3D Spiral Software Lifecycle Model Based on QFD Method

ANDREEA CRISTINA IONICA
Department of Management, University of Petrosani, Universitatii Str., no.20, Petrosani, 332006, Romania,
email: andreeaionica2000@yahoo.com

MONICA LEBA
Department of Electrical and Computer Engineering, University of Petrosani, Universitatii Str., no.20, Petrosani,
332006, Romania,
email: monicaleba@yahoo.com

Abstract:
In our research we focused on developing a methodology to apply the QFD method for software quality evaluation. We introduce a lifecycle model that includes in its representation the quality part. This new lifecycle has a 3D representation made out of multilevel circles spaced by an offset. The offset value is a comprehensive quantification, based on a mathematical model, of the accomplishment degree of the customers' requirements. Without pretending to having achieved the “perfect” tool for software quality evaluation, surely subject to improvements, we have developed the first version of the 3D lifecycle QFD embedded model software tool.

Keywords: software lifecycle, QFD, 3D spiral model, quality evaluation, continuous improvement

1. Introduction
The past, present and future of the software development process...
Regarding the past, the following milestones are worth being mentioned:
- From the mid 1950s the word “software” was used referring to the programs written for computers by specialised companies.
- In the early 1960s the software industry expanded given the increased demand from the universities, government and business customers.
- The 1968 NATO’s Software Engineering Conference first introduced the term “Software Engineering” meant to provoke thought regarding the perceived software crisis at the time.
- The software industry grew exponentially in the mid 70’s with the rise of the personal computers.
- In the mid 1980s the Software Development Life Cycle as a consensus for centralized construction of software was brought forward. The main concept introduced in this period was “software engineering”, defined as: applying systematic approach from the engineering to the process of software development in order to improve its quality.

Regarding the present we can notice that there is a need to reconsider the definition of software engineering one that should be focused on ways to produce cheaper, better, faster software (figure 1).

![Figure 1. Cheaper, better, faster software triangle](image-url)

The “cheaper” part was a primary concern of the IT industry since the 1990s and currently there are methods and tools employed for software costs reduction.
The “faster” part was also addressed and continues to be a permanent aspect in software development. The approaches revolve around a more efficient
use of hardware resources, optimized algorithms implementation and other technical aspects regarding the hardware and software tools involved in software development processes. The “better” part is related to the quality of the software. This has always been an important research issue in the software development process. While the “cheaper” and “faster” are objective and quantifiable terms, the “better” is subjective and intangible. As such, the most important problem is to understand what better means in quantifiable terms. The solution could arise from the quality tools applied in software evaluation. From all these tools we choose the Quality Function Deployment (QFD) method that was considered by different researchers to be a “structured methodology” [1], a “collection of structured methods” [2], a “systematic method” [3] and a “measurement approach” [3]. Above all, the common perception of QFD is that of a highly complex phenomenon still in evolution, perpetually growing with new applications and customisations. The goal of our research is to contribute to this evolution by proposing a customization of the QFD method for software quality evaluation. In just a few words, the achievements of our research presented in this paper consist in the methodological development of QFD meant to ease the understanding and application of this method in the software development process. For this reason, we introduce a conceptual model that integrates three elements: the QFD method, the spiral software life cycle and the Juran’s quality spiral. This conceptual model contains a mathematical model that determines a global index as quality quantification in terms of degree of the users’ requirements accomplished by the final software product.

2. Literature review
QFD is well known to be the method that transforms the user requirements into design quality, the method that translates the client’s “voice” into engineering characteristics. This is the result of combining Akao’s work [4] in quality assurance and quality control points with function deployment from value engineering. The main goals in implementing QFD method are: discover the spoken or unspoken users’ needs, translate these needs into quality characteristics, built and deliver quality products and services by focusing on customer satisfaction [5]. Here are some ideas about QFD that sparked up our interest in starting the research on its application in software evaluation. QFD is an encompassing method, but “loosely defined and structured, QFD sometimes becomes an art more than a science which makes it difficult for practitioners to use QFD”. [6] The proper use of this method consists of the same activities that are usually performed, “but it replaces erratic, intuitive decision making processes with a structured methodology that establishes all relevant information and experiences that are available” [1]. As “QFD is not a panacea for solving design problems or for developing “perfect” products” [1] yet it can be customized for different types of applications. These could summarize the goal of our research that is to adapt QFD for software evaluation, based on an easy to use methodology. Regarding the existing software products that implement the QFD method, Lai-Kow Chan and Ming-Lu Wu [6] and Georg Herzwurm et.al [7, 8] performed a thorough and comprehensive analysis. From their work we can conclude that the software products implementing QFD method field is very dynamic and the majority of these products are for the purpose of general use of the QFD method. We believe that a possible enhancement, welcomed by the practitioners, would be the customization of these software products for a specific field of application. This type of application field customized QFD software could implement the entire specific characteristics, leading to a software product easy to understand and apply by the field specialists. Taking into account the works studied in QFD [9, 10] we strongly believe that there are no fields in which the QFD method was not applied, tried to be applied or impossible to be applied and “essentially, there is no definite boundary for QFD’s potential fields of applications”. [6] In our previous researches we explored the possibility to apply the QFD method in eLearning systems evaluation.
3. Methods
The software development process is based on life cycle models. There are several life cycle models that are well defined and widely used by software development companies. The most complex one, that actually contains the other models characteristics, is the spiral life cycle model or Boehm’s model.
This spiral model consists of applying the software development activities in loops in a two-dimensional representation. Each loop represents a new development phase and the radius of the loop represents the cumulative costs of the software development.
From the array of cheaper, better, faster software triangle in the spiral life cycle model is “visible” only the cost (“cheaper”) part, while the quality (“better”) and time (“faster”) parts are integrated although not “visible”.

We propose a life cycle model that includes in its representation the quality part, quantified through the evaluation of users’ requirements accomplishment. In the original spiral model the quality evaluation is an activity included in each development phase. In order to represent explicitly the quality, we included a third dimension to the original model, inspired from the conceptual model of continuous improvement of the Juran’s quality spiral representation. This third dimension represents the step to a new phase in the software development, a visual representation of the quantified quality by an index called offset.
Therefore, we have a three-dimensional life cycle model that is made out of multilevel plane circles spaced by the offset. Each plane circle represents a software development phase and has its own radius representing the development costs of the phase.
The offset is calculated using the quality function deployment (QFD) method that is embedded in all the activities of each phase. (figure 2)

In order to determine the offset, a mathematical model was developed and implemented in a software application. The variables and their signification are presented below:
n = number of customer requirements
m = number of quality characteristics
c1 = value for Mandatory requirement; c2 = value for Desirable requirement; c3 = value for Optional requirement; c4 = value for future Enhancement requirement
\( CR(i)_{n,m} \) = customer requirements classified according to Kano model: \( CR(i) \in \{c_1, c_2, c_3, c_4\} \)

\( QC(j)_{m,n} \) = quality characteristics distribution functions according to the difficulty level
cor1 = value for strong positive correlation; cor2 = value for strong correlation; cor3 = value for neutral correlation; cor4 = value for strong negative correlation; cor5 = value for negative correlation;

\( TQC(i,j)_{n,m} = \) correlation matrix of quality characteristic i by the quality characteristic j:
\( TQC(i,j) \in \{cor_1, cor_2, cor_3, cor_4, cor_5\} \)
\( ICQ(i,j)_{i=1,...,n,j=1,...,m} = \) influences matrix or how much of the customer requirement \( i \) is assigned to the quality characteristic \( j \): \( ICQ(i,j)\in [0,1] \)

\( AQ(j)_{j=1,...,m} = \) achievement matrix of the quality characteristics: \( AQ(j)\in [0,1] \)

\( RCQ(i,j)_{i=1,...,n,j=1,...,m} = \) relationship matrix or how much of the customer requirement \( i \) is achieved by the quality characteristic \( j \): \( RCQ(i,j)\in [0,1] \)

\( \text{offset} = \) the degree of customer requirements achievement by the quality characteristics

The first quadrant in the Boehm’s spiral life cycle model regards the objectives, alternatives and constraints [11]. In our 3D spiral model from figure 2, the first quadrant contains the following requirements analysis and specification activities from the QFD method: objectives definition, customer requirements (CR) capture and CR classification based on Kano model.

The activities of the first quadrant are all related to customer requirements capture and classification, and form an essential phase in the software development process [12]. If the customer requirements are rigorously captured and classified then the next phases are correctly developed. In this regard, for our test applications of the 3D spiral life cycle, we have used the interview and questionnaire techniques, taking into account the model of “Summary of structured customer requirements” [7, 8]. Also, we have based the classification of the requirements on the Kano model, adapted for software products. We have considered four priorities related ranking levels: mandatory, desirable, optional and future enhancement. Results the CR\( a \) matrix.

The second quadrant in the Boehm’s spiral life cycle model contains the evaluation of alternatives and identification of risks [11]. The second quadrant from our model (figure 2) is dedicated to the design phase and contains the following activities from the QFD method: quality characteristics (QC), also called technical characteristics, identification and technical correlations matrix establishment.

For each quality characteristic we have defined a level of achievement difficulty, ranked from hard, medium to easy, and an improvement direction that could be “maximize”, “minimize” or “maintain”.

Usually, in the system evaluation the level of achievement difficulty of the quality characteristics is not taken into account. In our model, we have considered different distribution functions for each difficulty level, like quadratic functions for hard and easy levels and linear function for medium level. According to the software features being evaluated the best suited distribution functions for the difficulty levels can be chosen. If the difficulty levels are not considered relevant, the distribution functions will all be linear. Results the QC\( m \) matrix.

The technical correlations matrix is valuable in the context of software product evaluation providing a means to both estimate the influences and integrate the interdependencies between the quality characteristics. For the software products there are cases of non bi-univocal dependencies between the quality characteristics. So, the technical correlation matrix can be either symmetrical, in the case of bi-univocal dependencies, or asymmetrical, in the case of non bi-univocal dependencies.

The model will deal with five types of dependencies: strong negative, negative, neutral, positive and strong positive. These dependencies are introduced in the mathematical model as rated values reported to a maximum estimated possible influence. Results the TQC\( m \times m \) matrix.

The third quadrant from Boehm’s spiral is related to development and evaluation activities [11]. The third quadrant from the 3D spiral (figure 2) is also related to the development, evaluation and testing activities. From the QFD model the third quadrant contains the activities involved in establishing the relationship matrix between the CRs and QCs.

During the development of the software product all the customer requirements are assigned to one or more quality characteristics. These assigned percentages are all quantifiable, because the quality characteristics are identified starting from the users’ requirements in such a way that the result is a functional software product that satisfies the users’ needs. Results the ICQ\( n \times m \) matrix.

During the evaluation and testing activities the accomplishment degree of each quality characteristic, using known testing tools is being determined. Results the AQ\( m \) matrix.

The CRs and QCs relationship matrix contains the products between the accomplishment degree of QCs and the assigned percentages of CRs. Results the RCQ\( n \times m \) matrix.
The fourth quadrant from Boehm’s spiral represents the planning for the next phase [11]. The fourth quadrant from the 3D spiral (figure 2) contains the following activities from the QFD method: evaluation, benchmarking and simulation. The evaluation is achieved by computing the offset using the relation:

$$\text{offset} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} RCQ(i,j) \cdot CR(i)}{\sum_{i=1}^{n} CR(i)}$$

Where:

$$RCQ(i,j) = ICQ(i,j) \cdot AQ(j) + \sum_{i=1}^{n} TQC(i,j) \cdot AQ(j)$$

The offset value is a comprehensive quantification by the resulted software product of the accomplishment degree of the customers’ requirements. The offset could be a comparison basis for different software products also useful in the benchmarking analysis. The simulation provides the inputs for the next phase, which represents in our model the transition to a new subversion (a new plane in the 3D spiral). The simulation could also be considered a means to perform a risk analysis as part of Boehm’s spiral.

4. Results and discussions

The main contribution of our research is the 3D spiral lifecycle model and methodology. From the development and application of this model stems the other contributions that will be highlighted below.

Besides the methodological aspects of the model, previously presented, this model offers a suggestive 3D visual representation of the current stage of the software development process. For the 3D visualisation there are equipments available that are nowadays of common use.

Moreover, the 3D representation is also useful for the graphical representation of the software development process efficiency, because the radius of each plane circle represents the costs of each phase activities. So, if the representation is in the form of an inverted cone (figure 3.a) the process has increasing improvement costs and decreasing efficiency.

An acceptable and desirable version would be a spiral form that can be inscribed in a cone (figure 3.b), which means that the process has decreasing improvement costs and increasing efficiency. Such a form is a representation of the continuous improvement principle, synthetically containing the idea that generated our research.

We’ll consider an intermediary version of the 3D spiral to be the one having the approximated form of a cylinder (figure 3.c). This is the version obtained should we not take into account the development costs.

![Figure 3. 3D spiral lifecycle forms: a) increasing costs; b) decreasing costs; c) constant costs](image)

Another visual estimation offered by the 3D representation of the software life cycle is “how much” the software product satisfies the customers’ requirements after n phases of development. This is measured by a mean that we called offset. There are two types of offsets, an absolute and a relative one. The relative offset is a measure of the improvements resulted from the
current phase’s activities and represents the distance between two consecutive phases, in our model represented by two consecutive planes. The absolute offset is a measure of the cumulative step by step improvements and is, in fact, the 3D spiral’s height. The existence of a preset or an objective offset leads to the identification of the right moment for the release of the current version software product. The opportunity to produce a new version after the release of the current one can be analysed, according to past experiences and new requirements.

When starting a new version two ways can be followed: one consists in starting the new version as a completely new software product employing a new spiral model (the case of major changes in the requirements of the new versions) while the other consists in continuing the spiral by translating the xOy plane along the z axis till it reaches the starting point of the new version (the case of maintaining a global view of the developed software product throughout all the versions).

The value of the offset is computed according to the developed mathematical model based on the general principles of the QFD method and its particularities in software product application. Thus, the inputs of the mathematical model in order to determine the offset are: \( n, m, CR_n, QC_m, TQC_{m \times m}, ICQ_{n \times m}, AQ_m \). All these generate the relationship matrix \( RCQ_{n \times m} \) as a state and the offset as output.

Compared with other approaches, this mathematical model contains the following:

- The use of different distribution functions for the characterisation of the different difficulty degrees of the quality characteristics;
- The generalisation of the correlation matrix with the asymmetrical form due to the different dependencies between the quality characteristics;
- The measurement of the achievement level of each quality characteristic;
- The use of the influence matrix between the customers’ requirements and the quality characteristics. The influence matrix together with the achievement matrix determines the relationship matrix. Thus, the relationship matrix contains exact values resulted from measurements instead of generally used estimations \( \{0, 1, 3, 9\} \).

Now, we present some key elements that prove the applicability of the 3D spiral model. The general purpose of QFD based software products is for the evaluation of a wide variety of products and services. Without pretending to having achieved the “perfect” QFD software tool [7] we have developed the first version, surely subject to improvement, of the 3D lifecycle QFD embedded model software tool.

We choose to present the ground up development of a software product. The idea behind this software product was a new type of human machine interface. For this reason, we have started from the users’ requirements capture, by using the interview and questionnaire techniques applied to graduate students. The results of the interviews and questionnaires showed that the users expect an online virtual environment that allows them to interact with 3D objects by gestures. These led us to the name M.O.V.E. (My Online Virtual Environment) for this new application.

The interviews and questionnaires also generated the customers’ requirements matrix \( CR_n \) consisting of all the identified requirements grouped in priority classes (figure 4.b). The design team established the main quality characteristics (figure 4.c) of the system that meet the users’ expectations. As a result the quality characteristics matrix \( QC_m \) created also contains the achievement difficulty level for each characteristic. These characteristics were assigned to routines. Each routine has well defined purpose, inputs and outputs. The “routines map” is useful in generating the influences matrix \( ICQ_{n \times m} \) (figure 5.b), that contains the way of distributing the users’ requirements on quality characteristics, and the correlation matrix \( TQC_{m \times m} \) (figure 4.d), that contains the interdependencies between the characteristics. The coding team implemented the routines and the testing team applied different methods of evaluation and validation of the routines, prompting the achievement level of each characteristic and thus the achievement matrix \( AQ_m \) (figure 5.a). The achievement matrix and the influences matrix determine the relationship matrix \( RCQ_{n \times m} \). This matrix shows the accomplishment level of the users’ requirements by the implemented quality characteristics. Based on all these matrices we can compute the offset (figure 5.c).
Figure 4. QFD software tool: a) QFD method; b) CR; c) QC; d) Correlation matrix

Figure 5. QFD software tool: a) QC achievement matrix; b) Influence matrix; c) Relationship matrix and resulting offset
The offset is a measure of customers’ satisfaction level regarding the software product. The offset only by its value is not very useful, but becomes important in the following contexts:

- If a preset target value was set, representing the acceptability threshold of the software product, then the computed absolute offset will be compared with the target value in order to decide whether to release the product or to go on to the next phase;
- If a multiple development phases product is analysed then several relative offsets are computed in order to evaluate the results of the continuous improvements achieved.

Using this method for software products development evaluation, could be a reference database containing acceptability offset values for different categories of software applications.

5. Conclusions

The methodology discussed in this paper contains many intrinsic elements from the fields of quality management and software engineering, but we have highlighted mainly the novelty and originality issues presented by our method, the others being considered a well known background theory. From the quality function deployment method we did not include in our method the benchmarking since the comparison of resulted offsets for different similar software products can be considered a global benchmarking form.

There are several aspects from our methodology that could be the subject of future researches, like: a more thorough benchmarking, the simulation of possible enhancements in the next phase of software product development based on the inverted mathematical model and the development of similar methodologies customized for other types of products/services.

As it was stated, the quality is in everything. The last years witnessed the introduction of computers with specific software applications in all activity fields. So, any improvement in the software products quality means improvement in the related products/services quality.

References