | 0.5 | | | 1.5 |
| | δ | 0.08176 | 0.6 | | 0.6 | 0.6 | | | 1.8 |
| Access any account | | | S | M | W | W | M | W | |
| | α | 0.084562 | 0.7 | 0.4 | 0.1 | 0.1 | 0.4 | 0.1 | |
| | β | 0.084562 | 0.8 | 0.5 | 0.2 | 0.2 | 0.5 | 0.2 | |
| | γ | 0.084562 | 0.8 | 0.5 | 0.2 | 0.2 | 0.5 | 0.2 | 2.4 |
| | δ | 0.084562 | 0.9 | 0.6 | 0.3 | 0.3 | 0.6 | 0.3 | 3 |
| Real-time update | | | W | S | M | M | | | |
| | α | 0.08176 | 0.1 | 0.7 | 0.4 | 0.4 | | | |
| | β | 0.08176 | 0.2 | 0.8 | 0.5 | 0.5 | | | |
| | γ | 0.08176 | 0.2 | 0.8 | 0.5 | 0.5 | | | 2 |
| | δ | 0.08176 | 0.3 | 0.9 | 0.6 | 0.6 | | | 2.4 |
| Security | | | | M | M | M | S | | |
| | α | 0.072263 | | 0.4 | 0.4 | 0.4 | 0.7 | | |
| | β | 0.072263 | | 0.5 | 0.5 | 0.5 | 0.8 | | |
| | γ | 0.072263 | | 0.5 | 0.5 | 0.5 | 0.8 | | 2.3 |
| | δ | 0.072263 | | 0.6 | 0.6 | 0.6 | 0.9 | | 2.7 |
| Fast response | | | S | W | M | M | | M | |
| | α | 0.048892 | 0.7 | 0.1 | 0.4 | 0.4 | | 0.4 | |
| | β | 0.048892 | 0.8 | 0.2 | 0.5 | 0.5 | | 0.5 | |
| | γ | 0.048892 | 0.8 | 0.2 | 0.5 | 0.5 | | 0.5 | 2.5 |
| | δ | 0.048892 | 0.9 | 0.3 | 0.6 | 0.6 | | 0.6 | 3 |
| Always available | | | | M | | | | | |
| | α | 0.219128 | | 0.4 | | | | | |
| | β | 0.219128 | | 0.5 | | | | | |
| | γ | 0.219128 | | 0.5 | | | | | 0.5 |
| | δ | 0.219128 | | 0.6 | | | | | 0.6 |
| Accuracy | | | | S | W | W | W | W | |
| | α | 0.241398 | | 0.7 | 0.1 | 0.1 | 0.1 | 0.1 | |
| | β | 0.241398 | | 0.8 | 0.2 | 0.2 | 0.2 | 0.2 | |
| | γ | 0.241398 | | 0.8 | 0.2 | 0.2 | 0.2 | 0.2 | 1.6 |
| | δ | 0.241398 | | 0.9 | 0.3 | 0.3 | 0.3 | 0.3 | 2.1 |
| Competitor 1 | | | | | | | | | |
| Competitor 2 | | | | | | | | | |
| Our Product | | | W1 | W2 | W3 | W4 | W5 | W6 | |
| Weighted Importance | γ | | 0.249 | 0.456 | 0.285 | 0.285 | 0.148 | 0.213 | |
| Weighted Importance | δ | | 0.296 | 0.531 | 0.361 | 0.361 | 0.188 | 0.266 | |
| Integral Priorities | (γ + δ)/2 | | **0.2725** | **0.4937** | **0.3228** | **0.3228** | **0.1683** | **0.2393** | |
| Relative Importance | | | **3** | **1** | **2** | | **5** | **4** | |
| Target | | | | | | | | | |

**Fig. 10: F-HoQ in requirements elicitation phase**

| Design Point Analysis Matrix - In Requirements Elicitation Phase | | | Initial Priorities | Access Accounts | Updates Account | Transfer Funds | Display Account status | Validate Access | Provide Receipt | Weighted Priorities | Final Priorities |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | SR1 | SR2 | SR3 | SR4 | SR5 | SR6 | | |
| Initial Priorities | | | | 0.2725 | 0.4937 | 0.3228 | 0.3228 | 0.1683 | 0.2393 | | |
| Time to learn | | QF1 | | | M | M | M | | | | |
| | α | | 0.3582 | | 0.4 | 0.4 | 0.4 | | | | |
| | β | | 0.3582 | | 0.5 | 0.5 | 0.5 | | | | |
| | γ | | 0.3582 | | 0.5 | 0.5 | 0.5 | | | 0.2040 | |
| | δ | | 0.3582 | | 0.6 | 0.6 | 0.6 | | | 0.2449 | |
| | (γ + δ)/2 | | | | | | | | | **0.2245** | **3** |
| Data integrity | | QF2 | | M | M | M | M | M | M | | |
| | α | | 0.4673 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | | |
| | β | | 0.4673 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | | |
| | γ | | 0.4673 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.4251 | |
| | δ | | 0.4673 | 0.6 | 0.6 | 0.6 | 0.6 | 0.6 | 0.6 | 0.5101 | |
| | (γ + δ)/2 | | | | | | | | | **0.4676** | **1** |
| Level of security | | QF3 | | S | M | S | M | S | | | |
| | α | | 0.2661 | 0.7 | 0.4 | 0.7 | 0.4 | 0.7 | | | |
| | β | | 0.2661 | 0.8 | 0.5 | 0.8 | 0.5 | 0.8 | | | |
| | γ | | 0.2661 | 0.8 | 0.5 | 0.8 | 0.5 | 0.8 | | 0.2711 | |
| | δ | | 0.2661 | 0.9 | 0.6 | 0.9 | 0.6 | 0.9 | | 0.3132 | |
| | (γ + δ)/2 | | | | | | | | | **0.2922** | **2** |
| Weighted Importance | γ | | | 0.122 | 0.269 | 0.202 | 0.176 | 0.075 | 0.056 | | |
| Weighted Importance | δ | | | 0.142 | 0.323 | 0.237 | 0.211 | 0.087 | 0.067 | | |
| Integral Priorities | (γ + δ)/2 | | | 0.1317 | 0.2964 | 0.2196 | 0.1938 | 0.0813 | 0.0615 | | |
| Final Priorities | | | | 4 | 1 | 2 | 3 | 5 | 6 | | |

**Fig. 11: Design point analysis matrix in requirements elicitation phase**

The importance index of System Requirements and Quality Factors, as an output from the Requirements Elicitation Phase, is shown in Table 6.

**Table 6: Priorities of System Requirements and Quality Factors**

| No | Requirement Category | ID | Requirement Title | Priority |
|---|---|---|---|---|
| 1 | System Requirement | SR2 | Updates Account | 0.2964 |
| 2 | System Requirement | SR3 | Transfer Funds | 0.2196 |
| 3 | System Requirement | SR4 | Display Account Status | 0.1938 |
| 4 | System Requirement | SR1 | Access Accounts | 0.1317 |
| 5 | System Requirement | SR5 | Validate Access | 0.0813 |
| 6 | System Requirement | SR6 | Provide Receipt | 0.0615 |
| 1 | Quality Factor | QF2 | Data Integrity | 0.4676 |
| 2 | Quality Factor | QF3 | Level of Security | 0.2922 |
| 3 | Quality Factor | QF1 | Time to Learn | 0.2245 |

### 2.2.5 Phase 2: System Requirements – Subsystem Requirements

The Quality, House of Quality (Q_HoQ), in subsystem design phase is shown in Fig. 12

| Q-HoQ - In Subsystem Design Phase | Requirements Importance | Accuracy | Efficiency | Operability | Absolute Coverage | Relative Coverage | Competitor 1 | Competitor 2 | Our Product | Quality Goal |
|---|---|---|---|---|---|---|---|---|---|---|
| Access Accounts | 0.1317 | 1 | | | 1 | 0.26 | | | | |
| Updates Account | 0.2964 | 3 | | 3 | 6 | 1.58 | | | | |
| Transfer Funds | 0.2196 | 1 | | | 1 | 0.26 | | | | |
| Display Account Status | 0.1938 | 3 | 3 | 1 | 7 | 1.84 | | | | |
| Validate Access | 0.0813 | 3 | | 3 | 6 | 1.58 | | | | |
| Provide Receipt | 0.0615 | 1 | 3 | | 4 | 1.05 | | | | |
| Time to Learn | 0.2245 | | 3 | | 3 | 0.79 | | | | |
| Data Integrity | 0.4676 | 3 | | 3 | 6 | 1.58 | | | | |
| Levels of Security | 0.2922 | 3 | | 1 | 4 | 1.05 | | | | |
| Competitor 1 | | | | | 38 | | | | | |
| Competitor 2 | | | | | | | | | | |
| Our Product | | | | | | | | | | |
| Weighted Importance | | 4.4067 | 1.4394 | 3.0219 | 8.868 | | | | | |
| Relative Importance | | 4.969 | 1.623 | 3.408 | | | | | | |
| Target | | | | | | | | | | |

| | |
|---|---|
| Strong | 9 |
| Medium | 3 |
| Weak | 1 |

**Fig. 12: Q-HoQ in subsystem design phase**

The Functional, House of Quality (F_HoQ), in Subsystem Design phase is shown in Fig. 13.

| F-HoQ - In Subsystem Design Phase | Requirements Importance | Dispense Cash | Screen Display | Cash Deposit | Read Card | Access Customer Data | Account Handler | Absolute Coverage | Relative Coverage | Competitor 1 | Competitor 2 | Our Product | Quality Goal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Access Accounts | 0.1317 | 1 | | | 3 | 9 | 3 | 16 | 1.39 | | | | |
| Updates Account | 0.2964 | 3 | | 3 | 3 | 3 | 9 | 21 | 1.83 | | | | |
| Transfer Funds | 0.2196 | 1 | | | 3 | 3 | 9 | 16 | 1.39 | | | | |
| Display Account Status | 0.1938 | 3 | 3 | 1 | 1 | 1 | 3 | 12 | 1.04 | | | | |
| Validate Access | 0.0813 | 3 | | 3 | 3 | 1 | 3 | 13 | 1.13 | | | | |
| Provide Receipt | 0.0615 | 1 | 3 | | | | | 4 | 0.35 | | | | |
| Time to Learn | 0.2245 | | 3 | | 1 | | | 4 | 0.35 | | | | |
| Data Integrity | 0.4676 | 3 | | 3 | | 9 | 3 | 18 | 1.57 | | | | |
| Levels of Security | 0.2922 | 3 | | 1 | 1 | 3 | 3 | 11 | 0.96 | | | | |
| Competitor 1 | | | | | | | | 115 | | | | | |
| Competitor 2 | | | | | | | | | | | | | |
| Our Product | | | | | | | | | | | | | |
| Weighted Importance | | 4.407 | 1.439 | 3.022 | 2.898 | 8.093 | 8.144 | 28.0 | | | | | |
| Relative Importance | | 1.574 | 0.514 | 1.079 | 1.035 | 2.890 | 2.908 | | | | | | |
| Target | | | | | | | | | | | | | |

| | |
|---|---|
| Strong | 9 |
| Medium | 3 |
| Weak | 1 |

**Fig. 13: F-HoQ in subsystem design phase**

The Design Point Analysis Matrix in Subsystem Design Phase is shown in Fig. 14.

| Design Point Analysis Matrix - In Subsystem Design Phase | Initial Priorities | Dispense Cash | Screen Display | Cash Deposit | Read Card | Access Customer Data | Account Handler | Weighted Priorities | Final Priorities |
|---|---|---|---|---|---|---|---|---|---|
| Initial Priorities | | 1.574 | 0.514 | 1.079 | 1.035 | 2.890 | 2.908 | | |
| Accuracy | 4.969 | 1 | | 1 | 9 | | 1 | 73.9 | 5.010 |
| Efficiency | 1.623 | 1 | | 3 | 1 | 3 | 3 | 37.7 | 2.556 |
| Operability | 3.408 | 3 | 3 | 3 | 1 | | | 35.9 | 2.434 |
| | | | | | | | | 147.5 | |
| Weighted Priorities | | 26.468 | 5.255 | 21.647 | 51.493 | 14.071 | 28.609 | 147.5 | |
| Final Priorities | | 1.794 | 0.356 | 1.467 | 3.490 | 0.954 | 1.939 | | |

| | |
|---|---|
| Strong | 9 |
| Medium | 3 |
| Weak | 1 |

**Fig. 14: Design point analysis matrix in subsystem design phase**

The importance index of Subsystem Functions and Subsystem Constraints, as an output from the Subsystem Design Phase, is shown in Table 7.

**Table 7: Priorities of Subsystem Functions and Subsystem Constraints**

| No | Requirement Category | ID | Requirement Title | Priority |
|---|---|---|---|---|
| 1 | Subsystem Function | SF4 | Read Card | 3.490 |
| 2 | Subsystem Function | SF6 | Account Handler | 1.939 |
| 3 | Subsystem Function | SF1 | Dispense Cash | 1.794 |
| 4 | Subsystem Function | SF3 | Cash Deposit | 1.467 |
| 5 | Subsystem Function | SF5 | Access Customer Data | 0.954 |
| 6 | Subsystem Function | SF2 | Screen Display | 0.356 |
| 1 | Subsystem Constraint | SC1 | Accuracy | 5.010 |
| 2 | Subsystem Constraint | SC2 | Efficiency | 2.556 |
| 3 | Subsystem Constraint | SC3 | Operability | 2.434 |

### 2.2.6 Phase 3: Subsystem Requirements – Class Requirements

The Quality, House of Quality (Q_HoQ), in Class Design phase is shown in Fig. 15.

| Q-HoQ - In Class Design Phase | Requirements Importance | Reusability | Security | Fault Tolerance | Recoverability | Absolute Coverage | Relative Coverage | Competitor 1 | Competitor 2 | Our Product | Quality Goal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Dispense Cash | 1.794 | 1 | 1 | | | 2 | 0.31 | | | | |
| Screen Display | 0.356 | 1 | | | | 1 | 0.16 | | | | |
| Cash Deposit | 1.467 | 1 | 1 | 1 | 9 | 12 | 1.88 | | | | |
| Read Card | 3.490 | 1 | 3 | 3 | 3 | 10 | 1.56 | | | | |
| Access Customer Data | 0.954 | | 9 | 9 | 3 | 21 | 3.28 | | | | |
| Account Handler | 1.939 | | | 9 | 3 | 12 | 1.88 | | | | |
| Accuracy | 5.010 | | | 3 | | 3 | 0.47 | | | | |
| Efficiency | 2.556 | 1 | | | 1 | 2 | 0.31 | | | | |
| Operability | 2.434 | 1 | | | | 1 | 0.16 | | | | |
| Competitor 1 | | | | | | 64 | | | | | |
| Competitor 2 | | | | | | | | | | | |
| Our Product | | | | | | | | | | | |
| Weighted Importance | | 12.10 | 22.32 | 53.00 | 34.91 | 122.33 | | | | | |
| Relative Importance | | 0.989 | 1.824 | 4.333 | 2.854 | | | | | | |
| Target | | | | | | | | | | | |

| | |
|---|---|
| Strong | 9 |
| Medium | 3 |
| Weak | 1 |

**Fig. 15: Q-HoQ in class design phase**

The Functional, House of Quality (F_HoQ), in Class Design phase is shown in Fig. 16.

| F-HoQ - In Class Design Phase | Requirements Importance | Control ATM | Handle ATM Card | Handle Customer Data | Interface | Manage Account | Absolute Coverage | Relative Coverage | Competitor 1 | Competitor 2 | Our Product | Quality Goal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dispense Cash | 1.794 | 3 | 1 | | 3 | 3 | 10 | 1.18 | | | | |
| Screen Display | 0.356 | | | | 3 | 1 | 4 | 0.47 | | | | |
| Cash Deposit | 1.467 | 1 | 1 | 1 | 3 | 3 | 9 | 1.06 | | | | |
| Read Card | 3.490 | 1 | 3 | | 1 | 3 | 8 | 0.94 | | | | |
| Access Customer Data | 0.954 | 1 | 3 | 9 | | 3 | 16 | 1.88 | | | | |
| Account Handler | 1.939 | 3 | 1 | 1 | | 3 | 8 | 0.94 | | | | |
| Accuracy | 5.010 | 1 | | 9 | | 3 | 13 | 1.53 | | | | |
| Efficiency | 2.556 | 3 | | | | 9 | 12 | 1.41 | | | | |
| Operability | 2.434 | 3 | 1 | 1 | | | 5 | 0.59 | | | | |
| Competitor 1 | | | | | | | 85 | | | | | |
| Competitor 2 | | | | | | | | | | | | |
| Our Product | | | | | | | | | | | | |
| Weighted Importance | | 37.09 | 20.97 | 59.52 | 14.34 | 67.32 | 199.2 | | | | | |
| Relative Importance | | 1.862 | 1.052 | 2.987 | 0.720 | 3.379 | | | | | | |
| Target | | | | | | | | | | | | |

| | | |
|---|---|---|
| ◎ | Strong | 9 |
| ○ | Medium | 3 |
| ▽ | Weak | 1 |

Fig. 16: F-HoQ in Class Design Phase

The importance index of Class Functions and Class Constraints, as an output from the Class Design Phase, is shown in Table 8.

Table 8: Priorities of Class Functions and Class Constraints

| No | Requirement Category | ID | Requirement Title | Priority |
|---|---|---|---|---|
| 1 | Class Function | CF5 | Manage Account | 3.379 |
| 2 | Class Function | CF3 | Handle Customer Data | 2.987 |
| 3 | Class Function | CF1 | Control ATM | 1.862 |
| 4 | Class Function | CF2 | Handle ATM Card | 1.052 |
| 5 | Class Function | CF4 | Interface | 0.720 |
| 1 | Class Constraint | CC3 | Fault Tolerance | 4.333 |
| 2 | Class Constraint | CC4 | Recoverability | 2.854 |
| 3 | Class Constraint | CC2 | Security | 1.824 |
| 4 | Class Constraint | CC1 | Reusability | 0.989 |

The resultant priorities of requirements / Functions / Constraints across the three phases of object oriented software design process are shown in Table 9.

Table 9: Priorities of Requirements / Functions / Constraints across the phases of design

| Phase 1 | | | Phase 2 | | | Phase 3 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Customer Requirements | | | System Functions/Constraints | | | Subsystem Functions/Constraints | | | Class Functions/Constraints | | |
| ID | Title | Priority | ID | Title | Priority | ID | Title | Priority | ID | Title | Priority |
| CR9 | Accuracy | 0.2414 | SR2 | Updates Account | 0.2964 | SF4 | Read Card | 3.490 | CF5 | Manage Account | 3.379 |
| CR8 | Always available | 0.2191 | SR3 | Transfer Funds | 0.2196 | SF6 | Account Handler | 1.939 | CF3 | Handle Customer Data | 2.987 |
| CR1 | Ease of use | 0.1540 | SR4 | Display Account Status | 0.1938 | SF1 | Dispense Cash | 1.794 | CF1 | Control ATM | 1.862 |
| CR4 | Access any account | 0.0846 | SR1 | Access Accounts | 0.1317 | SF3 | Cash Deposit | 1.467 | CF2 | Handle ATM Card | 1.052 |
| CR3 | Easy to correct | 0.0818 | SR5 | Validate Access | 0.0813 | SF5 | Access Customer Data | 0.954 | CF4 | Interface | 0.720 |
| CR5 | Real-time update | 0.0818 | SR6 | Provide Receipt | 0.0615 | SF2 | Screen Display | 0.356 | CC3 | Fault Tolerance | 4.333 |
| CR6 | Security | 0.0723 | QF2 | Data Integrity | 0.4676 | SC1 | Accuracy | 5.010 | CC4 | Recoverability | 2.854 |
| CR7 | Fast response | 0.0489 | QF3 | Level of Security | 0.2922 | SC2 | Efficiency | 2.556 | CC2 | Security | 1.824 |
| CR2 | Readable screen | 0.0162 | QF1 | Time to Learn | 0.2245 | SC3 | Operability | 2.434 | CC1 | Reusability | 0.989 |

## 3.3 Requirements Traceability

Requirements traceability through all three phases of ATM machine object oriented software design is shown in the traceability diagram, Fig. 17.

Requirements traceability can be demonstrated in two ways: Forward Requirements Traceability and Backward Requirements Traceability.

Table 10 shows the Forward Requirements Traceability (in phase 1: Requirements Elicitation) for ATM machine object oriented software design process.

Table 11 shows the Backward Requirements Traceability (in phase 3: Class Design) for ATM machine object oriented software design process.

## 2.3 Solving Software Contradictions using TRIZ Methodology

The feature parameters that may have contradictions are shown in table 12 [11]:

Table 13 shows part of the dependence (Contradiction) matrix of OOD.

The numbers in the above table indicate which design plans that can solve the problem as shown in Table 14

We can see from the Q-HoQ in the subsystem design phase that the two contradicting requirements (Quality

constraints) are: SC1 (Accuracy) and SC3 (Operability) and these two requirements have priorities one and three respectively. So it is needed to solve the contradiction between them using the TRIZ methodology for OOD.

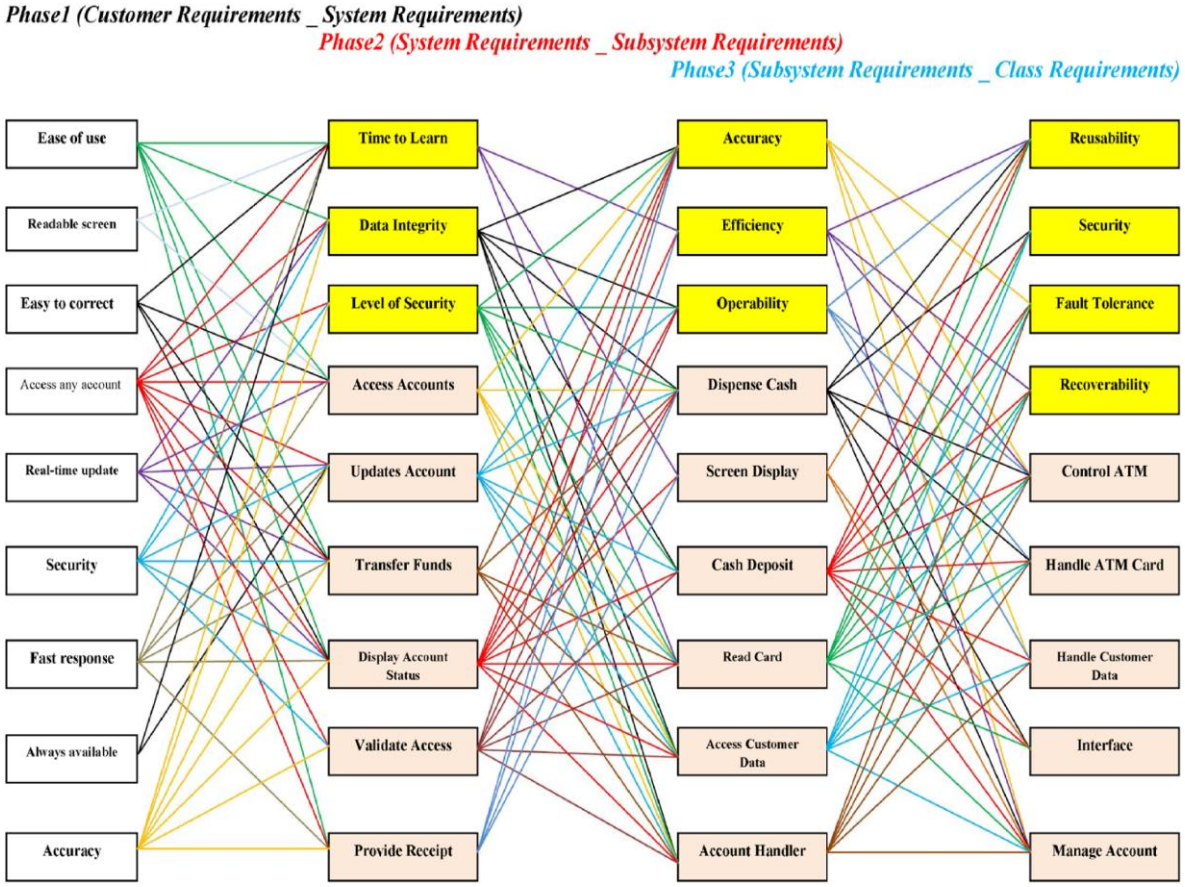



**Fig. 17: Requirements traceability diagram for atm machine object oriented software design process.**

**Table 10: Part of Forward Requirements Traceability Matrix**

| Phase 1: Customer Requirements _ System Requirements | | | | | |
|---|---|---|---|---|---|
| **Weight** | **ID** | **Customer Requirements** | **Correlation** | **ID** | **Forward Traceability-I** |
| 0.154023 | CR1 | Ease of Use | S | QF1 | Time to Learn |
| | | | S | QF2 | Data Integrity |
| | | | M | SR1 | Access Accounts |
| | | | M | SR3 | Transfer Funds |
| | | | M | SR4 | Display Account status |
| | | | S | SR6 | Provide Receipt |
| 0.016212 | CR2 | Readable Screen | S | QF1 | Time to Learn |
| | | | M | SR1 | Access Accounts |
| 0.08176 | CR3 | Easy to Correct | M | QF1 | Time to Learn |
| | | | M | SR1 | Access Accounts |
| | | | M | SR3 | Transfer Funds |
| | | | M | SR4 | Display Account status |
| 0.08456 | CR4 | Access any Account | W | QF1 | Time to Learn |
| | | | M | QF2 | Data Integrity |
| | | | S | QF3 | Level of Security |
| | | | S | SR1 | Access Accounts |
| | | | M | SR2 | Updates Account |
| | | | W | SR3 | Transfer Funds |
| | | | W | SR4 | Display Account status |
| | | | M | SR5 | Validate Access |
| | | | W | SR6 | Provide Receipt |
| 0.08176 | CR5 | Real Time Update | M | QF2 | Data Integrity |
| | | | W | SR1 | Access Accounts |
| | | | S | SR2 | Updates Account |
| | | | M | SR3 | Transfer Funds |
| | | | M | SR4 | Display Account status |
| 0.07226 | CR6 | Security | M | QF2 | Data Integrity |
| | | | S | QF3 | Level of Security |
| | | | M | SR2 | Updates Account |
| | | | M | SR3 | Transfer Funds |
| | | | M | SR4 | Display Account status |
| | | | S | SR5 | Validate Access |
| 0.04889 | CR7 | Fast Response | M | QF1 | Time to Learn |
| | | | S | SR1 | Access Accounts |
| | | | W | SR2 | Updates Account |
| | | | M | SR3 | Transfer Funds |
| | | | M | SR4 | Display Account status |
| | | | M | SR6 | Provide Receipt |
| 0.21913 | CR8 | Always Available | M | QF1 | Time to Learn |
| | | | M | SR2 | Updates Account |
| 0.2414 | CR9 | Accuracy | S | QF2 | Data Integrity |
| | | | M | QF3 | Level of Security |
| | | | S | SR2 | Updates Account |
| | | | W | SR3 | Transfer Funds |
| | | | W | SR4 | Display Account status |
| | | | W | SR5 | Validate Access |
| | | | W | SR6 | Provide Receipt |

**Table 11: Part of Backward Requirements Traceability Matrix**

| Phase 3: Class Requirements _ Subsystem Requirements | | | | | |
|---|---|---|---|---|---|
| | ID | Class Requirements | Correlation | ID | Backward Traceability-I |
| | CC1 | Reusability | W | SC2 | Efficiency |
| | | | W | SC3 | Operability |
| | | | W | SF1 | Dispense Cash |
| | | | W | SF2 | Screen Display |
| | | | W | SF3 | Cash Deposit |
| | | | W | SF4 | Read Card |
| | CC2 | Security | W | SF1 | Dispense Cash |
| | | | W | SF3 | Cash Deposit |
| | | | M | SF4 | Read Card |
| | | | S | SF5 | Access Customer Data |
| | CC3 | Fault Tolerance | M | SC1 | Accuracy |
| | | | W | SF3 | Cash Deposit |
| | | | M | SF4 | Read Card |
| | | | S | SF5 | Access Customer Data |
| | | | S | SF6 | Account Handler |
| | CC4 | Recoverability | W | SC2 | Efficiency |
| | | | S | SF3 | Cash Deposit |
| | | | M | SF4 | Read Card |
| | | | M | SF5 | Access Customer Data |
| | | | M | SF6 | Account Handler |
| | CF1 | Control ATM | W | SC1 | Accuracy |
| | | | M | SC2 | Efficiency |
| | | | M | SC3 | Operability |
| | | | M | SF1 | Dispense Cash |
| | | | W | SF3 | Cash Deposit |
| | | | W | SF4 | Read Card |
| | | | W | SF5 | Access Customer Data |
| | | | M | SF6 | Account Handler |
| | CF2 | Handle ATM Card | W | SC3 | Operability |
| | | | W | SF1 | Dispense Cash |
| | | | W | SF3 | Cash Deposit |
| | | | M | SF4 | Read Card |
| | | | M | SF5 | Access Customer Data |
| | | | W | SF6 | Account Handler |
| | CF3 | Handle Customer Data | S | SC1 | Accuracy |
| | | | W | SC3 | Operability |
| | | | W | SF3 | Cash Deposit |
| | | | S | SF5 | Access Customer Data |
| | | | W | SF6 | Account Handler |
| | CF4 | Interface | M | SF1 | Dispense Cash |
| | | | M | SF2 | Screen Display |
| | | | M | SF3 | Cash Deposit |
| | | | W | SF4 | Read Card |
| | CF5 | Manage Account | M | SC1 | Accuracy |
| | | | S | SC2 | Efficiency |
| | | | M | SF1 | Dispense Cash |
| | | | W | SF2 | Screen Display |
| | | | M | SF3 | Cash Deposit |
| | | | M | SF4 | Read Card |
| | | | M | SF5 | Access Customer Data |
| | | | M | SF6 | Account Handler |

**Table 12: Feature parameters of OOD [11]**

| No | Class/Object/ Code/Interface | Feature Parameter |
|---|---|---|
| 1 | Class | Quantity of class |
| 2 | | Add super class |
| 3 | | Add subclass |
| 4 | | Add abstract class |
| 5 | | Mend class |
| 6 | | Combination of classes |
| 7 | | Choose of classes |
| 8 | | Design of class |
| 9 | | Dependence of classes |
| 10 | | Levels of class |
| 11 | | Visit of class |
| 12 | Object | Quantity of objects |
| 13 | | Create object |
| 14 | | Access of a cluster of objects |
| 15 | | Collaboration of object |
| 16 | Code | Repetitive code |
| 17 | | Function design |
| 18 | | Conditional logic |
| 19 | | Branch statement |
| 20 | | Rely on basic types of variables |
| 21 | Interface | Add interface |
| 22 | | Uniform interface |

**Table 13: Part of Dependence (Contradiction) Matrix [11]**

| | Affected or dependent parameters | | | | |
|---|---|---|---|---|---|
| Improving parameters | | Uniform interface | Combination of classes | Add class | Function design |
| | Add class | | 8, 9, 18 | | |
| | Crate Object | 1, 3 | 4 | 1, 3, 4 | 12, 14, 20, 21, 23 |
| | Dependence between Classes | | 6, 15 | 1, 5, 3 | |

**Table 14: Part of Design patterns of OOD [11]**

| No | Design Pattern | Explanation |
|---|---|---|
| **1** | Factory method | Create a single entity |
| **2** | Singleton pattern | Create a secure object exactly |
| **3** | Abstract factory pattern | Create a cluster of objects |
| **4** | Prototype pattern | Involving "mixed and matched" |
| **5** | Builder pattern | Construct complex object |
| **6** | Façade pattern | Provide interface for the collection of objects |
| **7** | Decorator pattern | Increase object in the run-time |
| **8** | Composite pattern | Express tree structure of the object |
| **9** | Adapter pattern | Simplify the use of external feature parameters |
| **10** | Flyweight pattern | By using multiple examples to minimize space consumption |
| **11** | Proxy pattern | Provide a surrogate or placeholder for another object to control access to it |
| **12** | Bridge pattern | Abstract and realization of separation |
| **14** | Iterator pattern | Visit the set of elements |
| **20** | Strategy pattern | Make algorithm implementation independent on its customers |
| **21** | Template method pattern | Make subclass which don't change the structure of algorithm can re-define some specific steps of the algorithm |
| **22** | Memento pattern | Capture the internal state of an object and save the state out of the object |
| **23** | Visitor pattern | Through a unified interface to visit different types of elements of the operation |

In case of adding class, the dependence of classes could be improved. In the contradiction matrix, we choose the "Add class" from the column (to account for SC3) and choose the "Dependence of class" from the row (to account for SC1), the solution is being 1, 5, 3, and the concrete design plans is the factory pattern, builder pattern and abstract factory pattern. We can choose the factory pattern "Create a single entity" to solve this contradiction.

## 2.4 Class Diagram

UML class diagrams allow us to denote the static contents of, and relationships between classes. In a class diagram we can show the member variables, and member functions of a class. We can also show whether one class inherits from another, or whether it holds a reference to another. In short, we can depict all the *source code dependencies* between classes [12]. This can be valuable. It can be much easier to evaluate the dependency structure of a system from a diagram than from source code. Fig.  shows a simple class diagram of part of the above ATM system. The diagram immediately tells that WithdrawTransaction talks to a CashDispenser interface. Note the convention of horizontal association and vertical inheritance. The diagram is separated into three distinct zones. The transactions and their actions are on the left, the various UI interfaces are all on the right, and the UI implementation is on the bottom.

The functions of the class design phase are implemented according to their final priorities as methods of the subsystem design phase.

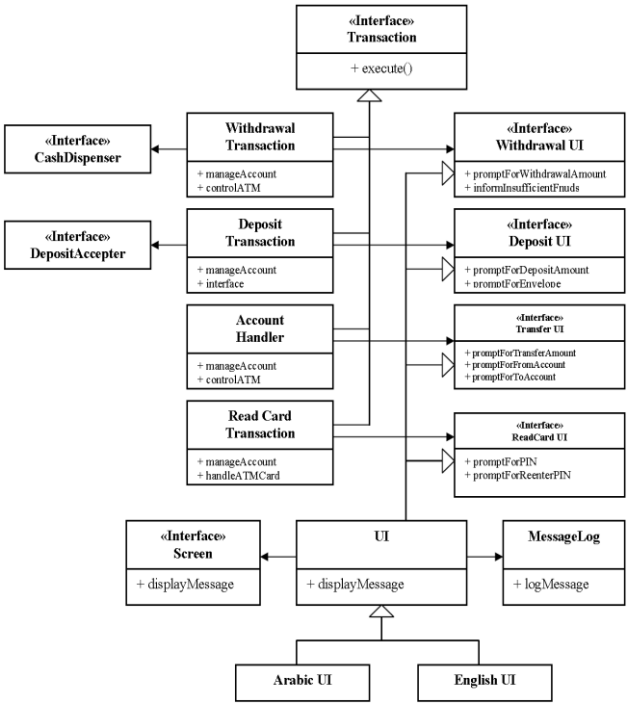



**Fig. 18: A simple class diagram of the above ATM system [12]**

**UI.java**

```
public class UI implements
   readCardUI,withdrawalUI,depositUI,transferUI
{
   private Screen itsScrean;
   private MessageLog itsMessageLog;
   public void displayMessage (String message)
   }
     itsMessageLog.logMessage (message);
     itsScreen.displayMessage (message);
   }
}
```

## 2.5 Use Cases

The ATM system behavior can be specified by stating in use cases how users interact with the system; it is not needed to know anything about the inside of the ATM at all. Use cases specify desired behavior; they do not dictate how that behavior will be carried out. The great thing about this is that it lets you (as an end user and domain expert) communicate with your developers (who build systems that satisfy your requirements) without getting hung up on details. Those details will come, but use cases let you focus on the issues of highest risk to you [13]. In the UML, all such behaviors are modeled as use cases that may be specified independent of their realization. A use case is a description of a set of sequences of actions, including variants that a system performs to yield an observable result of value to an actor.

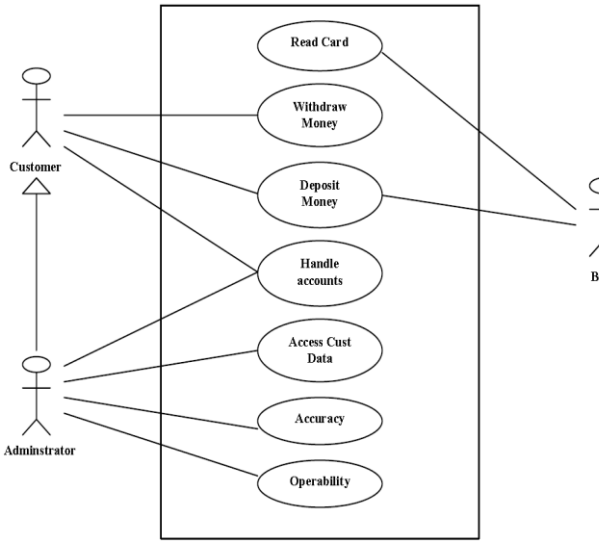



**Fig. 19: Use case diagram for the Subsystem functions and constraints [12]**

The use case diagram in Fig. 19 exhibits the important functions and important constraints and the actors of each.

## 4. CONCLUSION

It has been known that requirements traceability in the object oriented software development process is an important issue across all the design and development phases. Not only the requirements traceability is required but also the impact of these requirements on the design items which improves the design quality on the basis of limited resources. There isn't much research done relating to this work.

A new integrated methodology that integrates the most recent quality tools with the object oriented development process for the purpose of performance improvement is utilized.

The new integrated methodology in this case study, Design of ATM machine object oriented software, improves the quality of the design process by using the following tools and techniques:

- Three phases of object oriented system design are used.
- The automated Analytical Hierarchy process (AHP) is used to prioritize Customer Requirements
- Fuzzy set theory is used for the correlation between customer requirements and technical requirements to solve for the vagueness and inaccuracies among the members of the design team.
- The QFD, House of Quality is automated to save time and effort and ensure accuracy of calculations
- Traceability block diagram, forward traceability, and backward traceability tables are used to keep track of customer requirements through all phases of design.
- Class diagram and use cases of the development process are used to show graphically the important design items that should be taken care of.
- TRIZ methodology is used to solve the technical contradiction of the object oriented design process

Results obtained from this new methodology in the subsystem design phase show the following:

- Priorities of subsystem functions and constraints are:
  - Read Card, Account Handler, Dispense Cash, Cash Deposit, Access Customer Data, Screen Display

  - Accuracy, Efficiency ,Operability
- The subsystem classes will include attributes and methods of the following design phase (class design phase) according to their priorities as follows:
  - Manage Account, Handle Customer Data, Control ATM, Handle ATM Card, Interface
  - Fault Tolerance, Recoverability, Security , Reusability
- TRIZ methodology for OOD is used to solve the contradiction between the two conflicting subsystem constraints: Accuracy and Operability by using the Factory Pattern to solve for the contradiction between Dependence between classes and Add class feature parameters from the table explained before

## 3. REFERENCES

[1] Nancy R. Mead, Software Engineering Institute, Carnegie Mellon University, 2008, *"Requirements Prioritization Case Study Using AHP"*

[2] Nancy R. Mead, Software Engineering Institute, Carnegie Mellon University, 2008, *"Requirements Prioritization, Introduction"*

[3] Saaty, T. L., 1980, *"The Analytic Hierarchy Process",* New York, NY: McGraw-Hill.

[4] Karlsson, J., 1996, *"Software Requirements Prioritizing,"* 110-116. *Proceedings of the Second International Conference on Requirements Engineering (ICRE'96).* Colorado Springs, CO, April 15-18, 1996. Los Alamitos, CA: IEEE Computer Society.

[5] Karlsson, J. & Ryan, K., 1997, *"Cost-Value Approach for Prioritizing Requirements" IEEE Software 14,* 5 (September/October 1997): 67-74.

[6] Kanishka Bedi, U21Global, Singapore, 2006, *"Automating the Quality Function Deployment House of Quality"*, U21 Global, Graduate School for Global Leaders

[7] Ming-Chyuan Lin , Chieh-Yuan Tsai , Chao-Chun Cheng , and C. Alec Chang , 2004 *"Using Fuzzy QFD for Design of Low-end Digital Camera",* © 2004 Chaoyang University of Technology, ISSN 1727-2394- International Journal of Applied Science and Engineering - 2004. 2, 3: 222-233.

[8] Xiaoqing (Frank) Liu, Yan Sun, Praveen Inuganti, and Chandra Sekhar Veera, University of Missouri-Rolla, Yuji Kyoya, Toshiba Corporation, 2007. *"A Methodology for Tracing the Requirements in the Object-Oriented Software Design Process Using Quality Function Deployment"*, SQP VOL. 9, NO. 4/© 2007, ASQ.

[9] Akao, Yoji, ed., 1990, *"Quality function deployment: Integrating customer requirements into product design"*, Cambridge, Mass.: Productivity Press.

[10] Liu, X. 2000, *"Software Quality function deployment",* IEEE Potentials (Dec.-Jan.): 14-16

[11] Ma Jianhong, Zhang Quan, Wang Yanling, Zhang Wei, 2009 IEEE, *"Research and Application of the TRIZ Contradiction Matrix in OOD",* DOI 10.1109/WCSE.2009.244.

[12] Robert Cecil Martin, Object Mentor Inc., 2002, *"UML for Java Programmers".,* Prentice Hall, Englewood Cliffs, New Jersey.

[13] Grady Booch, James Rumbaugh, Ivar Jacobson, 1998, *"UML User Guide "*, Addison Wesley.