



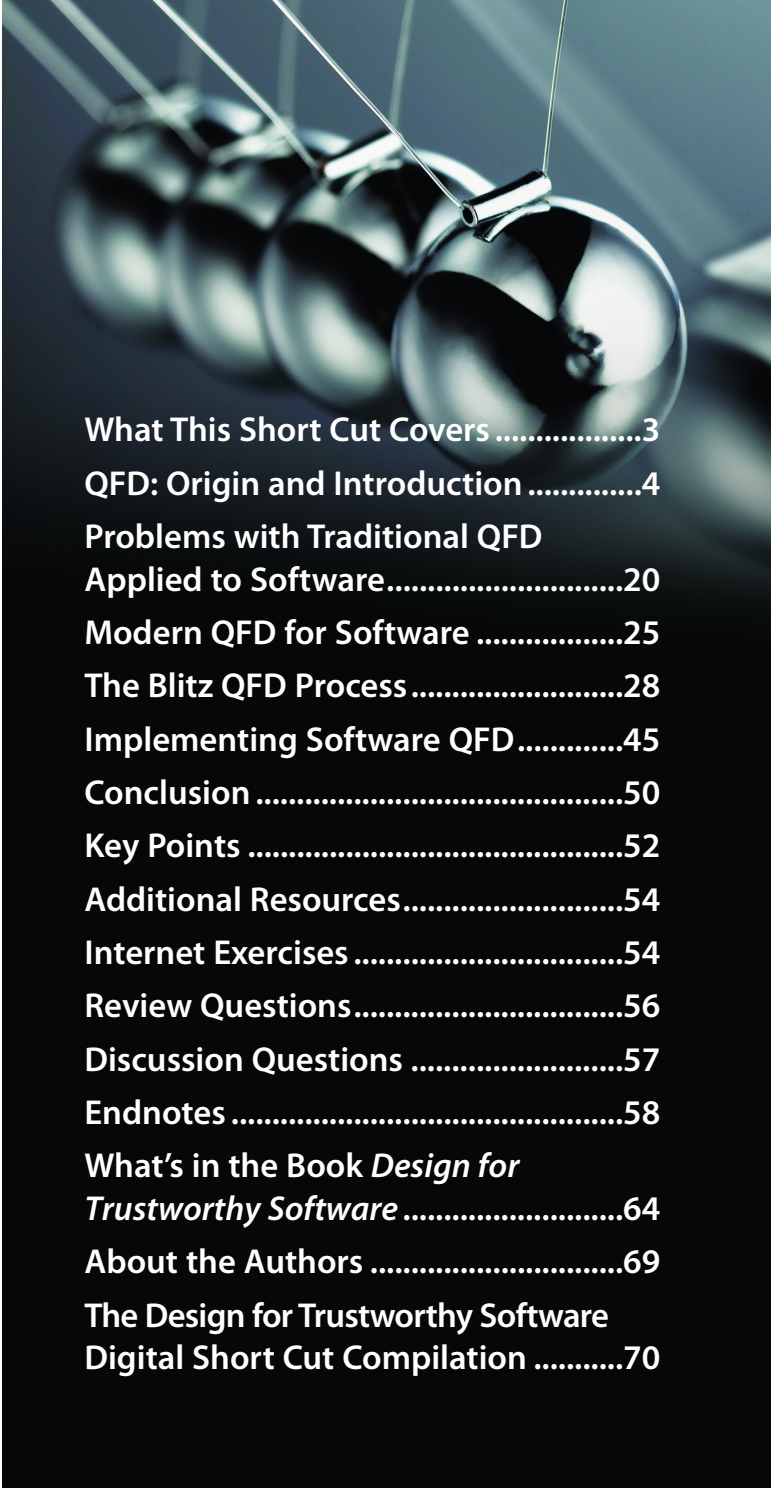
The Design for Trustworthy Software Compilation

Understanding Customer Needs


Software QFD and the Voice of the Customer

Bijay K. Jayaswal
Peter C. Patton
with Richard E. Zultner

This article is an adaptation of Chapter 11 from the book *Design for Trustworthy Software* by Bijay K. Jayaswal and Peter C. Patton (0-13-187250-8, Prentice Hall).



What This Short Cut Covers	3
QFD: Origin and Introduction	4
Problems with Traditional QFD Applied to Software.....	20
Modern QFD for Software	25
The Blitz QFD Process	28
Implementing Software QFD	45
Conclusion	50
Key Points	52
Additional Resources.....	54
Internet Exercises	54
Review Questions.....	56
Discussion Questions	57
Endnotes	58
What's in the Book <i>Design for Trustworthy Software</i>	64
About the Authors	69
The Design for Trustworthy Software Digital Short Cut Compilation	70



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this work, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this work, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Visit us on the Web: www.prenhallprofessional.com

Copyright © 2007 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
One Lake Street
Upper Saddle River, NJ 07458
United States of America
Fax: (201) 236-3290

ISBN 0-13: 978-13-235134-8

ISBN 0-10: 0-13-235134-X

First release, March 2007

SECTION 3

Problems with Traditional QFD Applied to Software

any answer to a *how* in the columns. This leads to the situation where newcomers make avoidable mistakes.

This is a special problem for software, as the terms *what* and *how* have been widely used in methods as far back as SA to explain the distinction between logical or essential (technology-and-implementation-free) models done in analysis and appearing in a functional spec, and physical or design (technology-and-implementation-specific) models done in design and appearing in a design spec.

That QFD continues to flourish despite these challenges is perhaps the best testimony to the method's robustness. Even poor QFD is significantly better than no QFD, so QFD continues to grow and spread anyway.

Criteria versus Solutions

If you have *hows*, or design solutions, in the columns, you will likely put the criteria for choosing the solutions in the rows. For technical people these are usually **technical requirements** or **design requirements**—the characteristics or capabilities that the design/technology/implementation must satisfy. So we wind up with what should be the columns in the rows, and the customer needs never appear. This is a very common situation, which you will find in the

well-known car-door case study. Although this is a useful matrix, it is a design matrix and not a “House of Quality” matrix, produced to bridge the world of the customer and the world of the developer. (And it may not be the most important matrix for you to do, if for some reason you are doing only one.)

“It Takes Too Long”

One of the most common complaints of applying traditional QFD to software is that it can take a great deal of time. Worse, once people begin the detailed analysis that any QFD matrix requires, they often lose sight of the reason QFD exists: to ensure customer satisfaction *efficiently*.

In the next section, we'll see how Modern QFD addresses the purpose of QFD but without using the matrix as the primary tool to do so and thereby avoiding this problem. The QFD framework will still aid us in keeping our focus on the “voice of the customer” and on cross-functional team communication concerning what matters most to customers, and it will act as an aid in discovering high-value customer needs and corresponding product requirements and features.

SECTION 3

Problems with Traditional QFD Applied to Software

To address this problem, some practitioners of traditional QFD tried to “speed up” or “streamline” traditional QFD by doing the following:

- ▶ Doing only the “House of Quality” matrix
- ▶ Doing a size-limited “House of Quality” matrix, with a restricted number of rows and columns, or by using the secondary level of the customer needs hierarchy instead of the tertiary level
- ▶ Focusing on only portions of the product where critical trade-offs occur when doing a model upgrade
- ▶ Terminating the QFD process when it has provided “enough” answers to critical questions

Unfortunately, such approaches did not succeed and are not pursued today. Instead, by going back to the fundamentals of QFD and applying QFD to itself, a more efficient form of QFD was developed, whereby the matrix is an optional tool and is used only when specifically required. Another benefit of this analysis was to clearly show the power of QFD for new-product development, including products that are new to the company and are new to the world.

“We Knew That Already”

A common complaint concerning Software QFD is that the results were not of sufficient value—in other words, they weren’t worth the effort required to produce them. As many people are still only doing the “House of Quality” matrix for their software development projects, this means that the “House of Quality” matrix required too much effort and yielded too few insights. The common complaint is “We knew that already. The answers were obvious.”

Weak Content

A major cause of lack of value in the “House of Quality” matrix is weak content. Several things cause content to be weak, and the most common is not having a clear understanding of the project’s strategy—in other words, not having a clear idea of the project’s success factors and how the project fits with the organization’s overall strategy. This can lead to irrelevant QFD results. The second cause of weak content is not defining what types of customers the project has to satisfy and how much it has to satisfy them—that is, which customers are most important to the project’s success and why. This leads to the wrong customers being visited, a greater proportion of

SECTION 3

Problems with Traditional QFD Applied to Software

customer needs from the least important customers, and a lack of important needs from the most important customers. The final cause of weak content is not having a strong upstream “voice of the customer” process prior to doing the “House of Quality” matrix—that is, not being clear, trained, or skilled at observing the context of the customers, gathering their statements, and organizing and prioritizing their needs. This leads to important needs being lost or misunderstood by the time they get to the “House of Quality.” Before you invest time and effort in doing a “House of Quality” matrix, make sure the inputs are correct.

Even if the content was correct, however, there were other problems. ...

Weak Understanding

If QFD was a newly developed technique, we could understand why certain errors are so common. But despite the fact that QFD is a mature method, an almost random pattern of learning and dissemination of bits and pieces of QFD occurred in the first 15 years of QFD’s spread to North America and Europe.

The problem is further aggravated by QFD’s role as a key component in both TQM³³ and DFSS.³⁴ Because

too few people in the Six Sigma community have acquired a comprehensive understanding of QFD, the biggest benefits to using QFD are still ahead for most organizations that use DFSS.

Filter the Input

The “House of Quality” matrix should have *only* customer needs in the rows and *only* the related software characteristics and capabilities in the columns. Nothing else should be in this particular matrix. To ensure this, you should sort out the customer statements using a Customer Voice Table (CVT), organize them using an Affinity Diagram, structure them using a Hierarchy Diagram, and prioritize them using the AHP (we will discuss all of these issues shortly). After this upstream processing is done, you will have good input to your “House of Quality” matrix.

This will prevent the problem of mixed items in your matrix, but what about the sheer size of the matrix. What if we have lots of customer needs?

Control the Size

There are several aspects to making sure you are not overwhelmed in the QFD process, and the first aspect is to focus on your most important customers by

SECTION 4

Modern QFD for Software

prioritizing your strategy and your customers and then having the *customers* prioritize their needs. This allows you to concentrate on the top n customer needs. The second aspect is to plan what matrices you should do (if any) and limit their size. There is no point in working through a matrix with many more customer needs than you can act on with best efforts. Take the top n customer needs and choose n to be within what the team can tackle. Finally, do not exhaust all of your QFD efforts in one shot at one point in the project. It is far better to do ten 10x10 matrices (or other QFD tools) to address ten key dimensions during your project at ten carefully chosen points in time than it is to do one 100x100 “House of Quality” matrix at one point.

If at any time, you feel that your matrix (or any QFD tool) is “too big,” it means you have not managed the size of your work objects properly. In software engineering, we have to do *complete* work objects. You have to put all the objects in your system on your class hierarchy diagram, not just the most important ones. But in Software QFD, we are focused on value, so working on the top n of anything is not only allowed, but also is essential for focus and efficiency.

Check the Matrix

Every step in the QFD process involves procedures for checking the step, catching errors, and ensuring correctness. (This is a basic property of any good quality method.) QFD matrices can contain a variety of structural errors, and methods for checking for such errors do exist, as do methods for conducting simple mathematical checks on the values in the matrix. Learn how to do the work right and how to check that you have done so.

Modern QFD for Software

In response to feedback from both software and nonsoftware organizations using QFD in the early 1990s, a group of experienced practitioners at the QFD Institute³⁵ (QFDI) applied QFD to QFD to develop a better QFD product.³⁶ What resulted was Blitz QFD: a faster, better way to do QFD, especially for those just starting to use the method. This is an engineered initial subset of QFD designed to provide maximum gains from a minimum of effort. In practice, Blitz QFD turned out to be the perfect way to get the benefits of QFD on very rapid development projects and any time that time or resources were tightly limited. The aim is to deliver essential value to key customers rapidly and efficiently.

SECTION 4

Modern QFD for Software

Blitz QFD

To do the least QFD and get the most gains, we must focus on QFD fundamentals and apply them to software development.

- ▶ **Exploration.** What are the most important things for us to know about satisfying our customers' needs? We need requirements exploration to answer this.
- ▶ **Execution.** What are the most important things for us to do to deliver value to customers? We need requirements deployment to plan this.

So what does Blitz QFD have that allows us to answer these questions quickly during software development?

The Seven Management and Planning (7 MP) Tools

The “official” tools for QFD are the Seven Management and Planning (7 MP) Tools,³⁷ as shown in Figure 9. While you are not limited to these tools, you should be familiar with them to take full advantage of Modern QFD.³⁸

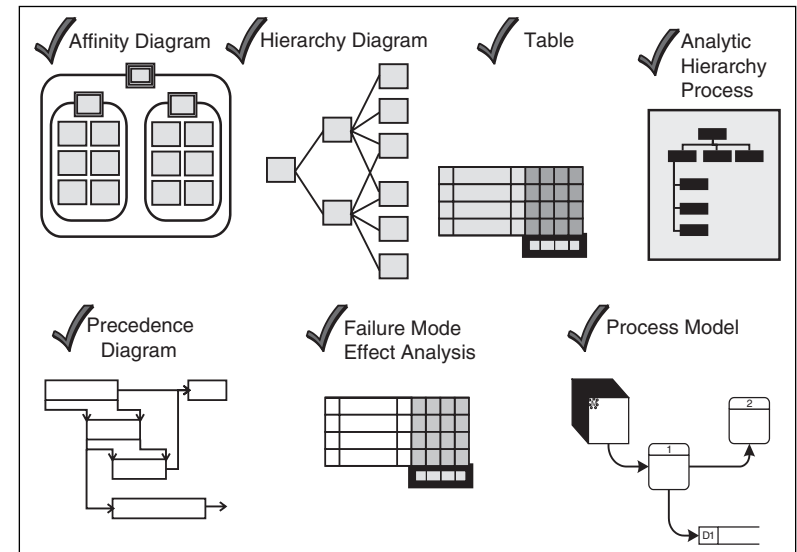


FIGURE 9 The 7 MP Tools for Blitz QFD. The basic tools for traditional QFD are modified for Blitz QFD. Gathered from a variety of disciplines, these tools support rapid requirements exploration and deployment. Note that matrices, which are for detailed analysis, are not used.

Customer Satisfaction and Value

Customers are satisfied when they receive value: benefits greater than the effort or cost to them of achieving those benefits.³⁹ Customers determine the amount of benefits and costs and, therefore, the value they received.

SECTION 4

Modern QFD for Software

What Is Value?

In QFD, **value** is recognized when customers see some benefit: a *problem* that has been solved or minimized; an *opportunity* to seize, maximize, or enable; *looking good* to significant others; or *feeling good* about themselves.⁴⁰ If a software package does nothing for any of your problems, does not enable you to seize any of your opportunities, does not help you to impress others whose opinion you care about, or does not help you feel better, it is of no value to you.

Even a single customer will desire multiple benefits (problems, opportunities, and look good and feel good issues) with different magnitudes and priorities. These are **customer needs** in QFD terms. A group of customers with similar perceptions of value (benefits and costs) and similar judgments about their magnitudes and priorities is a **customer segment**.

How Is Value Delivered to Customers?

Out of the large number of software work objects (objects, data, functions, scripts, and so on) produced during development, only a few directly or strongly relate to *any* high-priority customer need.⁴¹ This does not suggest that unnecessary items are

worked on during development. All the items are there for a reason. And if none of them was delivered, the customer would certainly not be satisfied. But many “necessary” items have no significant effect individually on the amount of value a customer receives. Whether they are done superbly well—or are absent entirely—individually they do not comprise the difference between customer satisfaction and customer dissatisfaction. In examining ten recently developed Information Technology (IT) development projects in a large retail organization, such “necessary” but nonessential (non-high-value-added) items averaged 73 percent of the items delivered.⁴² Doing a great job on items that don’t matter to customers is inefficient.

What Is Essential to Our Success?

A small number of items individually have a *very* significant effect on the amount of value a customer receives. Done well, they can deliver sufficient value to ensure customer satisfaction. Such high-value items are *essential* to satisfying the customer. In a large retailer study, the number of essential items in the ten systems studied ranged from 3 to 17.⁴³ If we could identify these *essential items* upfront, we could concentrate our best software engineering attention on them. Doing a