The following is an excerpt from a Short Cut published by one of the Pearson Education imprints.

Short Cuts are short, concise, PDF documents designed specifically for busy technical professionals like you.

We've provided this excerpt to help you review the product before you purchase. Please note, the hyperlinks contained within this excerpt have been deactivated.

Tap into learning—NOW!
Visit www.informit.com/shortcuts for a complete list of Short Cuts.
QFD use is now practically universal. And as companies such as Toyota and Ford demonstrated the effectiveness of QFD, its use in the software industry became inevitable (see Figure 3).

**FIGURE 3** The first 20 years of QFD in Japan and North America. Note that QFD development preceded its use at Toyota in the 1970s, and the first use of a matrix came before Mitsubishi Heavy Industry’s application at its Kobe shipyards. The first QFD book written in English was Bob King’s *Better Design in Half the Time*, published in 1987.

### The History of Software QFD

Why apply QFD to software? In almost all cases, software is developed to produce a result for the organization(s) that funded the software’s development. And in most of those cases, software is built to satisfy customers, both internal and external. To satisfy customers better (and/or faster), we must deliver value to them more efficiently. To do this, we need a framework for value and a process to ensure its efficient delivery. QFD was developed over the past 40 years for this purpose, and it has been applied to software for almost two decades.

### In Japan

Since 1982, Japanese software organizations have been using QFD with impressive results. Dr. Tadashi Yoshizawa and other leading software quality experts in Japan pioneered Software QFD. NEC IC Microcomputer Systems Ltd., which received a Deming Prize in 1987 for its world-class software-quality efforts, is a sterling example of the power of Software QFD. In 1989, with support from the Ministry for International Trade and Industry (MITI), the Information Processing Promotion Industry Association released a two-volume, 360-page book on Software QFD to further the development of high-quality software by Japanese
firms. These impressive efforts to combine sophisticated software engineering with state-of-the-art quality systems, such as QFD, are continuing.

In North America
Since 1988, North American software organizations have been using QFD. Firms such as AT&T Bell Laboratories, DEC, Hewlett-Packard, IBM, Texas Instruments, and others have reported good results with Software QFD. QFD has become an essential part of TQM for world-class software organizations and today it is part of DFSS as well.

So What Is QFD and Why Do We Need It?
To understand what QFD is and why we need it, it helps to consider a very simple model of the development process:

- **Analysis phase.** What the customer wants documented in a specification.
- **Design phase.** How the product will work documented in a design.
- **Development phase.** Make the product and test it.

Now consider two organizations with the same resources and the same time pressures to develop a product to satisfy a customer.

Incoherent Development
The first organization proceeds as usual (see Figure 4a). It gathers requirements, documents them in a specification, and attempts to meet all the requirements throughout the development process. But like most software development organizations, this organization does not have enough resources or time to meet all customer requirements and provide best efforts throughout development. Even if the spec notes the high importance of some of the customer’s requirements, there is no method to communicate such information to all the developers and through all the phases. Nevertheless, some items in each phase receive special attention, but this decision is made independently by those involved in each step of development; furthermore, by chance, some items that received best efforts in one phase also receive best efforts in a later phase. The result is a product that does not fully represent the effort the team put in, as most of the
team’s best efforts are wasted by lesser efforts upstream or downstream. Also, the product is strong in terms of meeting the requirements of greatest importance to the customer—but only by luck. Because the results depend on chance, a large organization with many development teams will have a few successes due to good luck and a few clear failures due to bad luck. In both cases, the efforts of the team are not responsible for the result.

**FIGURE 4A** An incoherent development process. The boxes on the left represent customer needs. Within each phase, best efforts are applied based on local judgments (and without regard to best efforts in the preceding phase). Over the life of the project, some best efforts align by chance, producing a product with strengths that are not predictable and that align with customer importance only by luck. Without a method to communicate what is most important to the customer end to end, alignment of best efforts to customer priority, and across phases, will be random.

**Coherent Development**

The second organization proceeds with QFD (Figure 4b). It uses QFD to discover the customer’s voice and to gather the customer’s requirements, including the priority the customer places on those requirements. There are not enough resources or time for all customer requirements to receive best efforts throughout development, so the team concentrates on the small number of high-value requirements. It uses specific formal and informal methods to communicate the high-value needs and their priority to all developers and through all phases. The high-value items receive best efforts in every phase, consistently. The result is a product that shows the team’s true capability. They performed at their absolute best, end to end, on the items that mattered most to the customer. For the same amount of resources and time as the first organization, the second organization has produced a result that is clearly superior. And because they used a systematic process to produce this result, they can repeat their success, as can all other development teams in the organization.