Embedding Quality Function Deployment In Software Development: A Novel Approach

Okonta, Okechukwu Emmanuel*, Ojugo, Adimabua Arnold+, Wemembu Uchenna Raphael*, Ajani Dele*

*Federal College of Education (Tech.) Asaba, Delta State
+Federal University of Petroleum, Warri, Delta State

Abstract

Software development differs widely in concept, requirement and framework. Therefore the software engineer has enormous task in engineering functional software that can work and be delivered on time..This paper focuses on how customers' voice can be heard in order to reduce development and manufacturing costs, improve product quality, provide features that satisfy customer needs, and reduce development time. Quality Function Deployment has proven very successful in producing products that appeal to customers. Metaphorically, the customer speaks one language and the manufacturer speaks another. Quality Function Deployment provides linguistic continuity from customer to manufacturer and brings corporate knowledge to bear on the product that achieves multifunctional consensus. This paper adopted a new approach by extending the Quality Function Deployment matrix beyond the House of Quality.

1.0 Introduction

The reliance of modern Society on Computer System and the dependency on software makes it permanent that software engineers, researcher and software development organizations should devise processes to make software products and their quality unique, reliable and free from developmental problems. Serious efforts should be made to fabricate tools to help this development and reduce the time that it takes the finished products to get to the customers. This paper provides a novel ***Quality Function Deployment*** model and framework that will aid the development process [2].

1.1 What is Quality Function Deployment (QFD) ?

Quality Function Deployment (QFD) was developed by Yogi Akao [1] in Japan in 1966 and by 1972 the power of the approach had been well demonstrated by Mitsubishi Heavy Industries Kobe Shipyard (Sullivan 1986 and 1978). The first book on the subject was published in Japanese and later translated into English in 1994 (Mizuno and Akao 1994)[1].

Quality Function Deployment is a customer-oriented approach to product and service innovation. It guides managers through the conceptualization, creation, and realization of new products and services. The QFD process encourages you to gain an

in-depth understanding of the requirements of your customers needs and wants thus enabling you to prioritize the features and benefits of your product or service to these requirements, meeting customer needs and providing superior value. This focus on satisfying the customer's needs places an emphasis on techniques such as Quality Function Deployment to help understand those needs and plan a product to provide superior value.

The "*voice of the customer*" is the term to describe these stated and unstated customer needs or requirements. The voice of the customer is captured in a variety of ways: direct discussion or interviews, surveys, focus groups, customer specifications, observation, warranty data, field reports, etc. This understanding of the customer needs is then summarized in a product planning matrix or "house of quality". These matrices are used to translate higher level "what's" or needs into lower level "how's" - product requirements or technical characteristics to satisfy these needs.

While the Quality Function Deployment matrices are a good communication tool at each step in the process, the matrices are the means and not the end. The real value is in the process of communicating and decision-making with Quality Function Deployment. Quality Function Deployment is oriented towards involving a team of people representing the various functional departments that have involvement in product development: Marketing, Design Engineering, Quality Assurance, Manufacturing or Manufacturing Engineering, Test Engineering, Finance, Product Support, etc. The active involvement of these departments can lead to balanced consideration of the requirements or "what's" at each stage of this translation process and provide a mechanism to communicate hidden knowledge - knowledge that is known by one individual or department but may not otherwise be communicated through the organization. The structure of this methodology helps development personnel understand essential requirements, internal capabilities, and constraints and design the product so that everything is in place to achieve the desired outcome - a satisfied customer. Quality Function Deployment helps development personnel maintain a correct focus on true requirements and minimizes misinterpreting customer needs. As a result, Quality Function Deployment is an effective communications and a quality planning tool.

2.0 Quality Function Deployment Phases

The basic Quality Function Deployment methodology involves four basic phases that occur over the course of the product development process. During each phase one or more matrices are prepared to help plan and communicate critical product and process planning and design information.

Phase 1, Product Planning: Building the House of Quality. Led by the marketing department, Phase 1, or product planning, is also called The House of Quality. Many organizations only get through this phase of a QFD process. Phase 1 documents customer requirements, warranty data, competitive opportunities, product measurements, competing product measures, and the technical ability of the organization to meet each customer requirement. Getting good data from the customer in Phase 1 is critical to the success of the entire QFD process.

Phase 2, Product Design: This phase 2 is led by the engineering department. Product design requires creativity and innovative team ideas. Product concepts are created during this phase and part specifications are documented. Parts that are determined to be most important to meeting customer needs are then deployed into process planning, or Phase 3.

Phase 3, Process Planning: Process planning comes next and is led by manufacturing engineering. During process planning, manufacturing processes are flowcharted and process parameters (or target values) are documented.

Phase 4, Process Control: And finally, in production planning, performance indicators are created to monitor the production process, maintenance schedules, and skills training for operators. Also, in this phase decisions are made as to which process poses the most risk and controls are put in place to prevent failures. The quality assurance department in concert with manufacturing leads Phase 4.

Quality Function Deployment begins with product planning; continues with product design and process design; and finishes with process control, quality control, testing, equipment maintenance, and training. As a result, this process requires multiple functional disciplines to adequately address this range of activities. QFD is synergistic with multi-function product development teams. It can provide a structured process for these teams to begin communicating, making decisions and planning the product. It is a useful methodology, along with product development teams, to support a concurrent engineering or integrated product development approach.

Quality Function Deployment, by its very structure and planning approach, requires that more time be spent up-front in the development process making sure that the team determines, understands and agrees with what needs to be done before plunging into design activities. As a result, less time will be spent downstream because of differences of opinion over design issues or redesign because the product was not on target. It leads to consensus decisions, greater commitment to the development effort, better coordination, and reduced time over the course of the development effort. QFD requires discipline. It is not necessarily easy to get started with. The following is a list of recommendations to facilitate initially using QFD.

3.0 House of Quality

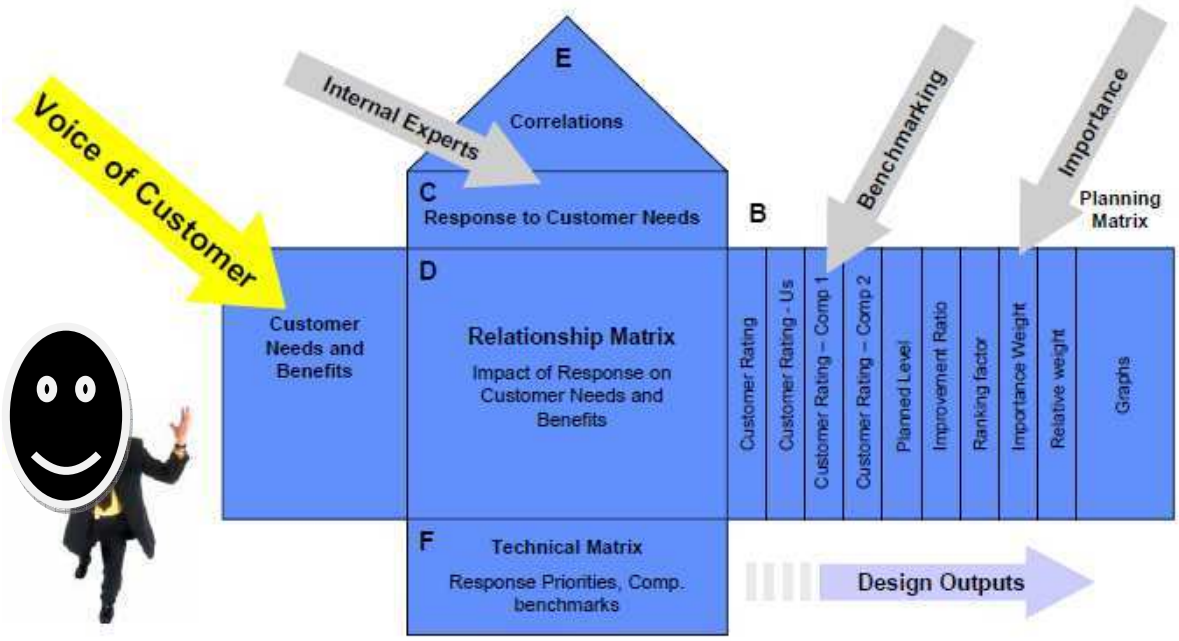

Fig. 1: House of Quality

QFD uses a matrix. This matrix of QFD is called the *"**House of Quality**"*. The first phase in the implementation of the Quality Function Deployment process involves putting together a "House of Quality" [4] such as the one shown above. Clausing [4], one of the pioneers of Quality Function Deployment usage, points out that Quality Function Deployment's primary goal is to overcome three major problems:

1) disregard for the "voice of the customer",
2) loss of information, and
3) different individuals and functions working to different requirements.

Quality Function Deployment consists of well-developed formats (matrices and charts) and a style of organizational behaviour that facilitates a novel response to customer needs [[4]. Quality Function Deployment can benefit an organization by increasing the company's market share and profit. It does this through reduced development and manufacturing costs, improved product quality, provision of features that satisfy customer need, and reduced development time. Quality Function Deployment has proven very successful in producing products that appeal to customers.

Metaphorically, the customer speaks one language and the manufacturer speaks another. Quality Function Deployment provides linguistic continuity from customer to manufacturer and brings corporate knowledge to bear on the product that achieves *multifunctional consensus*.

The key idea presented in [4] is that Quality Function Deployment is needed to deploy customer needs throughout the

corporate communication circle and return to the customer a new product that fully meets those needs. Quality Function Deployment is a product development methodology that systematically deploys customer requirement priorities into the product design and guides production operation on the factory floor. Quality Function Deployment provides a win-win development approach for the manufacturer and customer.

Although these ideas about Quality Function Deployment have been developed for building quality products in the product industry, there are compelling reasons for integrating this methodology in software development processes. This key concept provides a foundation for this paper. It provides the necessary information to transfer these product manufacturing management techniques into a novel model for software acquisition and development.

In Hauser [7], Hauser and Clausing present the "house of quality" (HOQ) as the basic implementation construct of the management approach known as Quality Function Deployment (QFD). The "*house of quality"* provides a conceptual, abstract view of a product design and provides the means for inter-functional planning and communication. The authors point out that the main challenge in design (product design, software design, etc.) is to learn from customer experience and reconcile what customers say they want with what engineers can reasonably build. The house of quality provides such a mechanism for product design, development, and manufacture. Traditionally, the house of quality has been used in the automobile industry and other factory environments, but the same challenge of managing design complexity that Quality Function Deployment tackles in the product industry also plagues the software development industry.

3.1 Methodology

Every Software Engineering should describe a unique set of framework activities for the software processes it adopts and regardless of the process model that is selected software engineers have traditionally chosen a generic process framework so the framework for software development process is no doubt a complicated one. The end product follows a chain of analysis, design, development and testing process. At each stage, it is important to follow a well-defined methodology to ensure a quality end product. For large scale projects, each stage in the whole process is a challenge. At this technical level we look at two software engineering approaches namely - ***The Waterfall Model and Prototyping***

3.2 The waterfall Model

There are times when the requirement of a problem are reasonably well-understood- when work flows from communication through deployment in a reasonably liner fashion. This situation is sometimes encountered when well-defined adaptations or enhancements to an existing system must be made (e.g. accounting software that has been mandated because of changes to government regulations). It may also occur in a limited number of new development efforts but only when requirements are well defined and reasonably stable.

The waterfall model sometimes called the Classic Life Cycle, suggest a systematic sequential approach to software development that begins with customer specification of requirements and progress

through planning, modeling, construction and deployment , culminating in on-going support of the completed software. The waterfall model is the oldest paradigm for software engineering. However, over the past three decades, criticism of this process model has caused even ardent supporters to question its efficiency. Among the problems that are encountered when the waterfall model is applied are:

(1) Real projects really follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does it indirectly. As a result, changes can cause confusion as the project team proceeds.

(2) It is often difficult for the customer to state the entire requirement explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exist at the beginning of many projects.

(3) The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed can be disastrous. Today software is fast paced and subject to never ending stream of changes. The waterfall model is often inappropriate for such work.

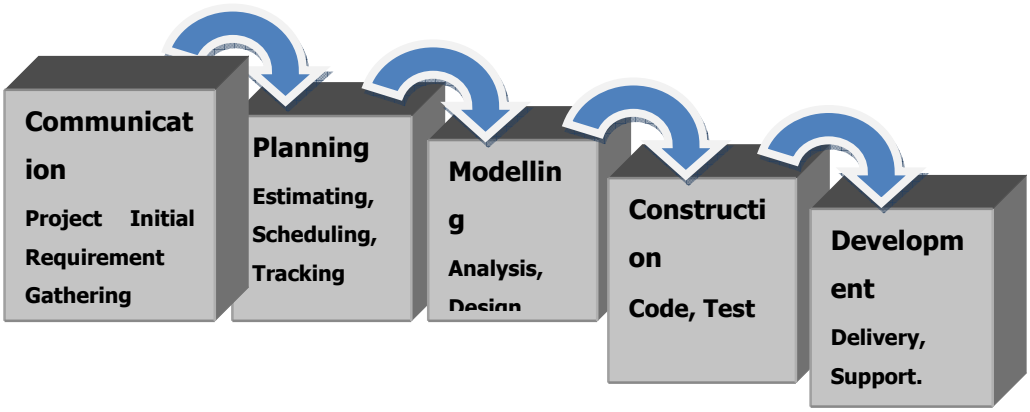

Figure 2 The Waterfall Model

3 Prototyping

Often a customer defines a set of general objectives for software, but does not identify detailed input *abinitio*, processing or output requirements. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system or the form that human machine interaction should take. In this and many other situations, a prototyping paradigm may offer the best approach.

Although prototyping can be used as a standalone process model, it is more commonly used as a technique that can be implemented within the context of any one of the process model like incremental model or Rapid Application Development Model. Regardless of the manner in which it is applied, the prototyping paradigm assists the software engineer and the customer to better understand what it is to be built *when requirements are fuzzy.*

The prototyping paradigm in Figure 3 begins with communication. The software engineer and customer meet and define the overall objective for the software, identify

whatever requirements are known and outline where further definition is mandatory. A prototyping iteration is planned quickly and modeling in form of a "quick design" occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer and end user e.g human interface layout or output display format. The quick design leads to the construction of a prototype. The prototype is deployed and then evaluated by the customer or user. Feedback is used to refine requirement for the software. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

Ideally, the prototype serves as a mechanism for identifying software requirements. If a prototype is built, the developer attempts to make use of existing program fragments or applies tools (e.g. report generators, window managers etc) that enable working programs to be generated quickly. Prototyping can be problematic for the following reasons:

(1) The customer sees what appears to be a working version of the software, unaware that the prototype is held together "with chewing gum and bailing wire", unaware that in the rush to get it working we have not considered overall software quality or long-term maintainability.
(2) The developer often makes implementation compromise in order to get a prototype working quickly. An inappropriate operating system or programming language may be used simply because it is available and known, an inefficient algorithm may be implemented simply to demonstrate capability. However, in this paper we have opted for prototyping model.

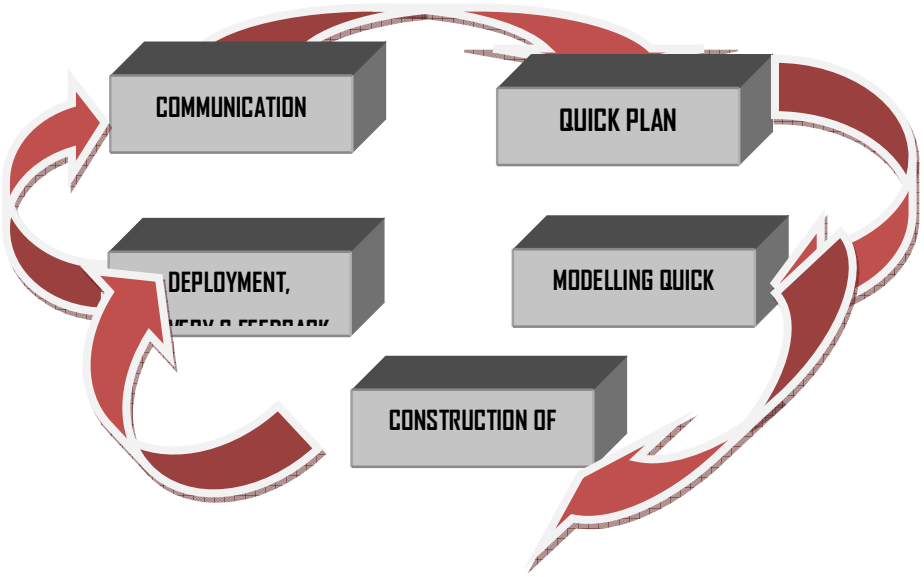

Figure 3: The Prototyping Model

3.4 Quality Function Deployment Steps

Step 1: Customer Requirements - "Voice of the Customer"

The first step in a QFD project is to determine what market segments will be analyzed during the process and to identify who the customers are. The team then gathers information from customers on the requirements they have for the product or service.

Step 2: Regulatory Requirements

Not all product or service requirements are known to the customer, so the team must document requirements that are dictated by management or regulatory standards that the product must adhere to.

Step 3: Customer Importance Ratings

On a scale from 1 - 5, customers then rate the importance of each requirement.

Step 4: Customer Rating of the Competition

Understanding how customers rate the competition can be a tremendous competitive advantage.

Step 5: Technical Descriptors - "Voice of the Engineer" **4.0**

The technical descriptors are attributes about the product or service that can be measured and benchmarked against the competition. **5.0**

Step 6: Direction of Improvement

As the team defines the technical descriptors, a determination must be made as to the direction of movement for each descriptor.

Step 7: Relationship Matrix

The relationship matrix is where the team determines the relationship between customer needs and the company's ability to meet those needs.

Step 8: Organizational Difficulty

Rate the design attributes in terms of organizational difficulty. It is very possible that some attributes are in direct conflict.

Step 9: Technical Analysis of Competitor Products

To better understand the competition, engineering then conducts a comparison of competitor technical descriptors.

Step 10: Target Values for Technical Descriptors

At this stage in the process, the QFD team begins to establish target values for each technical descriptor.

Step 11: Correlation Matrix

This room in the matrix is where the term House of Quality comes from because it makes the matrix look like a house with a roof.

Step 12: Absolute Importance

Finally, the team calculates the absolute importance for each technical descriptor

QFD Deployment in Software Development

Traditional development is incoherent. Quality Function Deployment focuses the development effort on those aspects of the design that are of greatest importance to the customer(s). Quality Function Deployment maintains this focus throughout the entire development process, from requirements to design, to coding, to documentation. Kano et. al. (as cited in Zultner [13,14] provides the Kano model that characterizes three types of requirements (exciting, normal, and expected) based on their influence on customer satisfaction (see Figure 4). Quality Function Deployment is critical because

customers cannot typically articulate all of their exciting and or expected requirements. Quality Function Deployment provides a customer centred software development approach that helps in this requirements elicitation process. Customer requirements can be explored through specification (why customer wants it), exploration (what is required to build great software), and design (how to build great software).

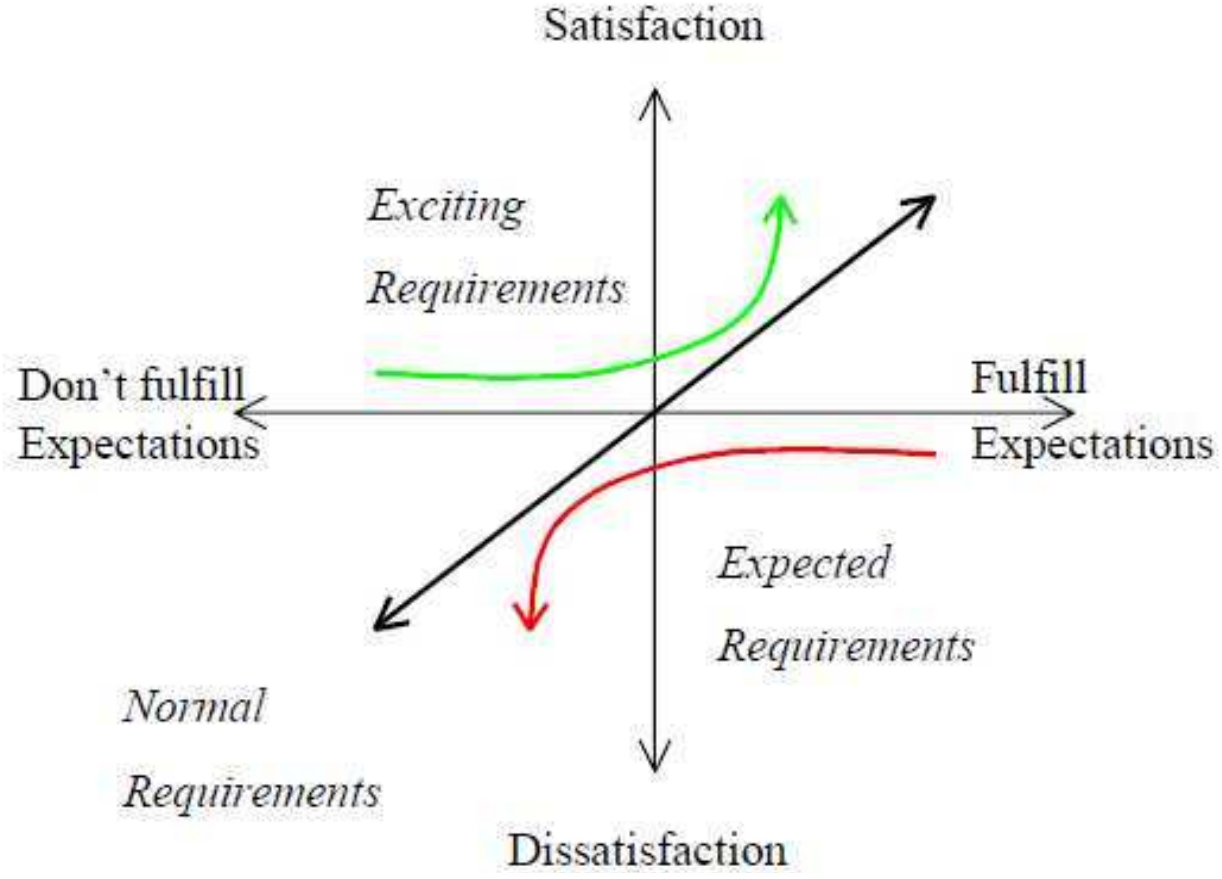

Figure 4 Customer's Satisfaction Requirement

Quality Function Deployment provides a process that conveys requirements throughout a development effort's data and process models. This process includes four steps: capturing raw customer expressions; translation of those expressions into clear, concise, and concrete objective items; organization of the expressions and objective items into a hierarchy that is meaningful to the customer; and prioritization of requirements (e.g. 1 to 5 scale, or by a more advanced method). Customer requirements can come from many sources and consist of several types (exciting, normal, and expected). Software engineers must meet these requirements to

build excellent software. Expected requirements deliver on expectations that the customer will not be disappointed. Normal requirements are the easiest customer requirements to uncover and generally have a direct relationship to customer satisfaction – giving the customer more of a "normal" requirement satisfies him more, giving him less satisfies him less. Exciting requirements are the most difficult requirements to uncover, but "wow" the customer instead of just satisfying them. Kano's model of how these three types of requirements correspond to customer satisfaction is shown in Figure 4.

In order to deploy the "voice of the customer" throughout the software development process it is necessary to employ a chain of matrices (see Figure 5). In this case, the analysis and design is shown as a five phased process of matrices consisting of: customer requirements/ specifications, specifications/software modules, software modules/code, code/unit tests, and unit test/customer requirements matrices.

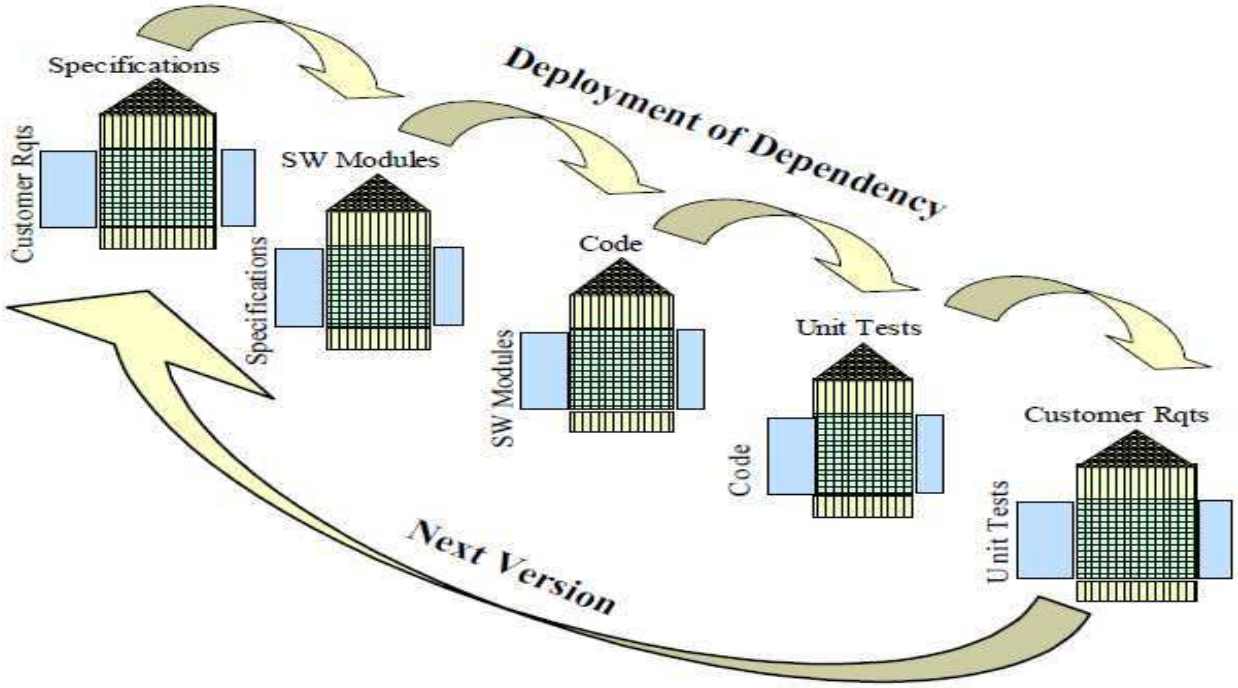

Figure 5 Chain Matrices

The key idea in [13,14] is that Quality Function Deployment is a development process to improve software products, process, and strategy. This is accomplished by making software development more coherent, building quality into the product, and providing rationale for development decisions. Tools are required to deal with the complexity of developing software systems with Quality Function Deployment.

Quality Function Deployment makes software development more coherent so that the best effort of one phase of the software development process feeds the best effort of the next phase of the process, and so on. Each of these best efforts can be directly traced back to what the customer views as

the most important part of the design and forward to future components of the design. Quality Function Deployment gives the software development effort a solid foundation for embedding quality into the product. Software quality can take on many different forms (e.g. functionality, efficiency, reliability, usability, maintainability, and portability), but Pressman [9] emphasizes the following three important points about software quality:

- Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.
- Specified standards define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.
- A set of implicit requirements often goes unmentioned (e.g., the desire for ease of use and good maintainability). If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.

Quality Function Deployment provides a methodology for handling these important points. Quality Function Deployment identifies and implements positive values of customer satisfaction (based on Kano's model). [Puett 2003] emphasizes "Traditional software engineering has generally focused on just removing the "dis-satisfiers" i.e. the defects – this approach is necessary, but not sufficient!" Quality Function Deployment is critical to software development because it provides a methodology for handling the important characteristics of software quality. Quality Function Deployment provides the mechanism for the deployment of quality throughout a software design through the use of linked houses of quality. This linkage helps

4.1 The Quality Function Deployment Solution

Quality Function Deployment provide partial solutions to the following problems:

(1) ***The Software Crisis.*** This crisis is caused by the difficulty in and the failure of successfully evolving complex software systems.

(2) ***Increased Demand for Quality Software***. Software demand is increasing at an astounding rate; industry has found it difficult to find the necessary professional talent required for meeting this software development demand.

(3) ***Inadequate Customer-Developer Communication***. Better communication between software customers and developers is needed. There is currently inadequate communication of requirements and risk throughout the development life cycle.

(4) ***Lack of High Assurance Systems.*** Industry requires ever increasing numbers of high-assurance software systems, particularly in mission-critical activities, business financial transactions, and life-critical medical applications.

4.2 A Novel Approach (Far Beyond House of Quality)

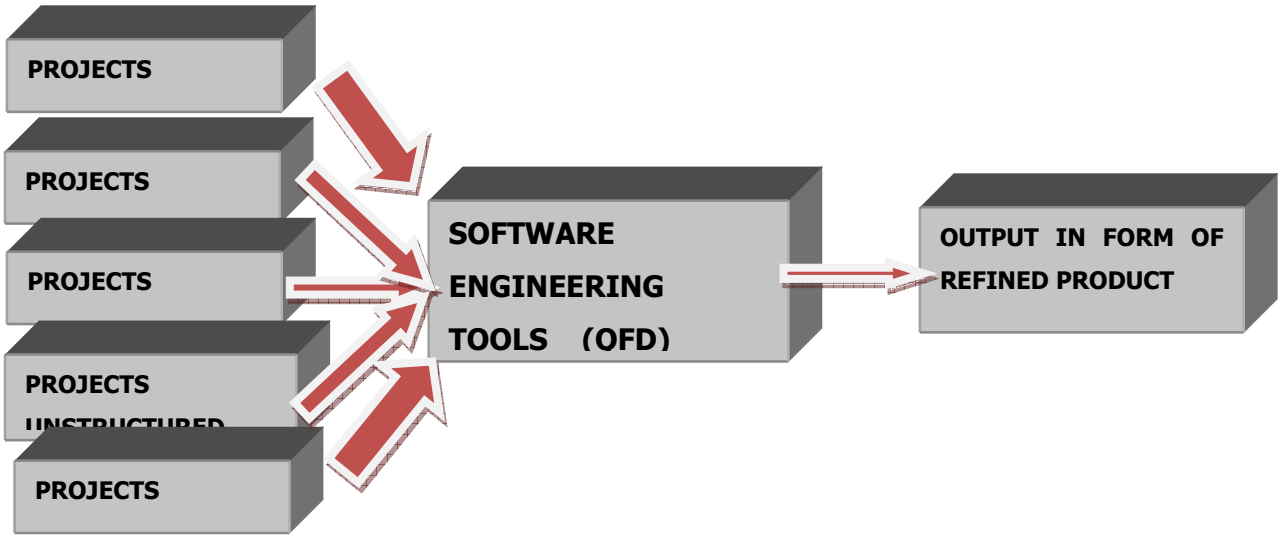

Figure 6 New Model Design Approach

For the current product development project, QFD supports the co-ordination of development assignments for specialized development groups. As such, QFD serves as an organizer of product innovation projects. The most valuable assets of QFD application lie in the: • Coherence and consistency of development assignments, facilitating concurrent engineering and development:

• Concreteness of development assignments that is meaningful to each development group, because they will be stated in their own language;

• Fine-tuning of improvement efforts. The project will only claim development resources that will have a direct impact on customer satisfaction.

Furthermore, in applying QFD you could use several "houses" to include the voice of the customer into downstream development processes. Design attributes are engineering measures of product performance. For example, a computer customer might state that he (she) needs something, which makes it "easy to read what I'm working on". One solution is to provide computer customers with monitors for viewing their work. Design attributes for the monitor might include the measured values for the illumination of alphanumeric characters, for character resolution, for legibility judged at a distance of 50 centimetres (on an optometric scale) etc.

The second house of QFD links these design attributes to the kind of action the company can take. For example, a product-development team might change the product features of the monitor. The product development team could modify the design attribute of legibility at 50 centimetres (as measured on an optometric scale) by changing the number of pixels, the size of the screen, and the intensity of the pixels or the refreshment rate. Changing or replacing monitor screen material also affects the

design attributes. A more radical step might be to eliminate the monitor altogether and provide a system which projects the work onto a wall or onto very small stereoscopic screens which the user wears as goggles.

The third house of QFD could link action to implementation decisions in areas like manufacturing process operations. For example, the third house might be used to identify the manufacturing procedures that produce the material selected for the monitor's screen. The final house of QFD in this example links implementation (manufacturing process operations) to production planning. Finally, substantial benefits from QFD come from re-using matrices. The results of a QFD study will be very useful in the definition and development of the next generation of the product and will have a significant spin-off for related products.

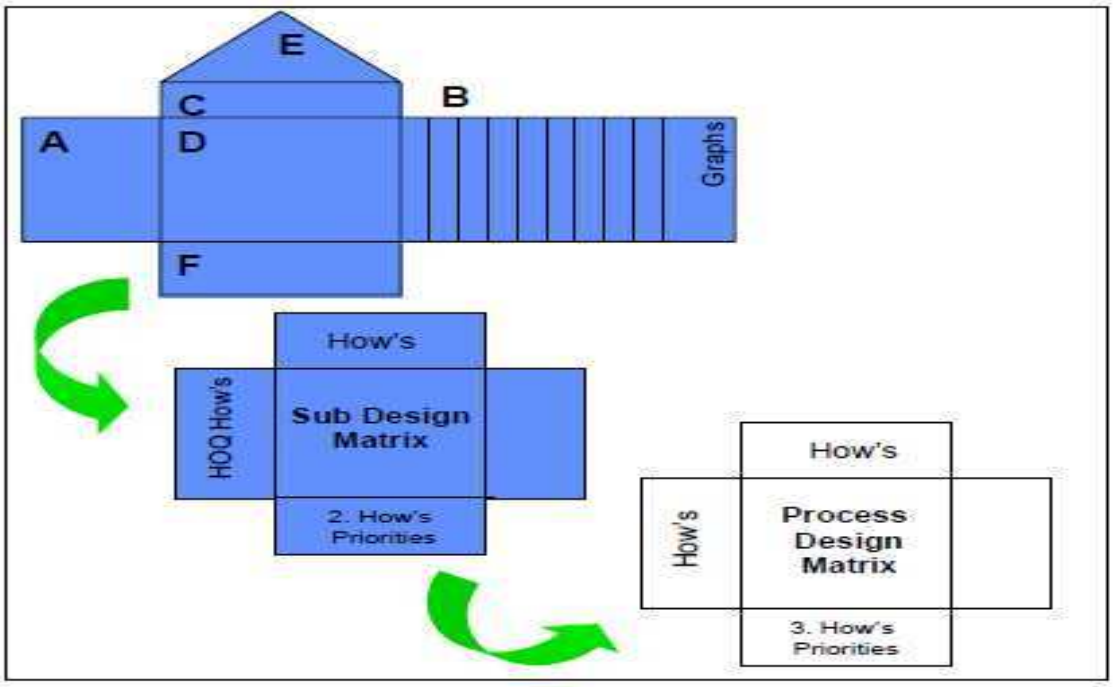

Figure 7 A New Design Matrix

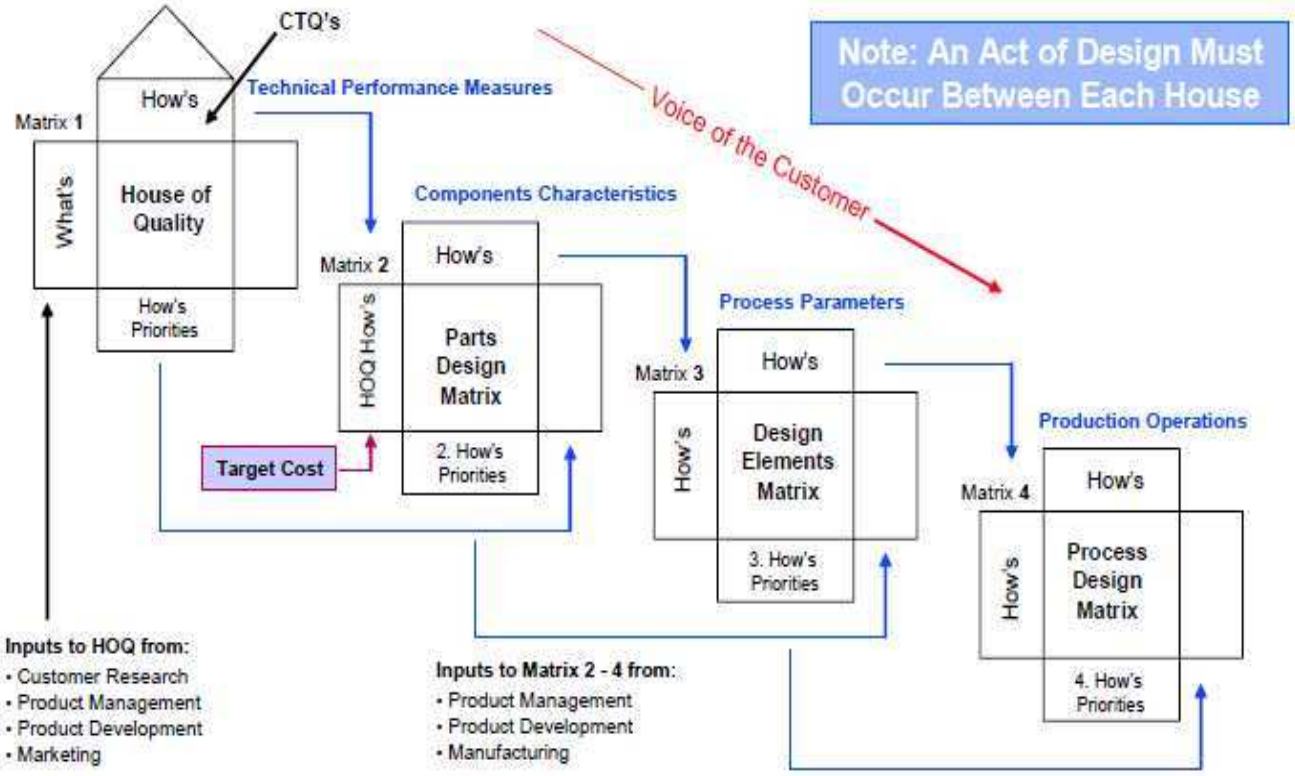

Figure 8: Novel Approach

5.1 Conclusion

QFD is a systematic means of ensuring that customer requirements are accurately translated into relevant technical descriptors throughout each stage of product development. Therefore, meeting or exceeding customer demands means more than just maintaining or improving product performance. It means designing and manufacturing products that delight customers and fulfill their unarticulated desires. This paper has shown when it is properly deploy in the software development industries it will yield good results.

QFD provides open, modular software architecture for future improvements. The flexibility of this tool allows software engineers to interface with existing and future software development tools and models while providing a holistic framework to view and reason about dependency information. Such capability hopes to reduce total life cycle costs, improve software product quality, and reduce evolution timelines. Companies growing into the 21st century will be enterprises that foster the needed novel innovation to create new markets.

References

[1] Akao, Y.(1990.), *Quality Function Deployment: Integrating Customer Requirements into Product Design*, Tamagawa University, Japan translated by Glenn H. Mazur and Japan Business Consultants, Ltd., Productivity Press, Cambridge, MA.

[2] Cohen, L.(1995), *Quality Function Deployment: How to Make Quality Function Deployment Work for You*, Addison-Wesley,.

[3] Conklin J. and Begeman M.(1988), "Issue-Based Information System: A Hypertext Tool for Exploratory Policy Discussion," *ACM Transactions on Office Information System*, Vol. 6, pp. 303-331.

[4] Clausing, D.(1988), "Quality Function Deployment," in Ryan, N. E., ed., *Taguchi Methods and Quality Function Deployment*, American Supplier Institute Inc., Dearborn, MI,.

[5] Harn, M.(1999.), "Computer-Aided Software Evolution Based on Inferred Dependencies" *Ph.D. Dissertation*, Computer Science Department, Naval Postgraduate School, Monterey, CA.

[6] Harn, M., Berzins, V., and Luqi(1999), "Software Evolution Process via a Relational Hypergraph Model," *Proceedings of IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, Tokyo, Japan, pp. 599-604.

[7] Hauser, J. R. and Clausing, D(1988)., "The House of Quality," *The Harvard Business Review*, No. 3, pp 63-73.

[8] Le, Hahn C. (1999), "Design of a Persistence Server for the Relational Hypergraph Model", Masters Thesis, Computer Science Department, Naval Postgraduate School, Monterey, CA.

[9] Pressman, R.(2005), "Software Engineering: A Practitioner's Approach", Fifth Edition, McGraw Hill,.

[10] Puett, J.,(2002) **"**Holistic Framework for Establishing Interoperability of Heterogeneous Software Development Tools and Models," *Proc. 24th Intl. Conf. on Software Engr.*, Orlando FL, pp. 729-730. 446

[11] Puett, J.(2003), **"**Holistic Framework for Establishing Interoperability of Heterogeneous Software Development Tools," *PhD Dissertation,* Computer Science Department, Naval Postgraduate School, Monterey CA,

[12] Rational Software Corporation (2001), "Rational Project_Console Getting Started", *Manual, Version: 2002.05.00, Part Number: 800-025142-000,* 2002.
Sullivan, L.P., 1986, "Quality Function Deployment", Quality Progress, June, pp 39-50.

[13] Zultner, R. E.(1990) , "Software Quality Deployment: Adapting QFD to Software," *Transactions of the Second Symposium on Quality Function Deployment*, Novi MI, 18-19 June 1990, pp. 132-149.

[14] Zultner, R. E.(1992), "Quality Function Deployment (QFD) for Software: Structured Requirements Exploration," in Schulmeyer, G. G. and J. I. McManus, ed., *Total Quality Management for Software*, Van Nostrand Reinhold, New York NY.
Zultner, R. E.(1993), "TQM for Technical Teams," *Communications of the ACM*,
Vol. 36, No. 10, 1993, pp. 79-91.