HIGHER CUSTOMER SATISFACTION WITH PRIORITIZING AND FOCUSED SOFTWARE QUALITY FUNCTION DEPLOYMENT

Priv.-Doz. Dr. Georg Herzwurm, Dipl.-Wirt. Inform. Sixten Schockert, Prof. Dr. Werner Mellis

University of Cologne,
Chair in Business Computing, Prof. Mellis
Albertus-Magnus-Platz
50923 Cologne, Germany
Tel.: +49 221 4705369
Fax: +49 221 4705386
Email: herzwurm@informatik.uni-koeln.de
WWW:http://www.informatik.Uni-Koeln.DE/winfo/prof.mellis/hello.htm

QFD-Institut Deutschland e. V. (QFD-ID)
Pohligstr. 1
50969 Cologne, Germany
Tel.: +49 221 4705368
Fax: +49 221 4705386
Email: qfdid@informatik.uni-koeln.de
WWW: http://www.informatik.Uni-Koeln.DE/winfo/prof.mellis/qfdid.htm

SUMMARY:

As empirical research indicates, not only the better process or realization technology determines a product's success, but also the customer. Quality is what the customer (!) assumes it to be. This article presents software engineering instruments that are not founded on the traditional way of development in software industry which is product-oriented and driven by technology. Instead, these instruments are based on customer needs. It is shown that the quality technique used in the manufacturing industry, Quality Function Deployment (QFD), after some necessary modification and extension can be applied in software engineering, as well.

1 Introduction

This paper describes the application of Quality Function Deployment (QFD) in software development. In chapter 2, we'll explain the fundamentals of Software QFD in theory and practice. After the introduction of the basic concept, two important Software QFD models will be presented in chapter 3: Zultner's comprehensive Software Quality Deployment and Ohmori's Matrix of Matrices Approach. Refining the concept and taking into account practical feasibility, Herzwurm and Schockert developed the Prifo (prioritizing and focused) Software QFD model, which is subject of chapter 4. Relating these models to practical application, chapter 5 presents experiences with Software QFD. Both the customers' point of view and that of developers will be outlined. Chapter 6 then concludes with an outlook given the actual standing of QFD.

2 Software QFD: Theory and Practice

2.1 QFD basics

Developed in Japan in the mid 60s by Yoji Akao and Shigeru Mizuno(Akao,1990, and Mizuno/Akao, 1994), Quality Function Deployment is a method to transfer customer needs into product and process requirements. The idea is to develop a product that doesn't possess all technically possible but those characteristics that customers demand ("fitness for use") and at the same time takes existing competitors into account. Any production activity is supposed to be traceable back at least indirectly to customer requirements. In each phase of development those tasks have to be pursued with the greatest effort which serve most to increase customer benefits and to satisfy customer needs.

2.2 Definition of Software QFD

Software-QFD stands for the adaptation of the classic QFD for manufacturing industries to software products. *Software* as defined by ISO/IEC 9126 means "programs, procedures, rules and any associated documentation pertaining to the operation of a computer system". *Software product* according to ISO/IEC 9126 refers to "software designated for delivery to a user". The term "user" in this context is misleading since it implies only one person who then is identical with the user. To be more precise the term "customer" or "client" should be used here as this includes a single person as well as an organization, and also the case of the customer not being the user but instead decision-maker about the acquisition. The term software product also comprises all additional services acquired with the software (e. g user support).

2.3 History of Software QFD

Like classic QFD, Software-QFD has its roots in Japan. First research in this field took place in the late 70s under the guidance of professor Yoshizawa at the Research Committee of the Japanese Society of Quality Control (JSQC). In the early 80s, the method was pushed particularly by professor A. Kanno, head of the Software Production Control Board (SPC) of the Japanese Union of Scientists and Engineers (JUSE), leading to QFD being widely accepted for the first time in the Japanese software industry. Today, Software-QFD is considered one of the most important techniques of product engineering in information technology in Japan.

In the U.S., QFD was applied to software for the first time in the late 80s, among the users were AT&T Bell Laboratories, Hewlett-Packard, Texas Instruments, Digital and lately also Andersen Consulting. The first book on Software-QFD in German was published in early 1997 and gives proof of the continuously spreading tendency in German speaking countries to apply QFD to software products (Herzwurm/Schockert/Mellis, 1997).

2.4 The House of Quality (HoQ)

The best known instrument of QFD is the so-called House of Quality (HoQ). Generally speaking, the HoQ is the matrix which analyzes customer requirements in detail and translates them into the developers' language. The HoQ is the framework of most of the matrices used in QFD. It generally consists of six different areas, also called by the generic names of WHAT, HOW, WHAT in relation to HOW, WHY, HOW in relation to HOW, and HOW MUCH (fig. 1).

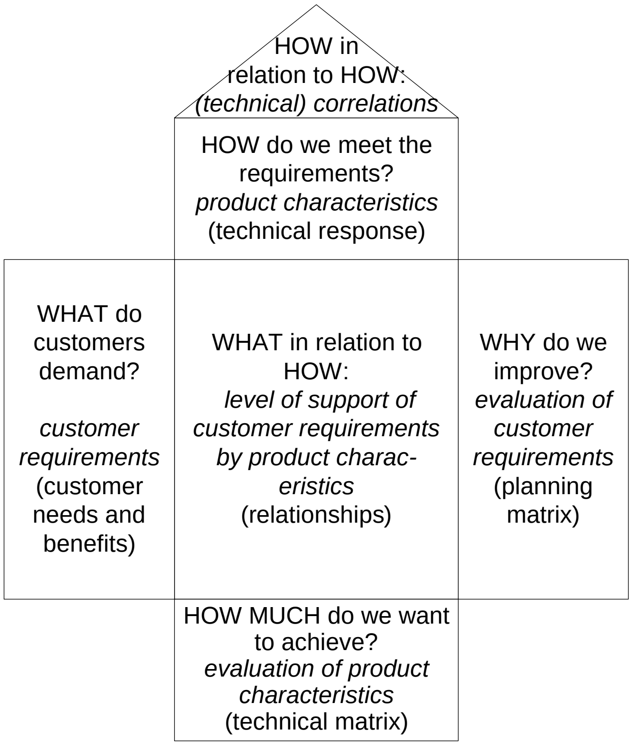

Fig. 1: Graphical presentation of the House of Quality (Cohen, 1995, p.12)

The basic idea is to set certain requirements that are given (WHAT) against (WHAT in relation to HOW) possible technical solutions (HOW). To make the house complete, concrete information about the existence of the requirements (WHY), the existing correlations among the possible technical solutions (HOW in relation to HOW) and detailed development targets (HOW MUCH) are added. However, these generic names don't fit the original content of the HoQ, because customer requirements rather represent the reasons **why** something is demanded and the product characteristics indicate **what** exactly is demanded.

Creating the HoQ is often mistaken to be the same as QFD (Zultner, 1995, p.25,27): it is only one matrix among several, although the most important one, just as the HoQ is only one element of QFD among several. Despite this, applying QFD (almost) always leads to the creation of the HoQ to form the basis for all further activities. It is also for this reason that the HoQ represents the heart of any QFD application in product planning.

2.5 Differences between Software QFD and classic QFD

The original QFD approach of Akao consists of up to 150 matrices and tables in a wide ranging matrix network. The following distinction between Software QFD and classic QFD in manufacturing is based on the widely spread Four Phase Model of the American Supplier Institute (ASI) (fig. 2), which focuses on the quality deployment part of Akao's comprehensive QFD framework. The first matrix corresponds to the *classic House of Quality (HoQ)*, which transforms customer requirements into measurable quality elements. The most important of these quality elements are then set against the characteristics of possible product components in a second matrix. These in turn correlate with the central process parameters in the third matrix, which are connected with definite production plans and means of production in the fourth matrix. So the four phases represent product planning, component planning, process planning and production planning.

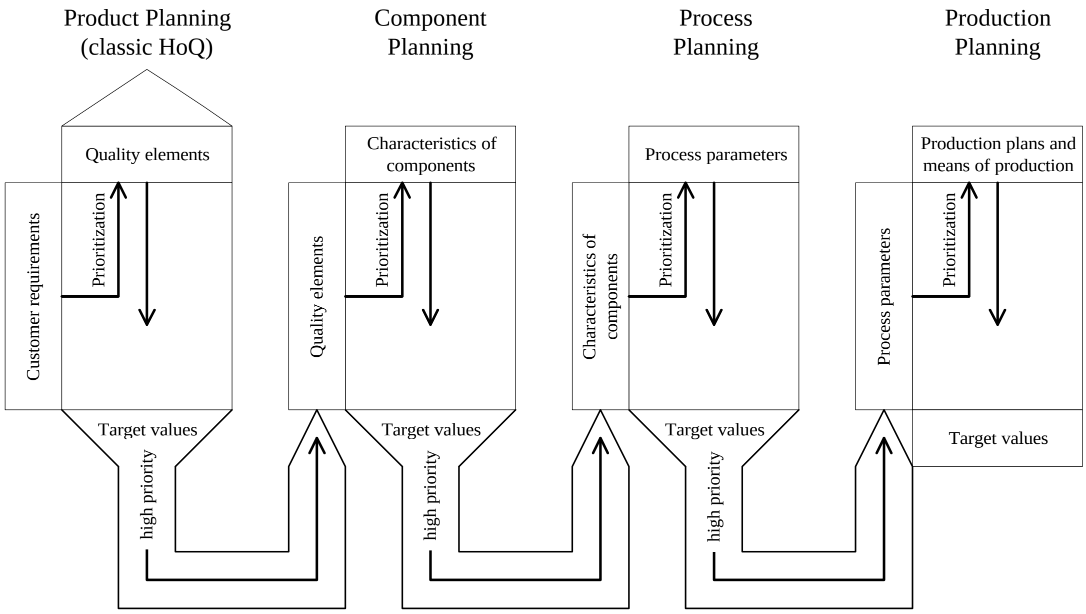

Fig. 2: Quality Deployment in the Four Phase Model (ASI, 1990, p.77)

The basic purpose of product development is universal: customers have certain demands concerning the use of products, to be satisfied by development in a complex process which has to consider time, costs and quality. In principle, QFD in its classic form can be applied to software products, as well. Two essential differences, however, have to be taken into consideration when transferring QFD to software development.

At first the product software is identified not by its physical characteristics but by its behavior. Put differently: "Software [...] is valued not for what it is, but for what it does." (Zultner, 1990, p.149) This means that simply transferring customer requirements into measurable quality elements, to be controlled in the development process, is difficult. Such a transfer would generally not be sufficient to appropriately take into account the various customer needs. Therefore, applying QFD to software products absolutely requires the function deployment part of Akao’s model, i. e. planning of functionality, which is not considered in the Four Phase Model. Besides the traditional differences between demand and solutions, in Software-QFD one has to distinguish between functional characteristics (product functions) and non-functional characteristics (quality elements) of a software product.

	Customer needs = demand	**Product characteristics = solution**	
Definition	needs resulting from using the product: business needs	characteristics or capabilities of the product, independent of implementation, which in case of high fulfillment give the customers the advantages their requirements imply	
Expression	*Customer requirement*	*Product function*	*Quality elements*
Definition	Brief, concise statements put in the customers´ words, about advantages which customers get or could get from using the product	Functional characteristic feature of the product, usually not measurable (creates perceptible output)	Non-functional characteristic feature of the product, possibly measurable during development and before delivery (does not create perceptible output)
Example for a diary	Scheduling of appointments	Setting up actions from schedule	Short time of reply

Table 1: Important definitions in Software QFD

Explicitly including the Voice of the Engineer in the form of product functions is also important in order to identify exciting factors according to the Kano model (Kano, 1984), i. e. software characteristics that customers themselves would not have come up with. The effect is such that the quality elements used in the classic HoQ in manufacturing are replaced for application in software development by product functions. According to the general definition of the HoQ in which particularly customer requirements are being analyzed this new first matrix can be called *Software-HoQ*. The quality elements are first taken into consideration in the second matrix.

Secondly, in the software industry the production process in a strict sense is a mere duplication process, just as the definite implementation process can hardly be influenced by special adjustable process parameters. Therefore, in a still higher degree than in manufacturing the problem lies in the early phases of development. Applying QFD in software development therefore has to focus on the ability to prioritize the engineering activities and pay less attention to the deployment down to the software's last line of code.

3 Software QFD Models

3.1 Zultner's comprehensive Software Quality Deployment

Nevertheless, Zultner developed a framework of how to apply Akao's comprehensive QFD to software development, including quality deployment according to the Four Phase Model (fig. 3). The most important additional aspect is the stress on *customer deployment* before quality deployment, developed from the notion that software like most other products rarely is conceived to satisfy the needs of only one homogenous customer. To identify customer groups and their importance for the engineering ahead, a table showing potentially relevant customer characteristics and a prioritization matrix of selected criteria are used.

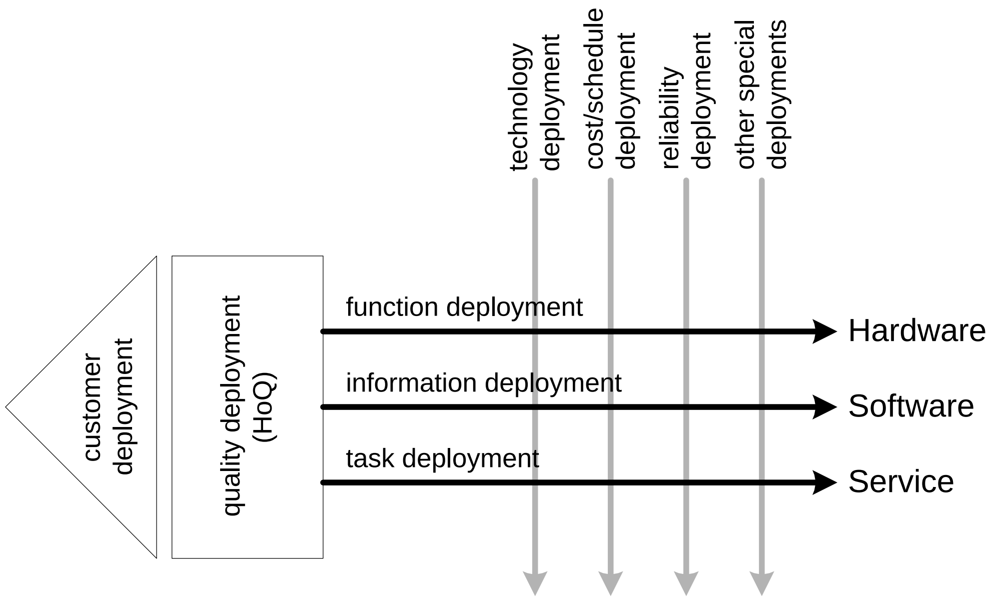

Fig. 3: Zultner's comprehensive software quality deployment (Zultner, 1991, p.451)

Quality deployment as a second step combines the classic and the Software-HoQ to form one single HoQ, such that functional as well as non-functional product characteristics are being considered (including Akao's function deployment). It is only through information deployment that at least the first two phases of the Four Phase Model have an equivalent in software development. In information deployment the product functions that have been prioritized in the HoQ are turned into entities, processes or objects, depending on the development techniques used. Function deployment (not to mix up with the term used in paragraph 1.4) proceeds similarly when hardware design is required, and task deployment is concerned with activities of the development process itself. Vertical deployments are meant to guarantee that general aspects such as reliability and costs are taken into account in any development activity. In this, (possibly) measurable quality elements taken from the HoQ are particularly important.

Only recently, Zultner reduced his QFD approach to absolutely essential activities (Zultner, 1995). This model, called "Blitz QFD", focuses entirely on the reception, analysis and weighting of customer requirements, after customers have been identified (customer deployment) and before the HoQ has been set up (quality deployment). Thereby, most advantages of applying QFD are expected to be achieved in less time with less effort.

3.2 Ohmori’s Matrix of Matrices Approach

Ohmori presents his framework of action for the new development of commercial individual software in a complex matrix-matrix-diagram, yet this approach with a total of 14 matrices only covers quality deployment as it is described in the first two phases of the Four Phase Model (Ohmori, 1993). Essentially new in this approach are several activities for analyzing a comprehensive business system which combines all tasks necessary to reach the organization’s goals. Software as part of this higher task system has to support some of these business system tasks with its basic functions. Once these high-level functions are known, customer requirements (here called software quality requirements) are being identified, to be set against the product functions (software additional functions) in the Software-HoQ and (software) quality elements in the classic HoQ. In another matrix, design points can be deduced from the importance of the quality elements for each product function. These design points indicate certain quality elements that have to be fulfilled in a higher degree when a particular product function is being implemented. In the next step, the functional product characteristics are being related to individual software components such as software subsystems, data files or, later, program modules. The great amount of matrices (fig.°4) results from rigorously taking into account quality elements (not necessarily measurable), concerning the business system as well as the business software.

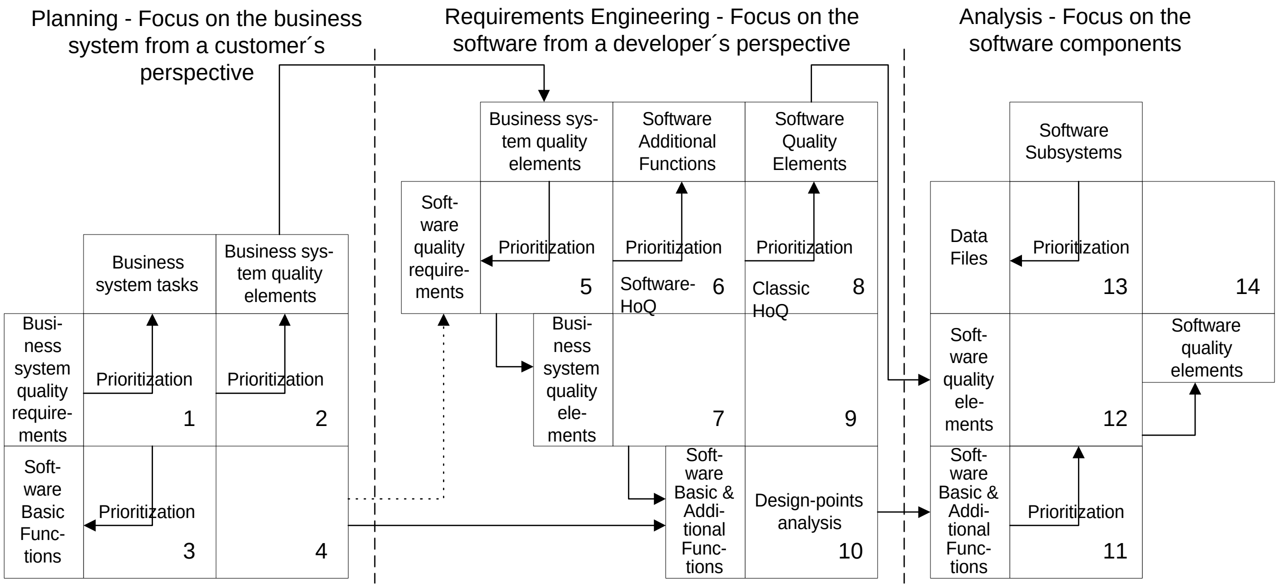

Fig. 4: Ohmori’s Matrix of Matrices approach (Ohmori, 1993, p.219)

Ohmori’s approach consists of three phases. In addition to supporting requirements engineering and the early stages of the analysis phase a planning phase exists in order to embed the software into a higher business sytem. To be able to handle concrete customer requirements concerning software, developers need to have an idea of how the software will be used by the customers. Therefore, the initial focus is a business process-oriented one on the functions the software has to support.

4 Herzwurm's and Schockert's PriFo Software QFD Model

The important first purpose of QFD in software engineering and the main focus of product planning is on setting prioritized development goals based on the most important customer requirements (Herzwurm/Schockert/Mellis, 1997). In planning software products the preference setting and focusing aspects of QFD by means of the HoQ are more important than the deployment by a matrix sequence. That is the reason why the approach presented in this paragraph is called PriFo (prioritizing and focused) Software QFD. Applying QFD, however, takes more than filling out a HoQ matrix. A number of techniques (e. g. the Seven Management and Planning Tools and the Seven Quality Tools) have to be combined in order to get all information that is necessary to form the matrices and to exhaust the potential of QFD as far as possible. Joining these techniques into a model of how to go about planning software products, including some organizational aspects that are important at the time of putting a QFD project into practice, is what the PriFo Software QFD Model is about.

4.1 Pre-Planning

The first task of a QFD project concerns *setting the project's goals*, discussing the *time schedule and cost planning* and putting together a *QFD team*. Apart from these activities, the planning phase of a QFD implementation includes also *defining the project's content (product definition), identification of the customer groups and their importance* for the development ahead as well as *selecting customer representatives*. This phase which we will call Pre-Planning consists basically of normal meetings and Brainstorming sessions of the persons in charge of the project. To define the product one or several QFD matrices from Ohmori's planning phase can be used, and Zultner's customer deployment can serve as guideline in identifying potential customer types and the weight of different customer groups.

4.2 The Voice of the Customer Analysis

The entire QFD process is carried by a QFD team put together from all departments (development, quality management, marketing, sales, service etc.), to be extended in several team meetings by the selected typical customer representatives. Substituting for a customer survey one of the first meetings tries to ascertain customer needs and classifies them in the Voice of the Customer Table to identify customer requirements. Then these requirements are structured using affinity- and tree diagrams, weighted [e. g. with the Analytic Hierarchy Process and comparison by pairs (Saaty, 1995)] by as many members of the customer groups as possible under the overall control of the customer representatives, and in case of a further development of a product the customer representatives will evaluate them according to the level of satisfaction the fulfillment of the requirements has reached up to that point (measured on a scale ranging from 1 indicating total dissatisfaction to 5 indicating perfect satisfaction). A (subjective) comparison with competitors at the requirements level is costly since one's customers cannot be assumed to be able to evaluate the competition's products, as well. Normally, therefore, additional customer representatives would have to be consulted. Combined with weighting and identifying satisfaction levels caused by the fulfillment of customer requirements, this could be done conducting a wide-ranging customer survey. The result of all these efforts is the *table of customer requirements.*

4.3 Building the classic HoQ and the Software-HoQ

The *table of customer requirements* is the input of both quality tables to set up, the *Software-HoQ* representing function deployment and the *classic HoQ* representing the Four Phase Model. Identifying the product functions that will go into the Software-HoQ as additional input is done similarly to the identification of customer requirements, using a Voice of the Engineer Table. The difference is that in this meeting usually only the members of the QFD team are present, among them particularly the developers. Turning requirements into measurable quality elements as an additional input of the classic HoQ takes place at another internal QFD meeting. Identifying the relationships between product characteristics and customer requirements in both prioritization matrices is ideally done together with the customer representatives. A simplified product benchmarking at further internal team meetings concerning the prioritized characteristics and including the setting of definite development goals, leads to a *table of the most important product functions* and a *table of the most important quality elements*. Figure 5 displays an excerpt of a Software HoQ including the tables of customer requirements and product functions.

	integration in R/3	starting of processes on determined dates	R/3-interface	integration of a company diary	operation	cut, copy & paste	activation of objects by mouse-clicks	network features	simultaneously usable by a high number of users	keeps appointments for a high number of persons	worldwide access	appointments	appointments can be connected with long texts	maintaining periodically recurring appointments	working on appointments	getting reminded of fixed appointments	group appointments	displaying a list of participants for group appointments	user interface	visualizing overlapping appointments	...
operation																					
avoidance of input mistakes			9			3	9							3	3					9	
easy switching between different views			9				9														
team-work																					
other persons can look up or work with saved data									9	9	9				9						
it can be worked on several persons' appointments				1					9	9	9				9						
managing appointments																					
working on appointments			1	3			9						3	7	9					3	3
easy shifting of appointments			1	3		9	3								9						
special input for periodically recurring appointments			1	3		9	3							9	7						
documenting appointments			1										9		3						
group features																					
detection of overlapping appointments			1						3	3	3				3					9	
easily acessible system for the information other persons				3					3	3	3							9		9	9
automatisms																					
getting reminded of appointments		9											3			9					
...		3																			
referring to overall importance of customer requirement																					
absolute importance		169	164	83		67	159		148	135	135		63	72	309	49		38		77	30
relative importance		3.46	3.36	1.70		1.38	3.26		3.03	2.77	2.77		1.30	1.48	6.34	1.01		0.78		1.58	0.61
standing		5	6	17		25	7		8	9	9		26	21	3	36		41		18	50
Maximum = 7.2 — relative importance Minimum = 0.0																					
quantitative assessment																					
current Diary		1	5	5		4	5		3	3	5		1	1	5	1		1		5	
objective		3	5	5		5	5		4	4	5		3	3	5	3		3		5	
degree of difficulty		4	1	1		5	1		5	5	1		3	2	1	3		3		1	
relation of improvement		3.00	1.00	1.00		1.25	1.00		1.33	1.33	1.00		3.00	3.00	1.00	3.00		3.00		1.00	

	weightiness of customer requirements	customer-group A (in %)	customer-group B (in %)	customer-group C (in %)	totality (in %)	customer satisfaction	customer-group A	customer-group B	customer-group C	importance of customer requirements	customer-group A (in %)	customer-group B (in %)	customer-group C (in %)	totality (in %)	overall standing	Minimum = 0.0 — totality (in %) Maximum = 5.0
operation																
avoidance of input mistakes		2.50	1.00	4.00	2.50		2.40	3.00	3.00		2.67	0.69	4.50	2.62	11	
easy switching between different views		3.30	1.00	4.00	2.77		4.60	3.33	5.00		1.84	0.63	2.70	1.72	22	
team-work																
other persons can look up or work with saved data		6.02	5.00	4.00	5.01		4.60	4.67	5.00		3.36	2.23	2.70	2.76	9	
it can be worked on several persons' appointments		4.90	4.00	3.00	3.97		3.80	4.67	5.00		3.31	1.78	2.03	2.37	14	
managing appointments																
working on appointments		7.14	6.00	12.00	8.38		4.80	4.00	5.00		3.82	3.12	8.11	5.02	1	
easy shifting of appointments		1.51	5.00	3.00	3.17		3.80	4.33	5.00		1.02	2.40	2.03	1.82	20	
special input for periodically recurring appointments		1.89	3.00	3.00	2.63		3.00	1.33	3.00		1.62	4.70	3.38	3.23	7	
documenting appointments		0.55	0.20	0.60	0.45		1.20	1.33	1.00		1.18	0.31	2.03	1.17	40	
group features																
detection of overlapping appointments		1.21	2.00	1.00	1.40		3.40	4.00	5.00		0.91	1.04	0.68	0.88	46	
easily acessible system for the information other persons		1.12	0.50	1.00	0.87		4.60	3.00	5.00		0.63	0.35	0.68	0.55	54	
automatisms																
getting reminded of appointments		2.60	2.40		1.67		1.00	1.00			6.68	5.00		3.89	3	
...		0.72	4.50		1.74		1.00	1.67			1.85	5.61		2.49	12	

Fig. 5: Excerpt of a Software-HoQ for SAP's R/3 Diary (generated with QFD/Capture™)

SAP's R/3 Diary may serve as an example: integrated into the standard software system R/3, this piece of software enables several persons to view and maintain each other's appointments from different locations at the same time. The numbers 0, 1, 3 and 9 in the cells represent the degree of correlation between the satisfaction of a customer requirement (lines) by a product function (columns). In the most simple of cases the importance of an individual product function is given by the sum of the multiplications of each requirement's weightiness and the corresponding degree of correlation. In the case of a further development of a product, however, instead of the weightiness of the customer requirements, the importance of the requirement can be taken as the quotient of weightiness of the customer requirement over customer satisfaction. Moreover, the present data allow for numerous other quantitative and qualitative analysis. For example, a consistency analysis can be carried out: blank lines (i. e. a customer requirement without correlation) indicate that product functions are missing respectively have been overlooked, and blank columns (i. e. a product function without correlation) hint at the possibility that an unnecessary product function has been defined.

4.4 Design points analysis

The development goals derived from the product characteristics are related in a third matrix according to Ohmori and as a final step of consolidation compared among themselves. At the same time, they are examined for potential synergy effects or conflicts when realizing them and then narrowed down to design points as two-dimensional target values. Together with the most important customer requirements, product characteristics and design points form the basis for setting up a requirements specification as a result of the Requirement Engineering process. In additional matrices, they can be turned into more specific development goals, at least of the analysis phase. Fig.°6 shows the procedure of software product planning with QFD as described above.

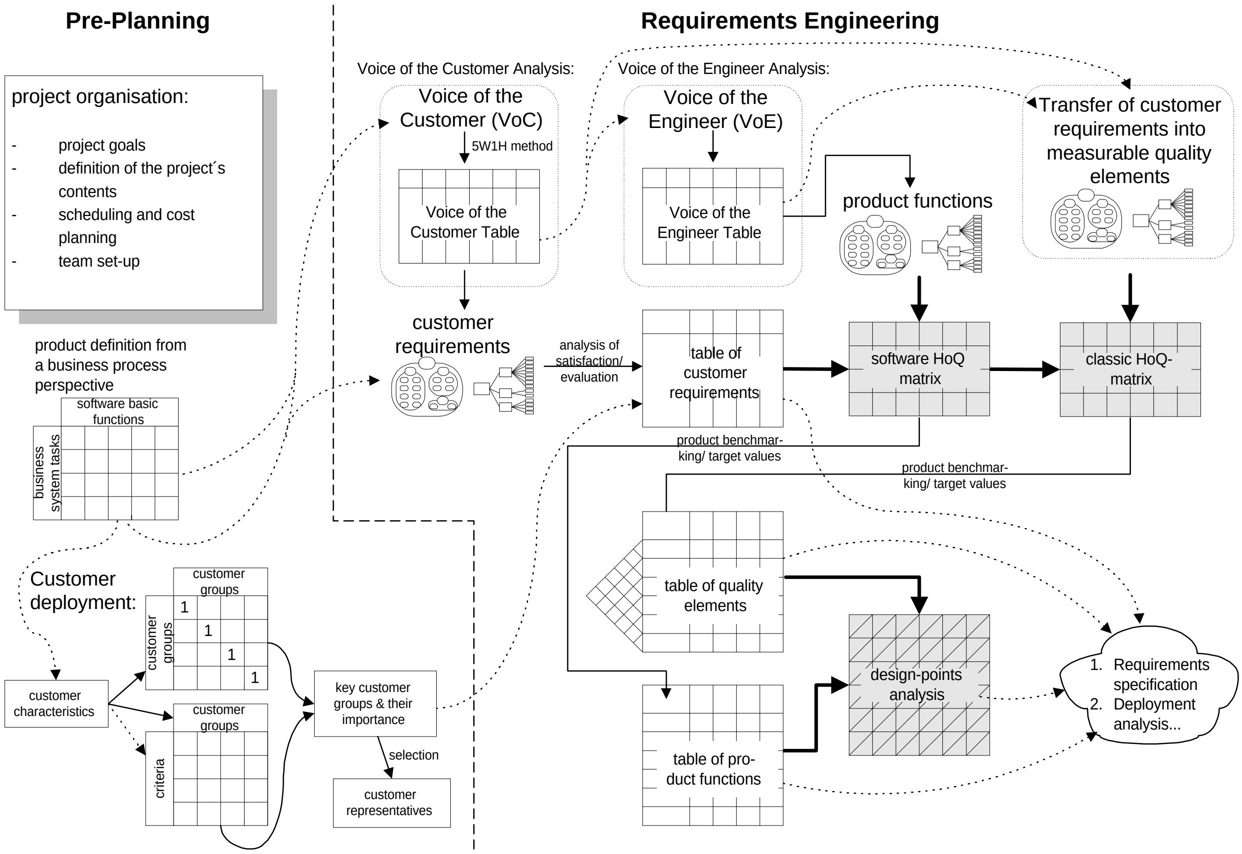

Fig. 6: Herzwurm’s and Schockert’s PriFo Software QFD Model

5 Experiences with Software QFD

Experiences with the application of QFD in software engineering show that QFD is particularly beneficial in the interdisciplinary communication, clear understanding of customer and/or user requirements, in the consent about the solutions found, a reduced number of post-delivery changes, complete documentation of all steps taken, a profitable product and satisfied customers. The disadvantages of QFD consist in its complexity, and the considerable amount of time needed for preparing, carrying out and evaluating the meetings afterwards.

In 1997 we conducted an empirical investigation of 16 QFD projects - among them seven software projects - in which product developers were asked about their experiences with QFD. It confirms that QFD fulfills the special expectation which is connected with the employment of the instrument in product development (figure 7). Concerning the customer-oriented objectives technical and relative quality the employment of QFD achieves very high satisfaction values. From the project-related goals point of view QFD particularly improves the co-operation of the persons involved and, due to focusing on the substantial, at the same time leads to a higher economy of the product development.

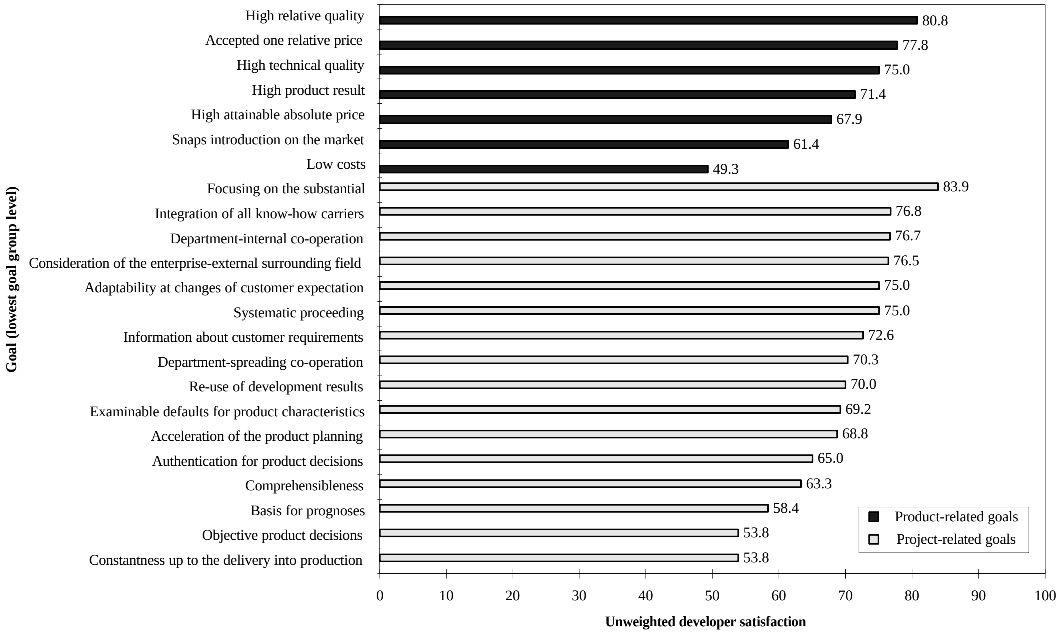

Fig. 7: Satisfaction of the developers with product development goals

In the SAP's R/3 Diary project (see chapter 4.3) we asked the customers before and after the project whether they were content with the Dairy and its development process. With the help of the opinions given about the criteria stated above, it was possible to determine an index of the degree of satisfaction with every single criterion and for the Diary on the whole (= index of customer satisfaction as mean of the evaluated assessment factors). The results show a fairly high degree of satisfaction with the Diary. Comparing the ”Diary with QFD” and the ”Diary without QFD”, however, shows that even this rather high level of satisfaction can still be increased by QFD. This is true for each and every criterion as well as for the Diary on the whole (without QFD 66.7; with QFD 82.0).

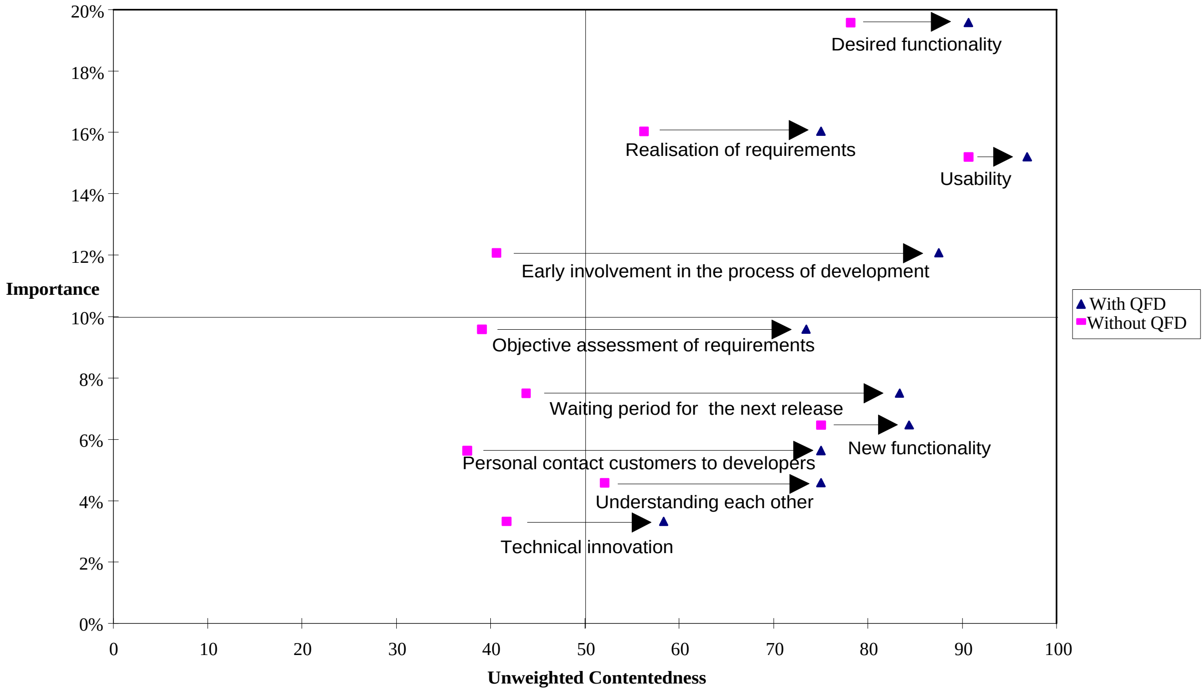

Fig. 8: Satisfaction of the SAP R/3 Diary customers with and without QFD

6 Conclusion

From the traditional software engineering point of view, the current support of Software QFD which only deals with the first part of the software engineering process must be criticized. It should be possible to transfer the results of QFD to the late phases of realization and implementation without losing information. Among other things, this requires the accurate integration of QFD into traditional software process life cycle models (IEEE, 1996), and a symbiosis of Software QFD and classic software engineering techniques.

Since experiences with QFD in software engineering have been extraordinarily positive up to date, however, this allows for hope that continued application and follow-up development of Software QFD in science as well as in practice will help to solve these problems in the short term. Along this line, in 1996 the QFD Institut Deutschland e. V. founded a working group called "QFD for software" which is particularly concerned with the integration of QFD into traditional models of the software engineering process.

The results of the working group show that QFD can be integrated without any problem into the V Model ’97 (http://www.v-modell.iabg.de), which is the standard model recommended by German federal authorities in software engineering. As support of QFD, a variety of software tools is being offered in the market. These tools allow the flexible definition of an individualized process, thereby simplifying the practical employment of software QFD. Some companies successfully use calculation sheet programs such as Microsoft Excel for processing and analyzing QFD data. As this article has shown, Software QFD can be employed successfully in practice even at the actual state of knowledge and experience.

Bibliography

Akao, Yoji: Quality Function Deployment: Integrating Customer Requirements into Product Design. Cambridge, Massachusetts 1990. ISBN: 0-915299-41-0

ASI: Quality Function Deployment - Excerpts from the Implementation Manual for Three Day QFD Workshop. Version 3.4. In: QFD-Institute (ed.): Transactions from the Second Symposium on Quality Function Deployment. Novi, Michigan 1990, pp. 21-85

Cohen, Lou: Quality Function Deployment. How to Make QFD Work for You. Reading u. a. 1995. ISBN: 0-201-63330-2

Herzwurm, Georg; Ahlemeier, Gabriele; Schockert, Sixten; Mellis, Werner: Success Factors of QFD Projects. In: Proceedings of the World Innovation and Strategy Conference, August 3-5, 1998, Sydney, Australia, pp. 27-41

Herzwurm, Georg; Schockert, Sixten; Mellis, Werner: Determining the Success of a QFD-Project - exemplified by a pilot scheme carried out in cooperation with the German software company SAP AG. In: Proceedings of the Eigth Synmposium on Quality Function Deployment, June 9-11, 1996, Novi, Michigan, USA, pp. 131-150

Herzwurm, Georg; Schockert, Sixten; Mellis, Werner, Claudius Weinberger: Customer Oriented Evaluation of QFD-Tools. In: Proceedings of the Third International Symposium on Quality Function Deployment, October 2-3, 1997, Linköping, Sweden, vol. 1, pp. 309-323

Herzwurm, Georg; Schockert, Sixten; Mellis, Werner: Qualitätssoftware durch Kundenorientierung. Die Methode Quality Function Deplyoment (QFD). Grundlagen, Praxisleitfaden, SAP R/3 Fallbeispiel. Braunschweig - Wiesbaden 1997. ISBN: 3-528-05577-4

IEEE (ed.): IEEE Standard for Developing Software Life Cycle Processes. IEEE Std 1074-1995. New York 1996

Kano, Noriaki; Seraku, Nobuhiko; Takahashi, Fumio; Tsuji, Shinichi: Attractive quality and must-be quality. In: Quality. No. 2, 1984, pp. 39-44 (in Japanese)

Mizuno, Shigeru; Akao, Yoji (ed.): QFD, the customer-driven approach to quality planning and deployment. Tokio 1994. ISBN: 92-833-1122-1

Ohmori, Akira: Software quality deployment approach: framework design, methodology and example. In: Software Quality Journal. No. 3, 1993, pp. 209-240

Saaty, Thomas L.: Decision Making for Leaders: The Analytical Hierarchy Process for Decisions in a Complex World. 3. edition., Pittsburgh 1995. ISBN: 0-9620317-8-X

Zultner, Richard E.: Before the House. The Voices of the Customers in QFD. In: QFD-Institute (ed.): Transactions from the Third Symposium on Quality Function Deployment. Novi, Michigan 1991, pp. 451-464

Zultner, Richard E.: Blitz QFD: Better, Faster, and Cheaper Forms of QFD. In: American Programmer. October 1995, pp. 24-36

Zultner, Richard E.: Software Quality [Function] Deployment. Applying QFD to software. In: QFD-Institute (ed.): Transactions from the Second Symposium on Quality Function Deployment. Novi, Michigan 1990, pp. 132-149