1999

Software Quality Function Deployment : A Method for Building Better Software

Dean Carruthers

Edith Cowan University

Recommended Citation


This Thesis is posted at Research Online.
https://ro.ecu.edu.au/theses_hons/839
Edith Cowan University
Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study.

The University does not authorize you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following:

- Copyright owners are entitled to take legal action against persons who infringe their copyright.

- A reproduction of material that is protected by copyright may be a copyright infringement. Where the reproduction of such material is done without attribution of authorship, with false attribution of authorship or the authorship is treated in a derogatory manner, this may be a breach of the author’s moral rights contained in Part IX of the Copyright Act 1968 (Cth).

- Courts have the power to impose a wide range of civil and criminal sanctions for infringement of copyright, infringement of moral rights and other offences under the Copyright Act 1968 (Cth). Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.
Honours Thesis
Edith Cowan University
Department of Computer Science
December 10\textsuperscript{th} 1999

Software Quality Function Deployment:
A method for building better software

Dean Carruthers, 0961559
Submission for Bachelor of Science with Honours, Computer Science
Thesis supervisor: Stuart Hope
## Table of Contents

Abstract .................................................................................................................................. 6

Declaration ............................................................................................................................. 7

1.0 Introduction .................................................................................................................... 1

1.1 Definitions and acronyms .......................................................................................... 1

1.2 Background information ............................................................................................ 2

1.2.1 What is quality? ................................................................................................. 2

1.2.1.1 What makes quality software? ....................................................................... 3

1.2.2 What is software development ........................................................................... 6

1.2.3 What is QFD? ..................................................................................................... 8

1.2.3.1 Traditional quality systems ............................................................................ 9

1.2.3.2 Modern quality systems ............................................................................... 10

1.2.3.3 The 3 types of requirements ......................................................................... 11

1.2.3.3.1 Revealed requirements ......................................................................... 11

1.2.3.3.2 Expected requirements ......................................................................... 12

1.2.3.3.3 Exciting requirements .......................................................................... 12

1.2.4 History of QFD ................................................................................................ 12

1.2.5 Variants of QFD ............................................................................................... 13

1.3 Aims of the study ..................................................................................................... 14

2.0 Literature review .......................................................................................................... 15

2.1 Limitation of current SQFD research......................................................................... 15

2.2 What can be learnt from other QFD areas ............................................................... 15

2.3 QFD variants in detail .............................................................................................. 18

2.3.1 Comprehensive QFD ........................................................................................ 18

2.3.1.1 Organization deployment ............................................................................. 19
Software Quality Function Deployment
A method to build better software

2.3.1.2 Customer deployment ................................................................. 20
2.3.1.3 Voice of customer deployment .................................................... 20
2.3.1.4 Quality deployment ................................................................. 20
2.3.1.5 Function deployment ............................................................... 20
2.3.1.6 Reliability deployment ............................................................. 21
2.3.1.7 Task deployment ................................................................. 21

2.3.2 Blitz QFD .................................................................................. 21
2.3.2.1 Plan the process ....................................................................... 21
2.3.2.2 Go to gemba ........................................................................ 22
2.3.2.3 Sort the verbatims .................................................................. 22
2.3.2.4 Structure the customer needs ................................................. 22
2.3.2.5 Analyse the structured customer needs ..................................... 22
2.3.2.6 Prioritize customer needs ...................................................... 23
2.3.2.7 Deploy prioritized customer needs ........................................... 23
2.3.2.8 Deploy value throughout the project ........................................ 23
2.3.2.9 Apply, evolve and mature the process ...................................... 23

2.3.3 Distributed QFD ......................................................................... 24
2.3.3.1 Planning .................................................................................. 24
2.3.3.2 Overview meeting ................................................................. 24
2.3.3.3 DQFD sessions ....................................................................... 25
2.3.3.4 Post DQFD Work .................................................................... 25

2.3.4 VOC Analysis ........................................................................... 25
2.3.4.1 Define project success criteria ............................................... 26
2.3.4.2 Identify key market segments ............................................... 26
2.3.4.3 Go to gemba ................................................................. 26

Table of Contents
2.3.4.4 Analyse gemba data ..................................................................................... 27
2.3.5 Software QFD .................................................................................................. 27
2.4 Tools and techniques ................................................................................................ 28
  2.4.1 Affinity diagrams ............................................................................................. 28
  2.4.2 Hierarchy diagrams (trees) ............................................................................... 29
  2.4.3 Matrices and tables ........................................................................................... 30
    2.4.3.1 Customer segment table ............................................................................... 30
    2.4.3.2 Success criteria/customer segment matrix ................................................... 32
    2.4.3.3 Customer context table ............................................................................ , .... 34
    2.4.3.4 Customer voice table .................................................................................... 35
    2.4.3.5 Quality planning matrix ............................................................................... 37
    2.4.3.6 House of quality matrix ................................................................................ 38
  2.4.4 Analytic hierarchy process (matrix data analysis charts)............................................ 43
  2.4.5 Precedence diagrams, process decision charts and relationship diagrams...... 44
3.0 Theoretical framework ................................................................................................. 46
  3.1 Usage of QFD in software development .................................................................. 46
  3.2 QFD case study ........................................................................................................ 50
4.0 Developing a new methodology ................................................................................... 58
  4.1 Step 1: Planning the SQFD process [Planning] ........................................................ 59
    4.1.1 Selecting a facilitator ....................................................................................... 59
  4.2 Step 2: Identify true customers and sources of data [Customer deployment]....... 60
    4.2.1 Identifying customer segments ........................................................................ 60
    4.2.2 Deciding upon gemba visits ............................................................................ 62
  4.3 Step 3: Gathering data from the customers [Voice of customer deployment]........ 62
    4.3.1 Preparing for gemba visits ............................................................................... 62

Table of Contents
4.3.2 Running a gemba visit ................................................................. 64
4.3.3 Analysing the gemba data ............................................................. 65

4.4 Step 4: Building quality into the product [Quality deployment] .......... 67

4.4.1 SQFD and non-functional requirements ........................................ 67
4.4.2 SQFD and functional requirements ............................................... 70
4.4.3 SQFD and data requirements ....................................................... 74

4.5 Step 5: Modifying the development process [Function deployment] ...... 74

4.6 Step 6: Ensuring product reliability [Reliability deployment] ............... 76

4.7 Step 7: Identifying useful new technologies [New concept deployment] .. 78

4.8 Step 8: Assigning the Work [Task Deployment] .................................. 79

5.0 Conclusion ....................................................................................... 81

5.1 Summary ......................................................................................... 81
5.2 Recommendations ........................................................................... 81

6.0 References ....................................................................................... 83
Abstract

In recent years it is becoming increasingly more apparent that quality even more than productivity is emerging as the key issue in the development of software. The quality systems currently employed by most software companies however are simply not up to the task, traditional quality systems focus upon conformance to company standards, automation to eliminate human error and in some cases quality improvement teams. These traditional quality assurance methods lead to quality as defined from the organizations point of view, all work performed is done to their standards, however a what it is that makes a quality product is defined by the consumer. The companies quality standards, only serve to make it easier for the company to maintain the product at later dates, they in no way assure the end user that the product is fit for their purpose.

Quality Function Deployment is a step in the right direction, towards defining quality from the customer’s point of view. It is designed to ensure that the company takes into consideration their users needs, and helps with analysis of these stated needs to uncover any missing or unstated needs. Once the true listing of customer’s needs has been established, QFD helps the company to prioritize the listing from the customer’s perspective, enabling the product to meet all of their most important needs. The QFD (quality function deployment) process continues onwards throughout the entire software development lifecycle, providing a comprehensive method to ensure that the quality specified by the user is delivered to them throughout the developed product. The aim of this study is to examine the current trends, advancements and methods of various QFD systems and combine them into a QFD model specifically targeted at the software development environment.
Declaration

I certify that this thesis does not, to the best of my knowledge and belief:

(i) Incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;

(ii) Contain any material previously published or written by another person except where due reference is made in the text; or

(iii) Contain any defamatory material.

Signature: [Redacted]
1.0 Introduction

1.1 Definitions and acronyms

<table>
<thead>
<tr>
<th>Word</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWIH</td>
<td>A standard table layout used in many quality assurance methods. The table is organized with the headings Who, What, When, Where, Why and How.</td>
</tr>
<tr>
<td>AHP</td>
<td>Analytic Hierarchy Process</td>
</tr>
<tr>
<td></td>
<td>A matrix that is constructed to perform pair wise comparison of the elements contained within it. Provides both ratio and percentage importance weightings.</td>
</tr>
<tr>
<td>Demanded Quality</td>
<td>A unique singular expression of a customer requirement, given in their language.</td>
</tr>
<tr>
<td>Gemba</td>
<td>A term created by the Japanese practitioners of QFD, meaning the true source of information, the customers workplace or the area that the system will be used.</td>
</tr>
<tr>
<td>HoQ</td>
<td>House of Quality</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Lifecycle</td>
</tr>
<tr>
<td></td>
<td>The time spent developing a software product, from start to end including all phases in the chosen methodology.</td>
</tr>
<tr>
<td>QFD</td>
<td>Quality Function Deployment</td>
</tr>
<tr>
<td></td>
<td>A quality system designed to maximize customer satisfaction. Discussed in full later.</td>
</tr>
<tr>
<td>Verbatim</td>
<td>In terms of QFD, a verbatim is a product requirement given by the customer in their terminology.</td>
</tr>
</tbody>
</table>

Table 1: Definitions and Acronyms
1.2 Background information

1.2.1 What is quality?

In recent years it has become increasingly more apparent that quality is becoming a dominant aspect when measuring the value of a product. Many attempts have been made to define quality, each with varying degrees of accuracy, however over the years a solid definition has been formed. Traditional dictionary definitions are always of the type “degree of excellence” (Oxford Dictionary) or “fitness for use”. The ISO (International Standards Organization) made an attempt in the ISO standard 8402 to define quality as the “Ability to satisfy stated and implied needs” (ISO, 1995). In software development these stated and implied needs belong to the customer, and are in essence their requirements, the elements that they seek in their system, what makes the product valuable.

The meeting of the customer’s requirements yields a quality product, which provides customer satisfaction. It is this principle that QFD was founded upon, to help ensure that the product satisfies the customer. QFD helps to gather all of the customer’s requirements (stated, implied and exciting) and helps to map these outwards into the development process. This boosts requirements traceability and helps to ensure that the final product will be found valuable by the customer.
1.2.1.1 **What makes quality software?**

Software development as a process is more human intensive than other engineering disciplines, "*it requires mostly engineering rather than manufacturing*" (Ghezzi et al, 1991, pg. 18). Most other engineering disciplines require a lot of thought to be put into the manufacturing process to avoid the introduction of errors, however software requires that the final product simply be duplicated. Most of the effort and consideration is in the design and implementation phases of the SDLC (software development lifecycle). In all traditional engineering disciplines it is clear what the products required qualities are, in software development much work is still being done in this area, Ghezzi (Ghezzi et al, 1991, pg. 18-35) covers this area extensively a summary is found below in table 2.

<table>
<thead>
<tr>
<th>Quality Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>A software product is deemed ‘functionally correct’ if it behaves according to the requirements specification for the functions that it should provide. Correctness is an absolute measure, any deviation from the specification results in the software element being incorrect.</td>
</tr>
<tr>
<td>Reliability</td>
<td>A software product is deemed reliable if it consistently provides the correct results. Reliability is a relative measure, if the software element consistently provides the correct results it is deemed reliable.</td>
</tr>
</tbody>
</table>
Robustness  A software product is deemed robust if it continues to behave reasonably even under circumstances not anticipated in the requirements. Robustness is a relative measure, based upon the level of consistency and safety built into the product.

Performance  A software product is deemed efficient if it uses computer resources economically. Performance is important as it is directly related to the usability of the system, poor performance lead to user dissatisfaction.

User Friendliness  A software product is deemed user friendly if its human users find it simple to learn and easy to use, user friendliness is directly related to the level of experience of its human users. Novices may appreciate detailed error messages, whilst expert users may detest or ignore them. Standardization of human computer interfaces plays an important role in achieving user friendliness.

Verifiability  A software product is verifiable if the results of system properties can be measured easily. A common method of building in verifiability is to include ‘software monitors’ into the program, that is functions that can be accessed by the developers, which monitor the various qualitative aspects.

Maintainability  A measure of the ease in which the software can be extended, corrected or adapted at a later date.

Repairability  A software product is deemed easily repairable if corrections to defects can be applied with a limited amount of work. Software repairability is enhanced by the use of correct tools and modular parts.
**Evolvability**

Evolvability is a measurement of the ease by which changes can be applied to the system. The evolvability factor changes over time as each modification is made, and the conditions under which it is made. Careful planning must be performed before the change is made, determining the feasibility of the change, its impact upon the system etc. The change must also be adequately documented after the event, so that when making additional changes at a later date, the specification reflects the current state of the system.

**Reusability**

Reusability measures the modularity and ease of change to the systems components, and level of possible reuse achievable from these components. The reuse factor is a subjective amount, dependant upon the type of application being developed.

**Portability**

A measurement of the amount of machine or hardware specific code in the software product. The more machine dependant the product is, the less portable the system is made.

**Understandability**

The level of understandability is a measure of how easily the code can be read and correctly interpreted by a different developer. The level of understandability directly affects the level of maintainability.

**Interoperability**

A measure of the way in which the system will cooperate and coexist with additional systems. A measurement of the standardized programs interfaces, this can also be a measure of the programs support for data communication standards.
Productivity  A measure of the efficiency with which the software engineers developing the software product are working. Productivity is related to many trade-offs in the choice of process. The less of certain high-time low-impact tasks performed the more time the developers have to build the product.

Timeliness  A process related quality attribute, timeliness is the ability to deliver a software product to market on time. Similar to productivity some trade offs may be needed to achieve the desired level of timeliness.

Visibility  A measure of the level of understanding of the current status in the SDLC by all team members. For the process of the software product to be visible, it must be clearly documented so that every member understands the current status of the project.

Table 2: Software Quality Attributes (Ghezzi et al, 1991, pg. 18-35)

1.2.2 What is software development

Software development is the process through which a software product is developed. "Software is a logical rather than physical system element" (Pressman, 1996, p. 10). The majority of the work performed on a software product is engineering based, the design and crafting of a system based upon the user’s requirements. The manufacturing process is a simple process of duplication of the completed program.
"Software is developed or engineered, it is not manufactured in the traditional sense" (Pressman, 1996, p. 10). Pressman asserts that whilst there are some similarities between software development and hardware manufacture, they have many differences; both activities achieve high quality through good designs, but in a manufacturing defects can be introduced as a result of the manufacturing process, which are non-existent in, or easily fixed for software. Both activities are people oriented, but the relationship between effort applied and work performed are different. They both require the completion of a product but the approaches are fundamentally different, and most importantly, software (unless you are a large manufacturer) is not mass-produced upon the scale of traditional manufactured products.

With these differences in mind it becomes clear how the application of QFD, traditionally a manufacturing based quality method to software development environment could be difficult. Some work has been done in this area, however like software engineering, it is still very much in its infancy. Most efforts have attempted to develop either simple SQFD (software quality function deployment) models, based upon only 1 aspect (deployment) of QFD or to develop generalized QFD methods which can handle manufacturing, service industries or software development. The problems with the former is obvious, by applying a limited section of QFD to the development process, limited benefits are achieved. The later may seem good in theory, but the results have been either models that are too generic
or abstract to be of much use, or extensions to manufacturing or service models to help them better handle software development. This study aims to take the knowledge and experience that has been gained since QFD’s conception and combine with the techniques and tools included within recent variants of QFD, to produce a model designed specifically with software development in mind.

1.2.3 What is QFD?

"Quality Function Deployment (QFD) is a method for structured product planning that enables a development team to specify customer wants and needs clearly, and to evaluate each proposed product capability systematically, in terms of its impact on meeting those needs" (Cohen, 1993, p. 13). It is through this structure that QFD provides the company with a set of guidelines to follow that help maximize customer satisfaction. The focus is taken away from meeting organization standards (which may have nothing to do with satisfying the customer) and placed upon building value into the system. The value contained within the system will reflect itself in customer satisfaction and repeat business.

QFD traditionally was used as a manufacturing based quality method, its power was quickly realized and adapted to suit service industries. With all of QFD’s proven benefits, it becomes very appealing to develop a method that can be applied directly to a software development environment. Haag asserts that “Quality, even more than productivity of software is emerging as the key issue in the 1990’s” (Haag et al, 1996, p. 42) and any method that
can assure quality is highly valuable. QFD is the only comprehensive customer focused quality method available today.

QFD delivers this quality through its many different phases (deployments) that provide the company with data upon the customer’s requirements, from the perspective of each of the potential users of the system. Through the QFD process this data is compiled into a prioritized listing of the customer’s requirements, allowing the developers to focus upon meeting the most important requirements first, moving down the list as time and budget constraints allow.

QFD is an improvement over traditional quality systems, adding a distinctive customer focus, and providing facilities to better help identify overlooked or hidden requirements. The end result being a product that provides the functions most valuable to the customer. When comparing the differences between traditional and modern (QFD based) quality systems the advantages QFD has to offer become clear.

1.2.3.1 Traditional quality systems

“Traditional approaches to assuring quality often focus on work standards, automation to eliminate human error-prone process, and in more enlightened organizations, Quality Improvement Teams to empower employees to resolve problems” (Mazur, 1996, p. 1).

Traditional quality systems ensure that all work produced is consistent and up to the organizations quality standards, these quality
systems however in no way ensure that the work produced is of value to the customer.

"Consistency and Absence of problems are not enough of a competitive advantage after the market shakes out all of the sub-optimal players" (Mazur, 1996, p. 2). QFD is the only quality system aimed directly at providing customer satisfaction, for companies today, a competitive advantage must be sought in other methods, no longer is "Zero defects is not good enough" (Zultner, 1992, p. 29). Companies today must learn to understand their customer's wants, needs and thinking in an effort to provide higher value systems to them.

1.2.3.2 Modern quality systems

Traditional software quality systems are aimed at ensuring consistency and minimizing defects, that is minimizing negative quality. However the absence of these negative quality aspects does not add any positive quality (value) to the system. "Just because there is nothing wrong with the software does not mean there is anything right with it from the customer's perspective. It does not mean it has any value" (Zultner, 1992, p. 30).

QFD concentrates upon maximizing customer satisfaction with the product. "The focus is on preventing dissatisfaction by a deeper understanding of the customer's wants and needs and then deploying
these expectations downstream in order to design value into the system” (Zultner, 1992, p. 30). Through this method QFD offers the same advantages as traditional quality systems with the improvements of understanding what the customers consider a quality product, allowing the company to build value into their products. This process starts by attempting to gain an understanding of the customers, and their requirements.

1.2.3.3 The 3 types of requirements

To create value and provide customer satisfaction, it is required to have an understanding of their requirements, and an understanding of how meeting these requirements will effect the level of customer satisfaction. There are 3 types of customer requirements, listed below. By understanding the different types of requirements an understanding of how to improve levels of customer satisfaction with the product can be developed.

1.2.3.3.1 Revealed requirements

Revealed requirements represent the normal list of customer requirements, these are easily identified and revealed simply by asking the customer what it is that they want. The presence of these revealed requirements will satisfy or dissatisfy in proportion to their presence/absence.
1.2.3.3.2 **Expected requirements**

Expected requirements are often so basic to the customer that they neglect to mention them, until they fail to be delivered. A prime example of this may be on-line help facilities. Meeting these requirement goes unnoticed by all customers, failing to meet these requirements however will cause severe customer dissatisfaction. It is the responsibility of the analysis team to identify these requirements.

1.2.3.3.3 **Exciting requirements**

These are requirements that are beyond the expectation of customers, it is something they have not thought about or even considered, or something that is beyond their expectations. Their presence greatly adds value to the system, the absence of these features however goes unnoticed. It is the responsibility of the analysis team to explore possible areas of exciting requirements.

1.2.4 **History of QFD**

QFD was developed and introduced in Japan in 1966, by a team of quality experts including Dr Yoji Akao and the late Dr Shigeru Mizuno. It was developed and tested at Mitsubishi's Kobe shipyards to develop logistics for building large and complex super tankers. After QFD's first success Toyota applied it in 1977-1984 to help reduce the cost of producing vehicles. The resulting improvements reduced product launch costs by 61%, increased
annual profits by 50% and reduced product time to market by one third.

After these initial successes the Japanese continued to refine and develop QFD, it is now used widely across Japan.

QFD was introduced to North America in 1983 and since then it has began to spread throughout various industries including Ford, General Motors, Chrysler, Proctor and Gamble, General Electric and many other companies. The spread of QFD throughout the USA automobile industry was due mainly to the efforts of Ford, GM and Chrysler. At the time these three companies had been looking for ways to better improve supplier quality, collectively they developed there own derivative of the ISO 9001 quality standard, QS-9000 a quality system for service industries. A limited form of QFD was included in QS-9000 as a supplier activity. “It was not until 1984 that companies began to consider using QFD methods in software development” (Kliewer et al, 1998, p. 3).

1.2.5 Variants of QFD

Since its conception in Japan more than 30 years ago, the QFD method has continuously been researched and enhanced. With new methods, deployments and techniques continually being developed. The following techniques are recent developments designed with specific improvements to the QFD process in mind. Each of these techniques is discussed in detail in chapter 2.
Each of these variants has something to offer and they are drawn upon as a source of knowledge for this study.

1.3 Aims of the study

To research and analyse the current trends, tools and methods available in the traditional and advanced QFD methods, and apply them to a software engineering domain. This will be used to develop a variant QFD method aimed specifically at software development encompassing all the most advanced techniques in addition to the full power of the traditional QFD model. The methods and techniques of VOC Analysis and Blitz QFD will be incorporated into the model along with the current research upon SQFD to produce a robust, fast and effective SQFD technique.
2.0 Literature review

This section looks at the current state of research into Software QFD methods, Traditional QFD methods, and what can be learnt and applied to new Software QFD developments.

2.1 Limitation of current SQFD research

Current SQFD methodologies are still only in the “Kindergarten QFD” (Zultner, 1995, p. 25) stages of development. “In Japan, organizations may start with an HoQ matrix, but they continue to learn and master the rest of comprehensive QFD. Many US organizations succeed with the HoQ then stagnate at that low level for years.” (Zultner, 1995, p. 25), unfortunately, the majority of SQFD research is at this low point, and has been for quite some time.

The view that QFD is simply a HoQ (house of quality) matrix is a result of the way in which QFD was accepted in the western world. QFD was originally introduced through the QS-9000 standard as a supplier activity, however the method specified was a simplified version, limited to only one phase (deployment).

2.2 What can be learnt from other QFD areas

Traditional SQFD models were based upon the QS-9000 standards description of QFD (Quality Deployment only), which involves only the production of the HoQ diagram, without supporting activities this can lead to inaccurate requirements, ineffective customer analysis and other problems. These potential problems will continue to propagate downwards through the entire software development process.
By including all the deployments from the traditional QFD model into the SQFD system being developed the model should be less prone to errors, and more successful upon projects of any size, without previous QFD experience being essential. After the analysis and inclusion of these deployments, additional information can be learnt from 2 new techniques that have been recently developed. Blitz QFD (a fast-as-possible approach to QFD that doesn’t sacrifice quality by eliminating most unnecessary steps and replaces slower techniques with faster equally as valid techniques) and VOC (voice of the customer) Analysis (A new technique aimed at enhancing customer voice communication coming into the QFD project).

Each of these new methods has a lot to offer to the field of SQFD, many people who have tried the QFD technique complain about the time required, Blitz QFD is an attempt by Zultner to address this, he states that the most common problems with QFD are;

- The misconception that their organization performs QFD. “many people doing software QFD think that since they are doing a House of Quality they are doing QFD” (Zultner, 1995, p. 27). Which is clearly not the case, as Zultner asserts they are merely doing a matrix.

- The time required to perform the QFD process. Problems with time are largely due to incorrect and oversized HoQ matrices. Zultner puts this down to “Garbage in, Garbage out” (Zultner, 1995, p.28) the organization has misunderstood the contents of the matrix and how it
works, they have seen examples using ‘what’s’ and ‘how’s’ and
misunderstood. Zultner asserts that if they must label the matrix rows
and columns, then the terms ‘Criteria’ and ‘Solution’s’ are better labels.

- No new knowledge acquired from the process. Zultner states that “Weak
  content is a major reason the HoQ matrix lacks value” (Zultner, 1995, p.
  29) and he provides the most common reasons why this occurs;
  - No clear project goals.
  - No clear definition of customers, or which customers to satisfy.
  - Lack of customer observation, missing customer requirements, or
    a misunderstanding of the customer’s requirements.

Zultner’s Blitz QFD model attempts to address this problem by showing the general
lack of understanding and forethought before the commencement of a QFD project.
Zultner asserts that Blitz QFD was developed to help organizations get a better start
with QFD and allow them to proceed through the process with the minimum
number of steps taken.

VOC Analysis introduced by Glenn H. Mazur (Mazur, 1995, p. 1-9 & Appendix)
details a front-end method aimed at getting a complete list of customer needs in the
minimum amount of time possible. This front-end process improves upon
comprehensive QFD’s Customer and VOC Deployments, it offers information upon
the latest tools and techniques for these 2 deployments. This document is
advantageous any organization in that the information is presented in a simple,
precise fashion with included examples, allowing the data to be assimilated into the organization as quickly as possible.

2.3 QFD variants in detail

Since its conception some 30 years ago in Japan, there has been a large push to improve and refine the QFD technique. Comprehensive QFD is result of this refinement and continuous improvement, several other variants have also been produced, and each variant has its own specialization. In addition to Comprehensive QFD and Comprehensive QFD for service applications several other variant have been produced, including Blitz QFD, Distributed QFD, VOC Analysis and primitive forms of Software QFD. This section looks at each of these techniques in detail and discusses the tools and techniques involved for each step of the process.

2.3.1 Comprehensive QFD

Comprehensive QFD is the continual development of the original QFD system, it has been expanded to include many additional modules, Zultner (1665) and Mazur (1993, 1997) discuss this QFD model. Comprehensive QFD is a complete quality system working to improve quality, technology, cost and reliability of both the product and the methods that produce it. The comprehensive QFD model contains several individual deployment models, with each deployment addressing a different aspect of quality for the developing company. The comprehensive QFD model is detailed below
Comprehensive QFD allows the organization to tailor and improve upon the implemented deployments adding in others as they see needed. The Japanese implemented and have been using this form of QFD for over 30 years, the general misunderstandings with QFD stem from its introduction into the United States. QFD was introduced through the QS-9000, however only one component was actually introduced through this quality model, and that is Quality Deployment. This lead to a situation where "most American's in the auto industry and eventually most non-Japanese in nearly every industry failed to differentiate between QD and QFD" (Akao et al, 1998, p. 2). Each of the standard deployment areas is discussed below.

**2.3.1.1 Organization deployment**

This is used to map the QFD steps to different individuals throughout the organization, it shows who is responsible for what activities and when during the product planning and development process.
2.3.1.2 Customer deployment

This is the deployment of organizational goals (profit, utilization, market share, etc.) mapping them to defined customer segments (seniors, DINK’s, families, etc.) each defined by their individual customer attributes (income, impulse buying, marital status, children, etc.). This allows the organization to identify the customer segments that will contribute most to the success of the product.

2.3.1.3 Voice of customer deployment

This deployment is used to capture raw customer data, and classify it into sections (demanded quality, reliability, consistency, flexibility, etc.). These tables are also used to help uncover unspoken customer needs.

2.3.1.4 Quality deployment

This deployment maps customer’s demanded quality and priorities into measurable product quality characteristics. This section allows for several other items to be taken into consideration including: target measures, improvement ratings and competitor assessments.

2.3.1.5 Function deployment

This deployment is used to identify critical functional areas of the organization that are required to performing task that will achieve quality characteristic targets.
2.3.1.6  Reliability deployment

This is used to identify and prevent failures in meeting critical customer requirements.

2.3.1.7  Task deployment

This deployment is used to identify the tasks required for product completion, and help assign these tasks to organization resources.

2.3.2  Blitz QFD

Blitz QFD is a technique proposed by Zultner (1995) it was designed to address several problems and misconceptions of the QFD process in the USA. Blitz QFD is a streamlined variant of the QFD process developed to provide a greater chance for success with minimal work involved. Blitz QFD has no House of Quality matrix involved (which is a common misconception that QFD is just a HoQ Matrix), but still delivers a prioritized list of customer requirements.

Blitz QFD is broken down into 9 steps, only the first 7 however need to be implemented the last 2 are optional steps, the processes involved.

2.3.2.1  Plan the process

This step involves planning out the process of product development to establish what the organization’s goals and expectation are from this project.
2.3.2.2 Go to Gemba

During this step the developers send a team of people to the customers place of work (where the product will be used), to observe the customers process, problems and opportunities. These visits should be performed 12-15 times or until the organization feels that they have sufficient information to proceed.

2.3.2.3 Sort the verbatims

After completing the customer visits the organization has to sort out the information gathered breaking down statements into individual requirements and sorting these requirements into columns based upon the type of statement made (reliability, cost, functionality, technology, etc). The output from this step is the customer voice table.

2.3.2.4 Structure the customer needs

This step involved breaking down the customer's needs an affinity diagram to show the natural underlying structure.

2.3.2.5 Analyse the structured customer needs

This step converts the affinity diagram into a hierarchy tree to allow the organization to look for missing or overlooked customer requirements.
2.3.2.6 Prioritize customer needs

The aim of this step is to get a prioritized listing of customer requirements, this list is obtained through customer survey data. A large number (as large as possible) of customers are asked to show their preferences for the product requirements. The individual customer data is converted into an analytic hierarchy process (matrix), this allows us to see ratio priorities of each requirement through pair-wise comparison.

2.3.2.7 Deploy prioritized customer needs

This process involves correlating the prioritized list of requirements with the original customer needs data to see where relationships are formed, this creates a value table for the project, allowing us to see which areas being developed are of value to the customer.

2.3.2.8 Deploy value throughout the project

Developing a HoQ matrix for the project showing the linkages between customer needs and functional requirements of the product. The product developed will then be of maximum value to the customer.

2.3.2.9 Apply, evolve and mature the process

Successes and difficulties from this product development are then noted down so that the organization can continue to grow and improve at implementing the QFD process.
2.3.3 Distributed QFD

Kliweer et al (1998) discusses DQFD (distributed quality function deployment) in detail, the following is a summary of his analysis. DQFD is a technique defined and refined by Digital’s Corporate Telecommunications Software Engineering group. DQFD is simply a modification of the original QFD package to take advantage of geographically separated groups. DQFD makes heavy use of video conferencing (preferable due to the high costs of travelling around the world). This process is split up into 4 phases each having a component performed in different locations that come together during the overlapping time periods. One of the most promising benefits of this system is that more time can be spent working upon the QFD solution due to time differences.

2.3.3.1 Planning.

During this phase customer interaction allows the QFD teams to define and acquire as much customer data is possible. In addition to acquiring customer data, preparation must be made to ensure that both sites have the same materials available.

2.3.3.2 Overview meeting.

This is a preliminary meeting (usually conducted over video conferencing), members are introduced and roles are explained. A primary facilitator and primary customer are identified.
2.3.3.3 DQFD sessions.

The building of a HoQ Matrix is performed in a method adjusted to half days work, the sessions are tuned to accommodate the time differences between the geographical locations. Each time a session is performed without the other group being present the work is supplied to them as soon as it is completed.

2.3.3.4 Post DQFD Work.

During this phase work is assigned to the participants and additional resources needed are determined for the completion of the project.

2.3.4 VOC Analysis

VOC Analysis is a technique defined by Glenn Mazur (1997) in a series of papers, it can be summarized to the following. VOC Analysis is a compilation of the newest QFD tools into a method that is both fast and delivers the best possible results. The process is similar to Blitz QFD, in practice and delivered results. However VOC Analysis is more thorough in the analysis of identifying potential sources of requirements, VOC analysis also takes importance levels into account and several other advanced features. VOC Analysis produces a prioritized list of customer requirements based upon multiple requirement sources and can be used to provide input to a HoQ Matrix, or simply as inputs directly into any SDLC methodology.
The VOC Analysis follows through four steps, the methodology proposed later in this document for Software Quality Function Deployment will be partly based upon this method. The steps involved are as follows;

2.3.4.1 Define project success criteria

This process is used to align team members to the same set of goals. Organization goals are identified, categorized (using affinity diagrams) and hierarchy trees. This allows missing goals to be identified and helps determine selection criteria for which gemba(s) to visit.

2.3.4.2 Identify key market segments

Current and potential customer markets are identified, Customer segments are cross referenced with organization goals to identify the most promising customer markets for product deployment.

2.3.4.3 Go to gemba.

After identifying the most promising customer segments and the best gemba(s) to visit, the QFD team is deployed to analyse the customers process to look for problems and opportunities, to gather requirements and to examine the process itself to gain a better understanding.
2.3.4.4 Analyse gemba data

The data gathered from the gemba visit(s) is compiled and converted from customer verbatims into unique non-compound requirement statements. Statements are then sorted into their category based upon the type of requirement (functional, reliability, cost, etc), these requirements are then sorted using an affinity diagram to workout underlying structures and to help understand customer thinking better. The customer requirements are then prioritized using customer survey data to produce a quality planning table.

2.3.5 Software QFD

The current state of Software QFD variants available are all in the beginning stages of QFD advancement, in fact most of them simply involve drawing a HoQ matrix, which as Richard Zultner (Zultner, 1995, p. 25) puts it “QFD is not just a House of Quality matrix. That’s just doing a matrix. QFD is the comprehensive assurance of customer satisfaction through the development process – end to end”.

The current state of almost all Software based derivatives of QFD is simply drawing up this matrix, this is one of the primary limitations of the Software QFD based methodologies. SQFD methods do however have some advantages, the developers of these procedure have put a lot of thought into the structure of the matrix, the fields that need to be included and those which do not apply to a software based methodology.
2.4 Tools and techniques

Many new tools have been developed to improve software quality and help the quality assurance process, this section looks at the new tools developed specifically for QFD and the adapted traditional quality assurance tools. Comprehensive QFD employs the seven “new” management and planning tools to construct the individual deployments of the QFD quality assurance model. These tools were "developed to work on language data and relationships. These tools were specifically developed to be used by improvement teams outside of manufacturing areas" (Zultner, 1995, p. 27)

2.4.1 Affinity diagrams

Affinity Diagrams are designed to help surface the underlying structure of ideas, in QFD they are used to identify the thinking behind the customer’s requirements and help form them into natural groupings. Affinity Diagrams are very simple and fast being one of the easiest tools in the QFD set to perform. Affinity diagrams should be performed in small groups, working together without criticism.

1. Write each element being sorted onto a “Post-It” note.

2. Arrange all the elements silently into groups based upon shared ideas (affinity).

3. Group discussion for header cards to represent each group. Headers can be placed over multiple groups (grouping of groups is allowed, as the intention of the exercise is to develop a hierarchy).
2.4.2 Hierarchy diagrams (trees)

Hierarchy Diagrams allow all functional and non-functional elements of a project to be laid out in a tree-like fashion. In the QFD method, Hierarchy Diagrams are used to refine the information gathered from the affinity diagrams, add in overlapping of groups, and to help fine missing elements. Hierarchy Diagrams are generated immediately after the completion of an affinity diagram.

1. Layout the affinity diagram in order of abstraction from left to right (most abstract level to the left).

2. Adjust hierarchy nodes so that they represent the same abstraction at each level. Nodes at each level should be mutually exclusive.

3. For each node, review the children looking for any that may be missing, for each node the children should collectively represent an exhaustive set.
Matrices and Tables are used to deploy and communicate value and priority throughout the project. Matrices are used in QFD to explore the correlation between two project aspects, Tables are used to communicate target values, express priorities and document the details of processes and decisions. There are many different matrices and tables used throughout the QFD process the most common will be described here.

### 2.4.3.1 Customer segment table

This table is defined by Mazur (1997, Appendix p. 8), is designed to help identify all customer segments related to the project. The table works upon use and demographic data and allows the QFD team to quickly identify all customer segments and identify the most
important segments to target. The Customer Segment table is created as follows:

1. Create a table with the headings Who, What Why, Where, When and How (5W1H). Who will use the product, What they will use it for, Why they will use it, Where they will use it, When they will use it and How they will use it. These columns help to sort out the use and demographic data, more columns can be added as deemed necessary.

2. Fill every column with as much data as can be gathered upon specific groups of people, including market research, sales, percentages etc.

3. Circle together promising aspects of each customer and link then together in a chain to provide a customer segment profile. Try to identity as many customer segments this way.
2.4.3.2 Success criteria/customer segment matrix

This matrix is defined by Mazur (1997, Appendix p. 9), and is used to help evaluate the importance of each customer segment. The matrix compares the customer segments against the project success criteria, measures of impact are placed upon the level of impact the customer has upon each success criteria. This table is valuable to any large project facing multiple customers, the table helps the development organization decide which gemba to visit. This matrix is created immediately after the AHP (analytic hierarchy process) of project success criteria has been completed and the customer segments identified, it is created as follows:
1. Put hierarchy and weights from the AHP of the project success criteria into the rows of a standard relationship matrix. Place the customer profiles into the columns (Mazur recommends that “the top 10-15 most promising customer profiles” are used (Mazur, 1997, Appendix p. 9)).

2. Work through each of the rows, establishing the level of contribution each customer has to the project success criteria. Enter a value from 0-9, alternative graphical representations are also valid.

3. Multiply the AHP weights by the level of contribution for each cell and sum the products of these for each customer segment (column). Normalize the final values to a percentage.

4. Apply time, money, resources and gemba visits to the customer segments in proportion to the level of importance of each customer segment.
Software Quality Function Deployment
A method to build better software

2.4.3.3 Customer context table

Customer Context Tables were devised by Mazur and are defined in his article upon VOC Analysis (Mazur, 1997, Appendix p. 12). The table is based upon a simple 5W1H table with columns for verbatims and translations added. This table is used to record verbatims, translate them into the demanded qualities and to record additional environmental information from which these verbatims were generated. Customer Context Tables are constructed by through the following techniques:

1. Create a context sheet for each customer that is having information recorded. Record information about the customer
under the 5W1H columns, such as who they are, what there role in the organization is etc.

2. Capture all spoken and observed customer “verbatims” into the sheet for later translation and analysis.

3. Translate each verbatim into unique non-compound statements of customer requirements, convert all verbatims regardless of perceived difficulties or importance. These are dealt with later through the HoQ Matrix.

<table>
<thead>
<tr>
<th>Who</th>
<th>40 year old mail office worker</th>
</tr>
</thead>
<tbody>
<tr>
<td>What</td>
<td>Commute</td>
</tr>
<tr>
<td>When</td>
<td>Morning, evening</td>
</tr>
<tr>
<td>Where</td>
<td>High way</td>
</tr>
<tr>
<td>Why</td>
<td>Car pool</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verbatim</th>
<th>Translated Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>High performance, but sounds quiet</td>
<td>Accelerates quickly.</td>
</tr>
<tr>
<td></td>
<td>Good gas mileage.</td>
</tr>
<tr>
<td></td>
<td>Car is quiet.</td>
</tr>
<tr>
<td></td>
<td>Engine is quiet.</td>
</tr>
<tr>
<td></td>
<td>Absorbs sound.</td>
</tr>
<tr>
<td>Muffler doesn’t run out</td>
<td>Muffler doesn’t rust out.</td>
</tr>
<tr>
<td></td>
<td>Pipes don’t rust out.</td>
</tr>
<tr>
<td></td>
<td>Muffler is attached securely.</td>
</tr>
<tr>
<td>Starts easily when cold.</td>
<td>Starts easily when cold.</td>
</tr>
<tr>
<td></td>
<td>Starts easily when wet.</td>
</tr>
<tr>
<td></td>
<td>Can drive off immediately.</td>
</tr>
</tbody>
</table>

Figure 6: Customer Context Table Example (adapted from Mazur, 1997, Appendix p. 12)

2.4.3.4 Customer voice table

This table is used to analyse and sort the customer data from the gemba visits, the data is sorted into the grouping related to it. By utilizing this table, the QFD team can quickly establish what are the customers needs and what are statements about non-functional
aspects (cost, reliability, performance, etc.). The Customer Voice Table is constructed as follows:

1. Review each translated piece of gemba data generated from the above tools. Ensure that all translated statements are unique, non-compound statements.

2. If the data element being reviewed is a quality based expression of a customer benefit, place it into the demanded quality field.

3. If the data element being reviewed describes a measurable level of performance, reliability, availability, failure, a function, a solution or a methodology, place them in the appropriate column.

4. For each feature establish, search for other related demanded quality items that may have been overlooked.

<table>
<thead>
<tr>
<th>Demanded Quality</th>
<th>Performance</th>
<th>Function</th>
<th>Reliability</th>
</tr>
</thead>
</table>

Figure 7: Customer Voice Table
(Mazur, 1997, Appendix p. 12)
2.4.3.5 Quality planning matrix

The quality planning matrix is a reduced house of quality matrix, containing only the "right room" of the HoQ. This matrix is used in several variants of QFD to help to prioritize user requirements. This matrix can be used immediately before generating a HoQ matrix or in turn as a replacement on smaller projects. The quality planning matrix is constructed through the following steps.

1. Use modal survey data or data directly from the AHP to determine the rate of importance from each of the demanded qualities.

2. Take survey data upon customer views on the advantages and disadvantages of the competitors products.

3. Set improvement targets, sales points, percentage priorities etc.

<table>
<thead>
<tr>
<th>Operates Quietly</th>
<th>Rate of Importance</th>
<th>Quality Planning</th>
<th>Company Now</th>
<th>Competitor X</th>
<th>Competitor Y</th>
<th>Plan</th>
<th>Rate of Improvement</th>
<th>Sales Point</th>
<th>Absolute Weight</th>
<th>Demanded Weight (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car Operates Quietly</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>1.25</td>
<td>1.0</td>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Engine Operates Quietly</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1.67</td>
<td>1.2</td>
<td>6</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>Responds Quickly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8: Quality Planning Table
(Mazur, 1997, Appendix p. 13)
2.4.3.6 House of quality matrix

The HoQ matrix is the most well known matrix included in the QFD model. The house of quality is a large matrix, containing all of the information about what matters most to the customer. Current best QFD practice is to not develop a traditional QFD HoQ, but to instead “work on the rooms of the HoQ separately and simultaneously” (Zultner, 1995, p. 29). Development of the HoQ as a whole is said to be an “unwieldy and intimidating work object” (Zultner, 1995, p. 29). Where as the development of the individual rooms separately in individual matrices is faster and more focused. The steps to developing a HoQ differ from organization to organization, but the following are generally accepted steps.

1. Transfer importance values, and demanded quality attributes form the AHP to the HoQ matrix.

2. Derive a list of functions (system functions) that can be used to meet the demanded qualities.

3. Complete the matrix correlation table in the center of the HoQ

4. Take survey data from customers to gather data on the advantages and disadvantages of competitor products, importance ratings and features met.
5. Develop measurable targets for the performance of each functional requirement, and a method of measuring this performance.

6. Calculate importance weightings for each functional requirement of the proposed system, and allocate resources and budget spending to the more important aspects.
## Functional Requirements

### Contribution Level
- **•** = Extreme (9)
- **○** = Very Strong (7)
- **□** = Strong (5)
- **◦** = Moderate (3)
- **○** = Weak (1)
- **•** = None (0)
- **?** = Unknown (?)

### Demanded Quality

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>Numbers read accurately</td>
<td>12.0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>1.1.2</td>
<td>Complete orders</td>
<td>22.0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Correct Numbers on order</td>
<td>9.0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Correct customer info</td>
<td>14.0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Train associates quickly</td>
<td>15.0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Quicker answers</td>
<td>18.0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Faster credit check</td>
<td>10.0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

| Total | 19.3 | 28.6 | 20.1 | 23.9 | 33.7 | 37.0 | 16.4 |
| Priority (%) | 10.8 | 16.0 | 11.2 | 13.3 | 18.8 | 20.6 | 9.2  |
| Rank   | 6    | 3    | 5    | 4    | 2    | 1    | 7    |

### Technical Assessment

<table>
<thead>
<tr>
<th>Technical Assessment</th>
<th>Now</th>
<th>Competitor X</th>
<th>Competitor Y</th>
<th>Competitor Z</th>
<th>Plan (Target)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>40</td>
<td>25</td>
<td>70</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>70</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>P</td>
<td>F</td>
<td>F</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>215</td>
<td>181</td>
<td>76</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>

### Current Competitor Values

<table>
<thead>
<tr>
<th></th>
<th>Now</th>
<th>Competitor X</th>
<th>Competitor Y</th>
<th>Competitor Z</th>
<th>Plan (Target)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Priority (%)</td>
<td>1.25</td>
<td>1.5</td>
<td>1.67</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>Sales Point</td>
<td>15.0</td>
<td>41.3</td>
<td>35.0</td>
<td>18.0</td>
<td>25.0</td>
</tr>
<tr>
<td>Total</td>
<td>179.0</td>
<td>179.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance Measures</td>
<td>No. of detection's</td>
<td>% of Ns edited</td>
<td>Outside control limits (Pass/Fail)</td>
<td>N of logical relationships checked</td>
<td>N of basic ops simulated</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>----------------------------------</td>
<td>----------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Improvement Ratio</td>
<td>1.5</td>
<td>2.7</td>
<td>1.3</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Difficulty</td>
<td>1.5</td>
<td>1.2</td>
<td>1.0</td>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>Total</td>
<td>24.3</td>
<td>51.1</td>
<td>14.1</td>
<td>6.2</td>
<td>28.3</td>
</tr>
<tr>
<td>Priority (%)</td>
<td>12.2</td>
<td>25.8</td>
<td>7.1</td>
<td>3.1</td>
<td>14.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>Deployed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>✓</td>
</tr>
</tbody>
</table>

Figure 9: House of Quality
(Zaitner, 1995, p. 29)
This page has intentionally been left blank
2.4.4 Analytic hierarchy process (matrix data analysis charts)

Analytic Hierarchy Process is a more advanced method of prioritizing a list of elements. AHP uses pair wise comparison of the list to develop a ratio of importance for each individual element. QFD applies this concept to the prioritization of demanded quality items. AHP is more useful than standard questionnaires because it provides mathematical statistics defining exactly how much more important any element is when compared to another, instead of just a ranked list. To construct an AHP you must follow the following simple set of steps.

1. Create a matrix with the same data in both the rows and columns. This can be done for each node and it leaves immediately to the right.

2. Compare each pair of data of importance on a one to nine scale, with one meaning equally important and nine meaning extremely more important. The diagonal should be all ones, with the numbers below the line being the inverse of the numbers above.

3. A set of normalized columns is then created with their results summed, and normalized once again to yield a percentage of importance value.

4. On disagreements for values for a cell, a geometric average of their votes is entered into the matrix instead. This allows the process to yield accurate results even with team member disagreements.
Precedence diagrams, process decision charts and relationship diagrams

Precedence Diagrams, Process Decision Charts and Relationship diagrams are management tools that are applied to QFD in different implementations. Each of the tools is generally used to map out the customer’s process, providing a permanent record for later reference. Precedence Diagrams are the most commonly used of these techniques in QFD, they involve the construction of a network of arrows connecting geometric shapes. The most common use of precedence diagrams outside QFD is in PERT charts, inside QFD they are used to construct dataflow diagrams, State transition diagrams, Fault trees and simple flow charts. These techniques are all used to map out the customers process and express the manner in which they perform their functions. When documenting the customer’s process the organization should use one or more modeling methods that they are
comfortable with and that all members can readily understand, the following
should also be taken into account.

1. Visit the customer’s workplace (gemba) and discuss/observe the
customers work, make detailed notes upon the process and record
observations. Visit as many times as you feel necessary.

2. Map out the customer’s process using a simple arrow network.

3. Look for potential deviations, failures and improvements in the
customer’s process.

4. Uncover any implied customer needs.

5. Clarify any customer functions and sub systems that are used to perform
those functions. Propose new concepts that would be capable of better
performing those functions.
3.0 Theoretical framework

3.1 Usage of QFD in software development

To date there has been little empirical evidence made available regarding successful
SQFD projects, there is evidence however that a growing acceptance of the methods
and practices of SQFD in industry. The lack of available data is generally put down
to the nature of, and benefits of the topic, SQFD is a process improvement
technology, designed to provide a competitive advantage, this makes companies in
general unwilling to part with this information as that would risk their advantage.

Haag, Raja and Schkade (1996) recently did some work into gathering data upon the
level of industry acceptance of SQFD techniques. They interviewed 37 major
software vendors using a mixture of open-ended and closed question using both
telephone interviews and surveys. The data gathered is summarized below.

<table>
<thead>
<tr>
<th>Results Achieved</th>
<th>Mean Traditional Rating</th>
<th>Mean SQFD Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication satisfactory with technical personnel</td>
<td>3.7</td>
<td>4.09</td>
</tr>
<tr>
<td>Communication satisfactory with users</td>
<td>3.6</td>
<td>4.06</td>
</tr>
<tr>
<td>User requirements met</td>
<td>3.6</td>
<td>4.00</td>
</tr>
<tr>
<td>Communication satisfactory with management</td>
<td>3.4</td>
<td>3.88</td>
</tr>
<tr>
<td>Systems developed within budget</td>
<td>3.4</td>
<td>3.26</td>
</tr>
<tr>
<td>Systems easy to maintain</td>
<td>3.4</td>
<td>3.42</td>
</tr>
<tr>
<td>Systems developed on time</td>
<td>3.3</td>
<td>3.18</td>
</tr>
<tr>
<td>Systems relatively error-free</td>
<td>3.3</td>
<td>3.95</td>
</tr>
<tr>
<td>Systems easy to modify</td>
<td>3.3</td>
<td>3.58</td>
</tr>
<tr>
<td>Programming time reduced</td>
<td>3.2</td>
<td>3.70</td>
</tr>
<tr>
<td>Testing time reduced</td>
<td>3.0</td>
<td>3.29</td>
</tr>
<tr>
<td>Documentation consistent and complete</td>
<td>2.7</td>
<td>3.87</td>
</tr>
</tbody>
</table>

Table 3: Comparison of results achieved between traditional approaches and SQFD
(Haag et al, 1996, p. 46)

The data in the table above details the differences in the results achieved through
SQFD and traditional software development approaches. The data shows that the
only section that the only areas traditional development performed better was in
delivery time and project budget, this result most likely occurred through the development team(s) inexperience with the methodology, and its effects upon software estimates. The QFD process in general is very time consuming, far more so than traditional requirements engineering techniques and as such more costly, to perform (find a reference here), however it does offer far more benefits to the users. These benefits include improved satisfaction with the product across the board, a greatly improved level of user requirements satisfaction, an improved level of error control and greatly improved system documentation. The improvements in error reduction and documentation in turn lead into further cost reduction during future system maintenance.

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Mean Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured step-by-step methodology</td>
<td>5.00</td>
</tr>
<tr>
<td>Supports team involvement</td>
<td>4.80</td>
</tr>
<tr>
<td>Aids in avoiding the loss of information</td>
<td>4.60</td>
</tr>
<tr>
<td>Structured process for organizational communication</td>
<td>4.60</td>
</tr>
<tr>
<td>&quot;Preventive&quot; quality tool</td>
<td>4.60</td>
</tr>
<tr>
<td>Reduces departmental division</td>
<td>4.40</td>
</tr>
<tr>
<td>Leads to innovative responses to customer demands</td>
<td>4.20</td>
</tr>
<tr>
<td>Process to reduce complexity</td>
<td>4.00</td>
</tr>
<tr>
<td>Facilitates competitor analysis</td>
<td>4.00</td>
</tr>
<tr>
<td>Reduces design changes</td>
<td>3.80</td>
</tr>
<tr>
<td>Increases market share</td>
<td>3.80</td>
</tr>
<tr>
<td>Structured process for project documentation</td>
<td>3.80</td>
</tr>
<tr>
<td>As a knowledge repository</td>
<td>3.80</td>
</tr>
<tr>
<td>As a teaching tool</td>
<td>3.60</td>
</tr>
</tbody>
</table>

Table 4: QFD Manufacturing benefits realized in software development (Haag et al, 1996, p. 46)

The table above details the aspects of QFD that were found most valuable by companies when used in software projects. The companies surveyed were required to rank each benefit from 1 to 5 (strongly disagree to strongly agree), the most obvious conclusions that can be drawn from these results is that in terms of software development projects QFD is beneficial because it provides a structured
Software Quality Function Deployment
A method to build better software

methodology that supports team involvement. It also aids in avoiding the loss of information, provides a structured process for organizational communication and works as a preventative quality assurance tool (as opposed to a reactive tool).

<table>
<thead>
<tr>
<th>Factor by Rank</th>
<th>Mean SQFD Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Improved user involvement</td>
<td>4.60</td>
</tr>
<tr>
<td>2. Improved management support and involvement</td>
<td>4.40</td>
</tr>
<tr>
<td>3. Better trained user and management personnel</td>
<td>3.20</td>
</tr>
<tr>
<td>4. Technique to shorten SDLC</td>
<td>4.00</td>
</tr>
<tr>
<td>5. Methods which integrate techniques and tools</td>
<td>2.80</td>
</tr>
<tr>
<td>6. Better trained systems personnel</td>
<td>3.60</td>
</tr>
<tr>
<td>7. Increased use of automated tools</td>
<td>2.80</td>
</tr>
<tr>
<td>8. Improved project development technique</td>
<td>4.40</td>
</tr>
<tr>
<td>9. Improved cost/benefit analysis techniques</td>
<td>3.80</td>
</tr>
<tr>
<td>10. Improved computer hardware technology</td>
<td>3.60</td>
</tr>
</tbody>
</table>

Table 5: Impact of SQFD on factors necessary for developing improved computer-based information systems (Haag et al, 1996, p. 46)

The above table discusses QFD's impact upon software development, by looking at its impact upon the most important factors for the development of improved software based systems. Haag gathered the factors from the research of Necco, Gordon and Tsai's article "systems analysis and design current practices" from MIS Q Vol. 15, No 1 From December 1987. The survey results were again based from 1 to 5, the companies identified 4 major areas that QFD had a significant impact on: user involvement, management support and involvement, a technique to shorten the SDLC and improved project development technique. Haag cites that the list includes three of the four factors that Neeco described as the most important aspects from his study.

In addition to the tabulated results, Haag gathered data upon the determination of use of QFD in software projects. "80% of the organizations stated that the project
leader and project team determined whether SQFD will be utilized. In a limited number of cases, a management directive required the use of SQFD.” (Haag et al, 1996, p. 45). All organizations surveyed by Haag additionally cited QFD as a one of their best practice set of tools and management strongly encouraged its use. Haag additionally stated that two-thirds of the surveyed organizations had quality policies based upon TQM in place for 10 years and the rest of around 2 years, and all had these policies in place before the introduction of QFD. Haag asserts that “the implementation of QFD (SQFD in this case) can not be successful without the prior adoption of the TQM philosophy” (Haag et al, 1996, p. 45).

Through his survey Haag states that “the dominant purpose of SQFD utilization are analyzing customer demands, setting breakthrough targets, and analyzing competitors” (Haag et al, 1996, p. 46). Haag’s analysis of this data suggests that these tasks can be achieved through the usage of the first QFD matrix (a HoQ matrix). Haag states that the performance of QFD through this method is “consistent with how and to what extent the majority of the organizations utilize SQFD” (Haag et al, 1996, p. 46). This statement details the infancy of SQFD showing that most organizations have mastered the usage of a HoQ matrix but are yet to expand their knowledge and usage of SQFD. As discussed earlier there are several problems with only using a HoQ matrix, it ignores several of QFD’s major advantages, and in no way guarantees that the gathered information is correct or complete, in essence it fails to ensure quality for the customer.
3.2 QFD case study

Good QFD based software development case studies are difficult to find, in the western world, QFD is far from widely accepted and SQFD is far further behind that. There are however several good examples of traditional QFD in manufacturing and service industries, Glenn Mazur (Mazur, 1996), provides a comprehensive document covering the details from one of his successful QFD projects, for Host Marriott. Mazur worked as the QFD instructor for a project aimed at improving customer satisfaction with the products available at the Phoenix Sky Harbor International Airport. The team decided to target specifically the baked goods sold throughout the airport. The first step the team performed was to work and prioritize the project goals, this lead to the following table.

<table>
<thead>
<tr>
<th></th>
<th>CS</th>
<th>AS</th>
<th>LL</th>
<th>PI</th>
<th>WR</th>
<th>RAW Score</th>
<th>% of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Satisfaction (CS)</td>
<td>1.0</td>
<td>5.0</td>
<td>10.0</td>
<td>5.0</td>
<td>10.0</td>
<td>31.0</td>
<td>40.5%</td>
</tr>
<tr>
<td>Associate Satisfaction (AS)</td>
<td>0.2</td>
<td>1.0</td>
<td>5.0</td>
<td>5.0</td>
<td>10.0</td>
<td>21.2</td>
<td>27.7%</td>
</tr>
<tr>
<td>Landlord Satisfaction (LS)</td>
<td>0.1</td>
<td>0.2</td>
<td>1.0</td>
<td>0.2</td>
<td>5.0</td>
<td>6.5</td>
<td>8.5%</td>
</tr>
<tr>
<td>Profit Improvement (PI)</td>
<td>0.2</td>
<td>0.2</td>
<td>5.0</td>
<td>1.0</td>
<td>10.0</td>
<td>16.4</td>
<td>21.4%</td>
</tr>
<tr>
<td>Win &amp; Retain Contracts (WR)</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>1.0</td>
<td>1.5</td>
<td>2.0%</td>
</tr>
<tr>
<td>Totals</td>
<td>1.6</td>
<td>6.5</td>
<td>21.2</td>
<td>11.3</td>
<td>36.0</td>
<td>76.60</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Figure 11: Prioritized Project Goals (Mazur, 1996, p. 7)

After the project goals were finalized the team had to analyse the products available (within the baked goods grouping) and determine the impact each of these products has upon the individual project goals. From the resulting matrix (Figure 12) they determined that bagels were the best product, they had the largest effect upon all project goals. After finding this they examined the way in which the product was displayed (Figure 13) to find the best method of display, here they found that the large display cabinets proved to be the best.
Once the team had established the goals of the project they proceeded with identifying different customer segments. A partial example of this is shown in Figure 4 near the beginning of this document. Using this table they identified their key customer segment as core business travelers, once they had identified their strongest customer segment, they knew who to concentrate their surveys upon. The team was then lead by Mazur to the cafeteria (bagel gemba) to observe their customers in action. The data they observed here was recorded into customer voice tables, and later transferred into the rooms of the house of quality.

Mazur stated that through their observations, the customers wanted more choice of bagels and cream cheese, they noticed that the plastic utensils broke often, that the packaging on the cream cheeses was difficult to open. More importantly they noticed that the bagels were not cut or toasted, the company didn’t offer them in the way that they are most popularly eaten. Mazur asserts that the team “noticed that...
they were selling bagels in a way that speed of service (they wouldn't cut bagels or toast them which they thought could hold up the line), so they didn't offer the most popular ways bagels are eaten!” (Mazur, 1996, p. 9). Once the observations from the gemba visit had been recorded, a selection of customers was asked to prioritize the benefits and to compare the company’s bagels to those that they had eaten elsewhere. This process gave the QFD team data upon both customer preferences for improvements and information about the competition’s product. This process gave helped the team to develop a better focus, they now aimed to exceed the competition in the areas that the customers considered to be most important.

The QFD team then moved on to the production of a HoQ matrix, which they decided to approach at two levels. Firstly the analyzed the general categories of customer benefits, then extracting the most important benefits from this they compiled a more detailed second HoQ matrix focusing upon more detailed versions of these important categories. The team selected the quality attributes that were most important to the project and decided upon the levels of improvement that they wished to achieve in these areas, the performance in the other areas was to remain the same. This process gave them four distinct areas to improve; giving 50-60% of the display case to bagels, increasing the number of bagel varieties from 2 to 6, increasing the number of topping choices from 3-5 and adding the option to have your bagel toasted upon service.

Once the performance targets had been identified, the QFD team had to determine the activities that would need improvement and those that need to be maintained. As before they approached this with a two-step process, a generalized matrix to
identify business functions in need of improvement and then a matrix focusing upon
the most important demands and how they interact with the most important
functions, to help assure that these are met. Mazur also sought out information from
other companies involved in the manufacturers of bagels and cream cheese as to the
most popular varieties of each product. At this time Mazur also sought out
information regarding the toasting of the bagels, and located a company with a
toaster that could toast a bagel in the same time as it took to complete a sale.

After analysing the functions required and providing methods to meet some of these
through external suppliers. Mazur and his team set about ensuring that the system
was reliable by analyzing all of the possible failure modes that they could imagine
and examining their interaction with the demanded qualities, forming a reliability
matrix. The highest ranking failure points were to be closely examined during the
next phase, new concept deployment. During the new concept deployment, the
QFD team analyzed the failure points and provided alternative strategies to be
implemented when the problems became obvious. In addition to examining the
possible failure points, they also analyzed the new technologies available to them,
including bagel knives, display cases, heating elements, partially baked goods etc.

The team made their decisions based upon the most cost-effective reliable
technologies available to them. They chose to implement using partially baked
bagels that could be thawed quickly and baked in the kiosk in 6 minutes. They
selected the varieties of bagels and cream cheeses to be made available as well as
options for management to consider. After these selections were made the team
began to assign the tasks out using a standard task deployment method. The task
deployment matrix includes testing and training tasks it is designed to assure the
new process and to achieve the goals identified at the beginning of the QFD process

<table>
<thead>
<tr>
<th>What</th>
<th>Who</th>
<th>When</th>
<th>Where</th>
<th>How</th>
<th>How Much</th>
<th>Why</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cutting bagels</td>
<td>Mike Galvin</td>
<td>By Aug 28, 1995</td>
<td>In house</td>
<td>Test equipment</td>
<td>Until comfortable, but no less than 12 bagels</td>
<td>Specs for cutting method</td>
<td>Failure modes; safety, speed of service issues</td>
</tr>
<tr>
<td>Order bagel cutter</td>
<td>Joe Campbell</td>
<td>By Aug 25, 1995</td>
<td>In house</td>
<td>Purchase order</td>
<td>At least 3 for each testing unit plus one backup</td>
<td>For start of project in unit</td>
<td>Failure modes; proper knife, knife length</td>
</tr>
<tr>
<td>Scoops for topping portioning</td>
<td>Joe Campbell</td>
<td>By Aug 25, 1995</td>
<td>In house</td>
<td>Test order</td>
<td>6 of each for each test unit</td>
<td>Different types of toppings, portion control, speed of service</td>
<td>Possible equipment other than scoop, breakage</td>
</tr>
</tbody>
</table>

Figure 14: Excerpt from task deployment table for Phoenix bakery project (Mazur, 1996, p. 12)

Once the QFD team completed its analysis and assigned the tasks the procedure was converted into a set of standards, the standards that resulted from this QFD project were then adopted by all host phoenix catering venues where bagels were served. The results from this QFD project were sales that more than doubled as well as improved customer satisfaction, detailed below.
Software Quality Function Deployment
A method to build better software

**SERVED IN A FAST MANNER**

- Better: Pre: 20, Post: 40
- Same: Pre: 60, Post: 80
- Worse: Pre: 20, Post: 40

**WIDE VARIETY OF BAGELS**

- Better: Pre: 20, Post: 40
- Same: Pre: 40, Post: 60
- Worse: Pre: 20, Post: 40

**TASTE GOOD**

- Better: Pre: 20, Post: 40
- Same: Pre: 40, Post: 60
- Worse: Pre: 20, Post: 40
Figure 15: Improvements and Customer Benefits from QFD Bagel Project
(Mazur, 1996, p. 17)
The project took six people fifty three hours each (a total of 318 hours), Mazur attributes this length of time both to Host’s insistence upon working through the entire comprehensive service QFD and the lack of QFD experience within the team. Since this project host has gone on to use QFD for many other service related projects.
4.0 Developing a new methodology

This section deals with the development of a QFD based methodology specifically tailored to suit a software development environment. The methodology proposed here contains the full benefits of the traditional QFD set (comprehensive QFD), in addition to tools and techniques developed for both simple SQFD models and those from Blitz QFD and VOC Analysis. This section is divided into headings representing the steps in the SQFD process, a comparison between the developed SQFD and the traditional requirements elicitation process is shown below.

<table>
<thead>
<tr>
<th>SQFD Phase</th>
<th>Traditional Requirements Elicitation Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Planning</td>
<td>Performed in similar fashion, identifying the aims of the project.</td>
</tr>
<tr>
<td>Customer analysis</td>
<td>Not performed or performed informally.</td>
</tr>
<tr>
<td>Customer Deployment</td>
<td></td>
</tr>
<tr>
<td>Gather &amp; analyse the customer’s needs</td>
<td>Similar data gathered, usually performed in meetings or conferences. Minimal use of gemba visits. Generally no extra analysis upon underlying idea structures, leaving some exciting and unmentioned requirements overlooked. Generally no prioritization of customer needs, customer satisfaction with product suffers from larger variations.</td>
</tr>
<tr>
<td>Voice of Customer Deployment</td>
<td></td>
</tr>
<tr>
<td>Analysis of the customers demanded qualities</td>
<td>Performed by converting functional requirements into design statements, no guarantees that quality will be designed into the system, depends largely upon skill of designer.</td>
</tr>
<tr>
<td>Quality Deployment</td>
<td></td>
</tr>
<tr>
<td>Analyse required functions to meet quality attributes</td>
<td>Performed by analyzing non-functional requirements and looking for methods to help ensure that they are met.</td>
</tr>
<tr>
<td>Functional Deployment</td>
<td>Not performed, test functions are identified and performed during software testing, but generally no analysis of possible failures in the system or methods to correct them.</td>
</tr>
<tr>
<td>Reliability Deployment</td>
<td></td>
</tr>
<tr>
<td>Analysis of available technologies and benefits</td>
<td>Generally not performed or performed informally, dependant upon experience of developers and designers and the contact they had had with new technologies. Parts of this step are dependant upon reliability deployment, which is also generally not performed, another reason for this steps lack of performance.</td>
</tr>
</tbody>
</table>
Table 6: Comparison between SQFD Model and the traditional requirements elicitation process

4.1 Step 1: Planning the SQFD process

[Planning]

This is the initial phase of the QFD project, to decide exactly what the project is about and to establish which factors are critical to the success of the project (it may be additionally required to define what success means to this project). The defining of these criteria and the display of said criteria in a visible place helps to align the team members to the same goals, and helps the team to decide which gemba’s to visit. The success factors are easily defined through brainstorming within the team or with the customer, some customers will come along more prepared and have a list of what they want, others may not. Once an initial list of project success criteria has been established, they can be sorted using affinity diagrams (Figure 2) and analyzed for any missing elements that may effect the success of the project using hierarchy diagrams (Figure 3).

4.1.1 Selecting a facilitator

The most important role in any QFD project is the facilitator, it is their job to oversee the process and keep the team focused upon the goals of the project, stopping them from straying. When starting a QFD project, the facilitator should be chosen for experience in QFD and leadership skills, as it is important that they can keep the team motivated and focused. In
organizations new to QFD it is recommended that an external facilitator is brought in to the project, whilst a member of the staff is trained underneath them. If an external facilitator can not be found, it is recommended that an individual with experience in both project management and quality assurance is used, as they will most likely possess the required skills.

4.2 Step 2: Identify true customers and sources of data

[Customer deployment]

4.2.1 Identifying customer segments

Once the criteria directly responsible for project success have been identified, the next step is to analyse the groups of people who will be using the developed system in addition to what, when, where, why and how they will be using the system. The easiest method to achieve this goal is with a customer segment diagram (Figure 4). From a software development it is important to take into account the future maintenance personnel as customers in addition to the day to day users of the system. Although system maintenance will (hopefully) not be performed regularly, it is an important function and steps must be made to ensure the maintainability of the system (covered later).

This document shows an example of a standard 5W1H customer segment table, however there are more recent advancements, specifically by Glenn Mazur (Mazur, 1997, p. 6), in the field of data storage for comparisons in QFD. Mazur provides a template and suggests the usage of his 5W2H3C1F
matrix instead, which expands upon the 5W1H by adding columns for; how much, cost, control, checks and failure modes. Some of these extra columns may be useful for customer segment analysis, such as how much or how often the system will be used, control how much access do they need or should have to the system, checks what kinds of checking or auditing need be performed, and failures or frustrations they may have with the existing systems. All data relevant to identifying customers should be recorded, specifically how often they will use it but other aspects may be useful as well. The general guidelines for 5W2H3C1F tables are included below.

<table>
<thead>
<tr>
<th>5W2H3C1F</th>
<th>Current</th>
<th>New</th>
<th>Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who</td>
<td>Is / should be using or doing it?</td>
<td>Else could / should be doing it?</td>
<td>Should not be using or doing it?</td>
</tr>
<tr>
<td>What</td>
<td>Is / should be used or done?</td>
<td>Else could be used or done?</td>
<td>Should not be used or done?</td>
</tr>
<tr>
<td>When</td>
<td>Is it / should it be used or done?</td>
<td>Else could it be used or done?</td>
<td>Should it not be used or done?</td>
</tr>
<tr>
<td>Where</td>
<td>Is it / should it be used or done?</td>
<td>Else could it be used or done?</td>
<td>Should it not be used or done?</td>
</tr>
<tr>
<td>Why</td>
<td>Is it / should it be used or done?</td>
<td>Else could it be used or done?</td>
<td>Should it not be used or done?</td>
</tr>
<tr>
<td>How</td>
<td>Is it / should it be used or done?</td>
<td>Else could it be used or done?</td>
<td>Should it not be used or done?</td>
</tr>
<tr>
<td>How Much</td>
<td>Is / should it be used or done?</td>
<td>Else could be used or done?</td>
<td>Should not be used or done?</td>
</tr>
<tr>
<td>(What) Cost</td>
<td>Is / should be expended?</td>
<td>(What other) costs could be expended?</td>
<td>Should not be expended?</td>
</tr>
<tr>
<td>(What) Control</td>
<td>Measurements are / should be monitored?</td>
<td>Other measurements could be monitored?</td>
<td>Measurements should not be monitored?</td>
</tr>
</tbody>
</table>
4.2.2 Deciding upon gemba visits

Once the customer segments have been identified, the QFD team must prepare a matrix (Figure 5), comparing the customer segments to the project success criteria. This matrix allows the QFD team to see the impact each customer segment has upon each success criteria, in effect to work out what are the most important group or groups of customers. Gemba visits should then be divided up between the groups in order of importance.

4.3 Step 3: Gathering data from the customers

[Voice of customer deployment]

4.3.1 Preparing for gemba visits

Once the target gemba’s have been identified, the QFD facilitator must now help the team to establish their roles and responsibilities during the gemba visits. The roles that are required for a gemba team are as follows:

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facilitator</td>
<td>• Assist the team in defining the problem&lt;br&gt;• Stop the team from drifting</td>
</tr>
<tr>
<td>Interviewer</td>
<td>• Identify and interview customers&lt;br&gt;• Gather requirements through questioning</td>
</tr>
<tr>
<td>Observer</td>
<td>• Identify and observe customers</td>
</tr>
</tbody>
</table>

Table 7: General definitions of 5W2H3C1F
(Mazur, 1997, p. 6)
Software Quality Function Deployment
A method to build better software  
Dean Carruthers

(may also be interviewer)

• Gather requirements through observing the customers process

Recorder
(may also be interviewer)

• Identify and record customers
• Gather requirements through recording customers statements, problems and achievements

Lead talker

• Communicate with employer, management
• Maintain a single source of information for management to communicate with and/or question for information.

Customer

• Perform daily tasks
• Allow QFD team to observe them at work
• Point out failures in current system and possible improvements from their perspective

Table 8: Roles & Responsibilities required for a gemba team

The gemba team does not necessarily require all of these roles, all of these roles (with the exception of customer) are no independent, combinations can be assigned to one individual, Interviewers generally are also play the roles of the two other customer analysts Observer and Recorder. It is recommended however that with an inexperienced team that these roles are separated, because each offers unique input that may be overlooked if the roles are combined, the individual may concentrate upon one and neglect the other. The customer role appears in the table because in some organizations it is possible for the gemba team to be assigned a set of employees to work with, instead of being given free reign over the organization.

As a general exception to the above rule, the roles of observer and recorder are generally well suited to being combined together, as they are both passive customer analysis roles.

After the team has identified the roles that all members will play during the gemba visits, it is important for them to decide upon which employees in the
customers organization that they wish to visit, interview and observe. The best choices for people are those who know the most about the process, unfortunately these people are generally the busiest. It is important that the customer’s organization understands the benefits of the QFD project, and allows the team unrestricted access to the individuals that they select.

After selecting the individuals inside the customer’s organization to analyse, it is important for the team to select the equipment that they will require and to become familiar with it. Possible equipment choices include; tape recorders, video cameras, pens, paper, anything that will help to capture what the customer wants. And it is important that the team members know how to use it before hand, so that they do not waste time or miss valuable information. Once all of this has been planned, if the team is new to QFD it is a good idea if they go through a practice run, upon each other or employees of their company who are not going to the gemba.

4.3.2 Running a gemba visit

When running gemba visits the goal is to gather as much information with as little disturbance to the organization as quickly as possible. Try to book interviews consecutively, record everything that happens, pay close attention to the way the processes intended for automation are currently being performed, watch for problems or possible enhancements. The creation of state transition diagrams and process flow diagrams may also be helpful to the team (depending upon their experience with these techniques). The purpose of the gemba visits is to gather enough information so that you can
model the customer’s process and not have to return for any additional information. As Mazur states, you “walk a mile in your customer’s shoes to understand how he does business, what his customers need, and what problems he has satisfying their needs” (Mazur, 1996, Appendix p. 10)

The number of gemba visits required varies with project complexity and team experience, however it is widely agreed that around 10 should suffice for most projects. At the end of those visits the team will have collected almost all of the information that they could have, and have enough of an understanding about the process to identify any that they missed.

4.3.3 Analysing the gemba data

Once the data has been collected, on tape, hand written, video interviews, state diagrams, flow charts, etc the QFD team needs to concentrate its efforts on turning these unstructured statements into structured unique expressions of customer requirements. This is generally achieved using a combined customer context and verbatim translation table. Mazur (Mazur, 1996, Appendix p. 12) details this technique however the layout he suggests is inefficient it can be better represented using a table for each customer with there context data stored above it. This allows for quick references and comparisons, between departments, jobs etc. The preferred layout is detailed in figure 6. Additional data can be stored upon these sheets including their customer segment, the importance ratio of this segment and any measurements that they give upon the verbatims (e.g. very important). An enhanced layout is detailed below.
Once the CCVT tables have been filled in for each customer, the next step is to organize the translated data into different categories using a customer voice table, based upon what the data is describing. If the data is a qualitative expression of customer benefit, then it is placed under “demanded quality” if the data describes a measurable level of performance, reliability, data storage, etc it is placed under the heading for the appropriate quality attribute. For every column, the QFD team should look at the data contained within them and search for any elements that may be missing, based upon their experiences during the gemba visits. An example of a customer voice table can be found in figure 7 of this document.

After customer data has been sorted into the appropriate types, they remaining demanded qualities can be sorted further, by using affinity diagrams the SQFD team should group qualities based upon shared affinity...
Another use of affinity diagrams, particularly in large projects is to split the demanded qualities up into modules (or even sub-modules) of related functions. Hierarchy diagrams (trees) can then be employed to search for additional missing data elements. After the list is deemed complete by the QFD team, prioritization of the demanded qualities should follow immediately, achieved in one of two ways; through analytic hierarchy process or through performing customer surveys.

4.4 Step 4: Building quality into the product

[Quality deployment]

Software engineering is different to most other engineering disciplines, we receive requirements for the system we receive many requirements for the system, functional, non-functional and data requirements all play an important part in describing the design of a system. This software based methodology of QFD therefor takes all three of these into account during the quality deployment phase.

4.4.1 SQFD and non-functional requirements

In a software system, there are certain quality constraints placed upon a system, if it meets these constraints it is deemed to be a valuable system. Many of these constraints are not based purely on the system functionality, some are based upon and aspect that has nothing to do with the process it can perform, these constraints are not based upon function, they represent the other aspects of the system. Non functional requirements tend to be system wide and are usually given in open ended statements by the
customer, it is up to the developers to ensure that they are met, otherwise a perfectly functional product could be deemed useless.

In this model for SQFD, non-functional requirements are handled separately from data and functional since they relate to the system as a whole. These requirements are handled through a comparison of the customers needs (previously identified using the customer voice table) with the listing of quality attributes (table 2). All elements in the customer voice table that are not labeled demanded quality or data fit into the description of non-functional requirements. These are then compared using a quality attributes matrix to match up the customer's needs with the quality attributes. Through this process the SQFD team is able to measure the performance of the identified non-functional requirements against the quality attributes.

The table lay out is similar to the quality planning table (discussed earlier under 2.4.3.5), the remaining customer needs (not demanded quality or data) are laid out down the vertical axis, and the quality attributes (table 2) are placed along the horizontal. The level of contribution is marked using either numbers or symbols (although the symbols in the diagram are recommended as they provide a quick graphical representation) and totals provided, along with the totals and percentage of importance values for each quality attribute. If the customer has specified that certain quality attributes are more important (for example: the SQFD team is building a real-time system, so reliability is a must), then the matrix can be altered to include target values for the quality attributes. This helps to force additional functions
being developed and built in to ensure the appropriate level of quality is built into the product.

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Wt</th>
<th>Correctness</th>
<th>Reliability</th>
<th>Robustness</th>
<th>Performance</th>
<th>User Friendliness</th>
<th>Verifiability</th>
<th>Total</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal system downtime</td>
<td>5</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>Quick response time</td>
<td>4</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>76</td>
<td>2</td>
</tr>
<tr>
<td>Correct Responses to calculations</td>
<td>4</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>76</td>
<td>2</td>
</tr>
<tr>
<td>Comprehensive error messages</td>
<td>5</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>●●●●●●●</td>
<td>120</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Testing Measures</th>
<th>Walkthrough Results</th>
<th>System Tests</th>
<th>Invalid data tests</th>
<th>Algorithm Tests</th>
<th>User testing</th>
<th>Priority(%)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td>9</td>
<td>30</td>
<td>9</td>
<td>9</td>
<td>15</td>
<td>2</td>
<td>74</td>
</tr>
<tr>
<td>Priority(%)</td>
<td>12</td>
<td>41</td>
<td>12</td>
<td>12</td>
<td>20</td>
<td>3</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 17: Example Demanded Qualities Vs Quality Attributes Matrix
(note: the quality attributes list was cut short due to space constraints)

The SQFD team may be tempted to make additional modification so that this table can handle the inclusion of competitor analysis, sales points and improvement ratios, however these are better left a later stage. After the quality attributes matrix has been completed, the SQFD team now understands the quality attributes that the customer values most. The SQFD team now can expand upon the customers' non-functional needs, by examining the relations that they play with each quality attribute we can create additional non-functional needs for use in a contribution matrix for the
Software Quality Function Deployment
A method to build better software

non-functional requirements. An example of this break down, based upon the above quality attributes matrix is included below.

<table>
<thead>
<tr>
<th>General non-functional</th>
<th>Specified non-functional derivatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal system down time</td>
<td>• Minimal downtime for maintenance. [Reliability]</td>
</tr>
<tr>
<td></td>
<td>• Minimal downtime for backups. [Reliability]</td>
</tr>
<tr>
<td></td>
<td>• Minimal downtime due to unhandled errors. [Robustness]</td>
</tr>
<tr>
<td>Quick Response Time</td>
<td>• Consistent response time for similar operations. [Reliability]</td>
</tr>
<tr>
<td></td>
<td>• All functions return control rapidly to the user (under 10 seconds). [Performance]</td>
</tr>
<tr>
<td></td>
<td>• All functions inform the user if there is to be a delay. [User Friendliness]</td>
</tr>
<tr>
<td>Correct responses to calculations</td>
<td>• All functions mathematically correct. [Correctness]</td>
</tr>
<tr>
<td></td>
<td>• All functions provide consistent correct output. [Reliability]</td>
</tr>
<tr>
<td></td>
<td>• Confirmation asked before proceeding with ambiguous or questionable data. [Verifiability]</td>
</tr>
<tr>
<td>Comprehensive error messages</td>
<td>• Consistent error messages. [Reliability]</td>
</tr>
<tr>
<td></td>
<td>• Minimal (if not no) fatal errors. [Robustness]</td>
</tr>
<tr>
<td></td>
<td>• Informative text error messages. [User Friendliness]</td>
</tr>
<tr>
<td></td>
<td>• Confirmation asked before proceeding with ambiguous or questionable data. [Verifiability]</td>
</tr>
</tbody>
</table>

Table 9: Example breakdown of Non-functional requirements

4.4.2 SQFD and functional requirements

The functional requirements in SQFD are represented by the demanded qualities section of the customer voice table. If the data was split into modules (or sub modules) using affinity diagrams, then this process should be performed individually for each module. On a large software project, without modularization the matrices produced during this phase will grow to become quite large making it "an unwieldy and intimidating work object" (Zultner, 1995, p. 29). If the SQFD team skipped the affinity diagrams then they can perform an additional matrix here before continuing onwards. The
use of a pair-wise demanded quality comparison matrix will allow the data
elements to be split into modules of related customer needs. This matrix can
also be performed if the team wishes to break down their identified modules
further or to ensure that their modules are correct.

![Relationship Strength](image)

<table>
<thead>
<tr>
<th>Relationship Strength</th>
<th>View WBS</th>
<th>View Assigned Resources</th>
<th>View Gantt Chart</th>
<th>View Pert Chart</th>
<th>View Assigned Tasks</th>
<th>Automated Task Timers</th>
</tr>
</thead>
<tbody>
<tr>
<td>View WBS</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>View Assigned Resources</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>View Gantt Chart</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>View Pert Chart</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>View Assigned Tasks</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Automated Task Timers</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

**Figure 18: Example pair-wise comparison matrix for demanded qualities**

A pair-wise comparison matrix usually forms the top room or ‘roof’ of the
house of quality, most authors argue that it is unnecessary for SQFD,
however it still functions in SQFD as it was designed to in QFD to identify
related functions and show possible clashes in requirements. It is important
to note however that the pair-wise step is only required if the team for some
reason skipped the affinity diagram steps earlier.

Once the SQFD team has a modularized list of the demanded qualities, the
next step is to build a quality planning matrix. The quality planning matrix
makes up the right hand room of the house of quality. The matrix store data
regarding current business values, competitor values, target values,
improvement ratios and sales points. An example quality planning matrix can be found at figure 8 of this document.

Once the quality planning matrix has been completed, the SQFD team will have a prioritized listing of results (by percentage) and data regarding their products placement in the marketplace. The next step is to complete a contribution matrix, it is performed in the same fashion as the non-functional contribution matrix, but it refers to the technical requirements and how they contribute to meeting the demanded qualities. Technical requirements in SQFD represent the proposed system functions, the contribution matrix shows how each system function contributes to meeting the customers demanded system qualities. The matrix also outputs a priority for each function, showing which functions will produce the greatest level of customer satisfaction.

<table>
<thead>
<tr>
<th>Contribution</th>
<th>View WBS</th>
<th>View Gantt Chart</th>
<th>View Pert Chart</th>
<th>View Assigned Tasks</th>
<th>View Allocated Resources</th>
<th>Allocate Resources</th>
<th>Task Leveling</th>
</tr>
</thead>
<tbody>
<tr>
<td>● = 9</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>○ = 5</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>□ = 3</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>• = 0</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Management</th>
<th>Task Management</th>
<th>Resource Management</th>
<th>Total</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>869</td>
<td>11</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>144</td>
<td>99</td>
<td>99</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>100</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>114</td>
<td>114</td>
<td>114</td>
<td>114</td>
</tr>
</tbody>
</table>

Figure 19: Example contribution matrix for the functional quality requirements

Once the contribution matrix has been complete the next step for the SQFD team is to complete the competitive analysis matrix. This helps the team to
place forethought into how they are going to test each system function, as well as compare their functions to how the current system (if any) has them implemented and to compare to competitors standards. The matrix also helps to re-prioritize the functions based upon improvement levels and the difficulty of implementation.

<table>
<thead>
<tr>
<th></th>
<th>Priority (%)</th>
<th>Now</th>
<th>Competitor X</th>
<th>Competitor Y</th>
<th>Target</th>
<th>Improvement Ratio</th>
<th>Difficulty</th>
<th>Total</th>
<th>Priority</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>View WBS</td>
<td>21.0</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>N</td>
</tr>
<tr>
<td>View Assigned Resources</td>
<td>12.2</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>N</td>
</tr>
<tr>
<td>View Gantt Chart</td>
<td>15.8</td>
<td>F</td>
<td>F</td>
<td>P</td>
<td>P</td>
<td>1.0</td>
<td>1.2</td>
<td>18.9</td>
<td>31.0</td>
<td>2</td>
</tr>
<tr>
<td>View Pert Chart</td>
<td>13.2</td>
<td>F</td>
<td>F</td>
<td>P</td>
<td>P</td>
<td>1.0</td>
<td>2.0</td>
<td>26.4</td>
<td>43.0</td>
<td>1</td>
</tr>
<tr>
<td>View Assigned Tasks</td>
<td>20.8</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>N</td>
</tr>
<tr>
<td>Automated Task Timers</td>
<td>16.0</td>
<td>F</td>
<td>P</td>
<td>F</td>
<td>P</td>
<td>1.0</td>
<td>1.0</td>
<td>16.0</td>
<td>26.0</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 20: Example competitive analysis matrix for functional requirements

The above example demonstrates a simple competitive analysis matrix, the matrix can be extended to include as many competitors as the SQFD team feels is necessary, and also performance or test measures for each functional requirement. Values for the company now, competitors and targets should be given either as a numeric value (representing how many are offered) or as a modified boolean value P/F (Pass or Fail). The difficulty value is obtained from technical analyst’s recommendations, it is a value from 0 to 2 with 1 being the normal difficulty for a small function. The total is obtained by
multiplying the difficulty by the improvement ratio by the original priority.

These values are then normalized to produce the modified priorities.

4.4.3 SQFD and data requirements

The handling of data requirements in traditional QFD is non-existent, since data is more or less unique to the software-engineering domain. The handling of data requirements in this SQFD model is performed using a combination of data dictionaries and traditional QFD matrices. The SQFD team should construct a contribution matrix showing all of the users data (and data related demanded qualities) on the vertical axis and the proposed data storage structures/methods along the top. A standard data dictionary should be created to describe the data structures that are detailed along the vertical axis.

Through this method the SQFD team are forced to think ahead in terms of how they will store their data, and what they need to store, it minimizes the need for alterations, changes and rework at later dates. In addition to this it provides a comprehensive data dictionary for later use by the developers, that is both consistent with the design of the system and complete.

4.5 Step 5: Modifying the development process

[Function deployment]

Function deployment in traditional QFD is the process of examining manufacturing or business functions that need to be changed to reflect the proposed quality improvements. In SQFD function deployment only needs to be performed for the
non-functional requirements, since these are the only requirements that will effect the business practice of the development company. Function deployment is designed to help the development company ensure that the non-functional requirements are successfully integrated into the system.

Functional deployment is performed like most other matrices in SQFD by using a contribution matrix, between desired non-functional qualities on the vertical and new or modified business functions along the horizontal. Through this process the SQFD team can propose new quality measure and model their impact upon the desired non-functional qualities. As with the demanded non-functional qualities contribution matrix the data is grouped under the quality attributes (for the same reasons).

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Testing for complex algorithms</th>
<th>QA on error messages</th>
<th>User testing before system release</th>
<th>Automated Scheduling for Backups</th>
<th>Fully documented procedures</th>
<th>Total</th>
<th>Business Function WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimal downtime for maintenance</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10.0</td>
<td>20.4</td>
<td></td>
</tr>
<tr>
<td>Minimal downtime for backups</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7.0</td>
<td>14.3</td>
<td></td>
</tr>
<tr>
<td>Consistent response time for similar operations</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8.0</td>
<td>16.3</td>
<td></td>
</tr>
<tr>
<td>All functions provide consistent correct output</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12.0</td>
<td>24.5</td>
<td></td>
</tr>
<tr>
<td>Consistent error messages</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12.0</td>
<td>24.5</td>
<td></td>
</tr>
</tbody>
</table>

Figure 21: Example mapping business functions required to achieve quality attributes
4.6 Step 6: Ensuring product reliability

[Reliability deployment]

The process of reliability deployment aims to identify all (or as many as) possible system failure points, reliability deployment aims to seek out problems that could be encountered with the design and fix them before they occur. The SQFD team needs to brainstorm all the possible problems that could affect the system, from the more simple incorrect input types to more complex errors corrupt files, date compatibility with the year 2000, database problems, memory leaks etc. Once the problems are identified, an estimate of their impact upon the system needs to be made (from 0 little or no impact to 5 critical) Once these problems have been identified, a reliability matrix is constructed. The reliability matrix is used to show the occurrences of the identified problems in each function, i.e. what aspect of the system are susceptible to that problem.

<table>
<thead>
<tr>
<th>Failure Mode</th>
<th>Impact</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect data entry</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Incorrect data storage</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Year 2000 date incompatibility</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Corrupt file structure loaded into program</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Example table of possible system failures
non-functional requirements, since these are the only requirements that will effect the business practice of the development company. Function deployment is designed to help the development company ensure that the non-functional requirements are successfully integrated into the system.

Functional deployment is performed like most other matrices in SQFD by using a contribution matrix, between desired non-functional qualities on the vertical and new or modified business functions along the horizontal. Through this process the SQFD team can propose new quality measure and model their impact upon the desired non-functional qualities. As with the demanded non-functional qualities contribution matrix the data is grouped under the quality attributes (for the same reasons).

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Testing for complex algorithms</th>
<th>QA on error messages</th>
<th>User testing before system release</th>
<th>Automated Scheduling for Backups</th>
<th>Fully documented procedures</th>
<th>Total</th>
<th>Business Function WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>⬤ = 9</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>10.0</td>
<td>20.4</td>
<td></td>
</tr>
<tr>
<td>⬤ = 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⬤ = 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⬤ = 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⬤ = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>? = ?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 21: Example mapping business functions required to achieve quality attributes
4.6 Step 6: Ensuring product reliability

[Reliability deployment]

The process of reliability deployment aims to identify all (or as many as) possible system failure points, reliability deployment aims to seek out problems that could be encountered with the design and fix them before they occur. The SQFD team needs to brainstorm all the possible problems that could effect the system, from the more simple incorrect input types to more complex errors corrupt files, date compatibility with the year 2000, database problems, memory leaks etc. Once the problems are identified, an estimate of their impact upon the system needs to be made (from 0 little or no impact to 5 critical) Once these problems have been identified, a reliability matrix is constructed. The reliability matrix is used to show the occurrences of the identified problems in each function, i.e. what aspect of the system are susceptible to that problem.

<table>
<thead>
<tr>
<th>Failure Mode</th>
<th>Impact</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect data entry</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Incorrect data storage</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Year 2000 date incompatibility</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Corrupt file structure loaded into</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>program</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Example table of possible system failures
Software Quality Function Deployment
A method to build better software

Contribution

Reliability

Minimal downtime for maintenance 20.4
Minimal downtime for backups 14.3
Consistent response time for similar operations 16.3
All functions provide consistent correct output 24.5
Consistent error messages 24.5

<table>
<thead>
<tr>
<th>Reliability</th>
<th>Business Process Wt</th>
<th>Incorrect Data Inputs</th>
<th>System Crashes</th>
<th>Backups Damaged</th>
<th>Flawed Code</th>
<th>Incorrect Algorithms</th>
<th>Incorrect QA procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal downtime for maintenance</td>
<td>20.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Minimal downtime for backups</td>
<td>14.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Consistent response time for similar operations</td>
<td>16.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>All functions provide consistent correct output</td>
<td>24.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Consistent error messages</td>
<td>24.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

| Absolute Wt | 1253 | 220.5 | 226.5 | 128.7 | 236.6 | 220.2 | 220.5 |
| Fail point Wt | 100.0 | 17.6 | 18.1 | 10.3 | 18.8 | 17.6 | 17.6 |

Figure 22: Example reliability matrix (performed upon reliability requirements)

Once the reliability matrix has been completed the SQFD team will know what problems each area of the system is susceptible to, with this knowledge they can plan measures to reduce the errors. The SQFD team should concentrate its efforts upon reducing the occurrence of the errors with the biggest negative impact upon the project. Make note of the solutions inside the table of problems that is encountered, if the solutions are problem specific made additional notes regarding the problem that they relate to.
4.7 Step 7: Identifying useful new technologies

[New concept deployment]

In the traditional forms of QFD, new concept deployment is used to analyse the new process, look at the identified fail point and seek out possible alternatives, new suppliers, backup precautions. In addition to this it is also used to examine the new technologies available to the companies industry, and consider implementing the solution using these more advanced technologies. Whilst providing alternatives to possible failures is impossible in a software environment, the SQFD team can analyse new technologies in the area for the product.

During this phase the SQFD team, takes a look at the designed process, and the fail points, looking for areas where further optimization is possible. They also look to larger resources, trying to find solutions, enhancements and new techniques that can be used to achieve the task at hand quicker. This includes the use of case tools, code generators, etc, new technologies that can be incorporated to in some way enhance either the development environment or the final product. There is no formal process for this step, but it is an important step that should not be overlooked, it can potentially offer many advantages to the project. The process of new concept deployment should be performed to the individuals taste, if no obvious improvements exist, move on to deploying the tasks.
4.8 Step 8: Assigning the Work

[Task Deployment]

Task deployment is the final phase of the SQFD process, at the completion of this phase the tasks that will deploy the functions will be assigned to personnel. "The best laid plans come to fruition when individuals are made responsible for carrying out the specific tasks in a manner that achieves the targets that were designed and planned in the previous steps" (Mazur, 1997, p. 13). Task deployment in SQFD is handled much the same as it is handled in traditional QFD, there is no specific guideline for the identification of tasks, simply that each technical requirement, business function and data requirement must some how be translated into the final system. In addition to this all testing and performance measures should also manifest themselves as tasks during this phase.

Once the tasks have been identified, a contribution matrix should be constructed to ensure that each technical requirement, data requirement, business function requirement, testing/performance measure are met in some way by the tasks performed. Once it is established that all system requirements are met through one of the identified tasks, the tasks should be assigned out to the members of the development team. To record the process and help with planning a task deployment table should be constructed, the method of planning dates, constraints etc is left up to the individual project manager.
<table>
<thead>
<tr>
<th>What</th>
<th>Who</th>
<th>When</th>
<th>Where</th>
<th>How</th>
<th>How Much</th>
<th>Why</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding System Interface</td>
<td>John Smith</td>
<td>By Aug 25, 1995</td>
<td>In house</td>
<td>Using GUI builder</td>
<td>3 Hours</td>
<td>To use as a base to develop other modules</td>
<td></td>
</tr>
<tr>
<td>Coding New Project</td>
<td>Joe Brown</td>
<td>By Aug 30, 1995</td>
<td>In house</td>
<td>Coding in VB</td>
<td>1 Hour</td>
<td>To create new entries in the project database</td>
<td></td>
</tr>
<tr>
<td>Coding Save Project</td>
<td>Joe Brown</td>
<td>By Aug 30, 1995</td>
<td>In house</td>
<td>Coding in VB</td>
<td>1 Hour</td>
<td>To save data from new projects to the database</td>
<td>Failure modes; corrupt data files</td>
</tr>
</tbody>
</table>

Figure 23: Example task deployment table
5.0 Conclusion

5.1 Summary

As shown throughout the paper there is strong evidence that the QFD approach works in the design of goods and services. The evidence detailed also shows how QFD can be used to boost product quality and to increase customer satisfaction. This technique is appropriate to the domain of software development due to the strong reliance that software has upon its design and engineering, a quality design generally leads to a successful product. QFD can be successful in helping to reduce both the development time and overall cost of software production, due to its forward thinking and ability to help the team get the requirements right the first time.

The SQFD method that is described within combines the benefits of the traditional comprehensive QFD method with the advancements made in blitz QFD and Voice of Customer analysis. In addition to these advantages it is built with software development specifically in mind, taking into account most of today’s development strategies to provide a comprehensive software QFD model.

5.2 Recommendations

It is recommended that the method proposed in this study undergo continuous testing and refinement. New techniques in the fields of QFD and SQFD continue to be developed, it is recommended that these techniques are reviewed, and if found acceptable implemented. The methodology suggested herein is currently untested,
and it is hoped that in the near future this method will be trailed upon several projects.
6.0 References

(1) Tran, Tuyet-Lan & Sherif, Joseph S. "Quality Function Deployment (QFD): An effective technique for Requirements Acquisition". Jet Propulsion Laboratory, Software Assurance California institute of technology Pasadena.

(2) Tran, Tuyet-Lan. "QFD Application to a Software-Intensive Systems Development Project". Jet Propulsion Laboratory, Software Assurance California institute of technology Pasadena.


(7) Reich, Yoram. "AI-Supported Quality Function Deployment". Proceedings of the fourth international workshop on Artificial Intelligence in economics & management.


   “http://www.cpsc.ucalgary.ca/~johnson/SENG/SENG613/Project/report.htm”


   “http://cern.ucalgary.ca/~dweening/SENG613/QFDReport.html”


Software Quality Function Deployment
A method to build better software


