Domain focused requirements engineering: A case study of successful requirements engineering in a market-oriented automotive software development company

Return to Published Papers

Irene Caulfield, Norah Power
Lero, University of Limerick, Ireland
E-mail: irene.caulfield@lero.ie, norah.power@lero.ie

Abstract

The aim of this paper is to explore effective market-oriented requirements engineering practices in automotive aftermarket software development. We present a real world case study of how a particular combination of practices, including Software QFD, helped improve the product development cycle in an automotive aftermarket organisation involved in software development.

1. Introduction

The goal of this research is to methodically study the requirements engineering practices in use in a particular type of requirements situation, namely the market-driven automotive aftermarket.

Our research is a longitudinal case study that will encompass an in-depth study of all the requirements engineering practices from initial requirements elicitation and analysis through to negotiation, documentation, validation, verification and inspection of requirements artefacts. In particular we are investigating what requirements engineering practices are effective in this particular type of requirements situation and why. Having established which practices are effective we aim to examine the factors that make these practices a success.

This preliminary part of the investigation focuses on requirements elicitation and negotiation practices. This paper describes interim results of that research. The remainder of the paper is divided into four sections: section 2 outlines the different requirements situations that occur in software development projects and emphasises the need for automotive domain specific requirements engineering research.

Section 3 outlines the widely recommended requirements engineering practices according to published literature to-date. In this section we explore the use of Software Quality Function Deployment (Software QFD) as a technique for market-oriented requirements engineering and we describe the benefits of choosing this method particularly when considering new product development.

In section 4, we present a real world case study of how a market-oriented software development organisation has improved their product development cycle by embracing a combination of requirements practices that includes Software QFD. Section 5 analyses the success factors observed in the case study and compares these observations with key practices in the peer-reviewed literature.

2. Requirements Situations

There are many types of situations in which software is developed. Lauesen [1] recognises several different situations and categorises them according to seven project types: In-house, Product development, Time and materials, Commercial off the Shelf (COTS), Tender, Contract development and Sub-contracting.

Other research [2] on requirements documents describes a classification schema that categorises requirements situations into three main situation types: market-oriented, in which generic off-the-shelf solutions are developed, solution-oriented, in which a software product is delivered for a specific situation of use and problem-oriented, in which organisations document a problem but lack the technical resources to

develop the solution and subsequently seek solutions from external providers. The general situation addressed in this paper is market-oriented [2], often called Product Development [1].

2. 1. Market-Oriented RE

Market-oriented requirements engineering as distinct from problem-oriented or solution-oriented software development, requires a different approach to the requirements processes of elicitation, analysis, documentation, validation, verification and management.

In market-oriented requirements engineering there are numerous sources of requirements including end-users, existing and potential customers, regulatory bodies, competing products, market analysis and previous versions of the product.

2.2 Domain Focused RE

In recent years there has been increasing recognition of domain specific problems in requirements engineering and the need for solutions appropriate to those [3-5]. Automotive requirements engineering is one such domain [6, 7]. Requirements engineering has a major role to play in developing highly complex software-intensive automotive systems.

The first domain focused automotive requirements Engineering Workshop (AuRE 04) recognised that software has become a major component of a new car [8]. The workshop focused on a number of topics including the needs and challenges of automotive RE and specifically addressed the growing challenge of geographically distributed software development in this domain. The second International automotive requirements engineering workshops (AuRE 06) focused on improving automotive RE processes and practices [9].

There are two emerging research opportunities for market-oriented requirements engineering in the automotive domain. The first research opportunity is the impact of the AUTOSAR architecture on the market for automotive software. The AUTOSAR standard which is subscribed to by companies such as BMW, BOSCH, Ford, GM, Siemens, Toyota and VW will open up the supply side to existing and new software development companies [10]. Since this is a new approach there are very few opportunities to study how it has worked in practice.

The research reported here focuses on the existing and growing market for software that is used after a motor vehicle has gone into use, namely the automotive aftermarket.

2.3. Automotive Aftermarket

The automotive aftermarket consists of companies that distribute and sell many different products including diagnostic tools, parts and equipment for the maintenance and repair of vehicles.

The global automotive aftermarket is substantial in monetary terms. The total aftermarket forecast for 2006 in the US is $278 billion [11]. In Europe the light vehicle aftermarket is currently worth €160 billion across 15 markets including Germany, UK, Spain, Sweden, and the Czech Republic [12].

3. Effective RE Practices

Many well known practices have been recommended to assist with the task of requirements engineering. These include Joint Application Development (JAD), Rapid Application Development (RAD), Prototyping, Rational Unified Process (RUP), Structured Analysis and Design, Participatory Design, and eXtreme Programming (XP). Each of these approaches has both advantages and disadvantages depending on the situation in which they will be used.

In this section we argue that for market-oriented requirements engineering, adopting certain aspects of Software Quality Function deployment (Software QFD) may be more effective than the methods listed above. Software QFD has been adapted for software development from a manufacturing technique called Quality Function Deployment (QFD).

3.1. Quality Function Deployment

Quality Function Deployment (QFD) was invented by Yoji Akao and Shigeru Mizuno of Japan during the Total Quality Management (TQM) movement [13]. Akao and Mizuno wanted to develop a quality assurance method that would design customer satisfaction into a product before the product was manufactured. Prior to this, quality control methods focused on fixing a problem during or after manufacturing.

QFD is a customer-driven formal technique for deriving a product specification. At the outset, QFD aims to listen carefully to customer needs and it is these needs that drive the product design and production process [14]. According to Madu [15] QFD “is a process of listening to the voice of the

customer identifying and incorporating those needs in the design and production of goods and services".

3.2. Voice of the Customer

The QFD approach is premised on a very simple concept, namely gathering the customers' needs and representing these as the "Voice of the Customer". The QFD technique takes the "Voice of the Customer" and converts this into specifications at various stages of the product development process.

3.3. Software Quality Function Deployment

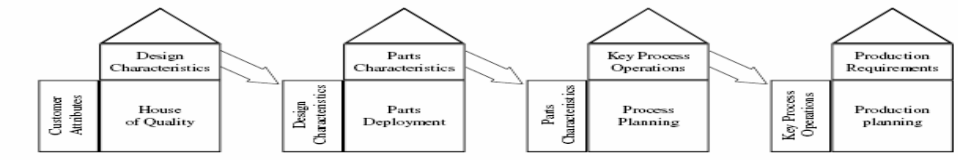
Although QFD has its origins in manufacturing it has adapted well to software development, because of its ability to bridge the gap between the natural language of customers and the scientific, formal language of designers and engineers. This is an important issue for requirements engineering, accepting and documenting the customers' input however informal those inputs may be. According to Herzwurm [16] QFD "bridges a gap in the software development process to the customer. This is done using a systematic procedure for teamwork and the ability to prioritize all information concerning product development in a justified way".

Software QFD focuses on improving the quality of the software development process by implementing quality improvement techniques during the front-end requirements elicitation activity. These quality improvement techniques lead to increased analyst and programmer productivity, fewer design changes, a reduction in the number of errors passed from one phase to the next [17, 18].

3.4. The Four Phase Model

In QFD, the various stages of the product development process are represented as a series of matrices. Whilst a number of different QFD approaches exist based on varying numbers of matrices, the most widely described and used model in manufacturing is the four phase model, sometimes referred to as the "Clausing" [19] or "ASI" model [20]. The model illustrated in Figure 1 contains four matrices, of which the first is the most important for requirements engineers: Product Planning (House of Quality), Parts Deployment, Process Planning and Production Planning.

Figure 1. The four-phase model



3.5. The Product Planning Matrix

The Product Planning matrix, otherwise known as the "House of Quality", can be used as a front-end requirements engineering technique for software development projects. The "House of Quality" can have a significant impact on assisting an organization with eliciting and prioritising the requirements [21].

3.6. House of Quality

The "House of Quality" is a tool that links the voice of the customer with design decisions that need to be made and is more relevant to requirements engineering than any other QFD concept. The essence of the "House of Quality" is to document the characteristics of a product as described by the customer and to list these as "Whats".

The "Whats" are represented in the vertical axis, while the horizontal axis of the matrix represent "How" the designers plan to convert those "Wants" into features of the product. For example, "Large Display" may represent a customers "What", while a "7 inch full VGA LCD" could be a "How" that the designers have determined will fulfill that "What". Once the list of requirements has been completed, they will be ranked in order of importance with the most important sitting at the top.

The next step is to complete the "Roof" of the house. The "Roof" acts as a place to evaluate the "Hows" in isolation from the "Whats" described by the customer. Using the "Roof", a team will discuss whether delivering one "How" will have a positive or negative impact on another and so on. The "Roof" helps the team to decide which "Hows" will go into the product and also helps identify where tradeoffs and further negotiations must take place before a final decision can be made.

The "Voice of the Customer" and the "House of Quality" are widely used parts of QFD and are two key Requirements engineering practices that we consider when we refer to Software QFD for the remainder of this paper. A basic "House of Quality" is illustrated in Figure 2.

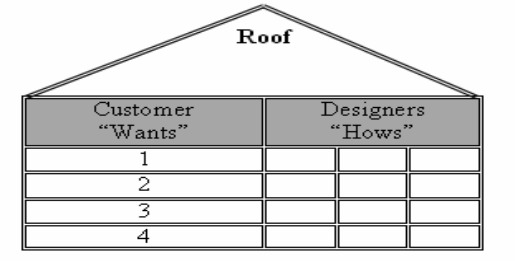


Figure 2 Basic "House of Quality" diagram

3.7. Requirements Engineering Methods

Capers Jones [22] studied more than 650 organisations representing approximately 9,000 projects in the period from 1984 to 2000. The data collected formed a comprehensive database of the methods and techniques used for the entire development process including requirements. From the data he short-listed the 14 most effective RE methods; these included Software QFD.

Haag conducted an analysis of 25 software development projects in five companies [23]. Using 12 criteria for requirements engineering, he found QFD rated more highly than "traditional" methods. Two of the reasons for success were better communication among the development team as well as between customers and developers, and greater fulfillment of customer expectations.

Text books such as [24] are concerned with effective RE methods and practices. Young lists 35 effective methods and techniques for RE and includes QFD in his recommendations.

Davis [18] has compared Software QFD against eight RE methods and has found many similarities between each approach and QFD. The Primary benefits of Software QFD described in the literature are [16, 18, 25]:

- Communicating clear understanding of customer requirements
- Improved customer satisfaction due to inclusion of the "Voice of the Customer"
- Promotes teamwork
- Reduction in development time as a result of concurrent engineering
- Consent about solutions
- Flexibility of the process
- Reduces post delivery changes
- Complete documentation of all steps
- Improves internal communications
- Improves morale and organisational harmony
- Increased competitiveness and high market acceptance of the products

4. Case Study

The term "case study" can be interpreted in two different ways. A case study can be used as an exemplar to demonstrate new method [26-28].

Another type of case study, such as the one presented in this paper focuses on observing a real world situation with the purpose of developing a holistic understanding of that domain and qualitatively examining what constitutes effective requirements practices in that particular situation of use.

The case study method is an accepted empirical enquiry method that "Investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident" [29].

Case studies rely on two sources of evidence, direct observation and interviews with the persons involved. The case study's unique strength is its ability to deal with a full variety of evidence – documents, artifacts, interviews and observations [29].

4.1. Case Study Method

This extended case study is investigating the current requirements engineering practices in a market-oriented automotive company engaged in distributed software development in the US, Europe and ASIA. At this interim stage of the extended case study we are focusing only on requirements elicitation, negotiation and prioritisation practices in the company. For the purpose of this paper the company will be called "Independent Tools"[1].

The data collection so far has consisted of a series of formal and informal meetings with stakeholders and semi-structured interviews with key informants, which were recorded, transcribed and analysed. The case study also involved product demonstrations, observations and a review of requirements documentation.

The Director of European Engineering, the Manager of New Product Development and core team members have been important sources of material in the case study so far.

4.2. Company Background

Independent Tools are developing software for the automotive aftermarket. The Aftermarket is that part

[1] Not the real name of this company for confidentiality reasons.

of the automotive industry that is concerned with motor vehicles after they have gone into use.

The aftermarket is a significant sector of the automotive market. A key aspect of any new car sales business is its after-sales service. Car manufacturers recognise the importance of the aftermarket business and provide main dealers etc. with proprietary diagnostic tools for their own models.

Independent Tools are engaged in market-oriented requirements engineering. They provide hardware and software diagnostic tools that can interface with cars from a variety of manufacturers to independent workshops and other customers, and sell these tools right across the market to a wide variety of customers throughout a well-developed franchise and dealer network.

The challenge for Independent Tools lies in the fact that the hardware tools they supply need to be updated regularly with software updates that keep abreast of changes in the automotive market. This case study examines the requirements processes involved in the production of those software updates.

In the case study we explore why the former Product Development Cycle was not suitable for handling the regular software updates. We follow this by describing how the company responded to the challenge by adopting the new improved Product Realisation Cycle.

4.3. The Original Product Development Cycle

As automotive engineers, the company has come from a mechanical engineering background. When they moved into software they continued to use traditional mechanical engineering techniques for product development, this became unworkable because the development process was not flexible enough for the new product range.

According to the Product Manager

> "It was so rigid that we had to go through this phase of the project and get sign off here and go to the next phase and if a project took two years it probably didn't make a whole lot of difference because those products were going to sell for 20 years probably".

The original Product Development Cycle was described as a hierarchical, sequential process that included 5 phases. The original Product Development cycle is outlined in Table 1.

Table 1. The original "Product Development Cycle"

Phase 1	Product Definition
Phase 2	Product Design
Phase 3	Pilot Runs
Phase 4	Production Run
Phase 5	Customer Feedback

The main problems perceived with the original Product Development Cycle were:

1. Not customer focused enough
2. Product Development Cycle was too long
3. Reaching a consensus was difficult
4. Apportioning blame "The Silo Problem"
5. Project Managers acting as "Program Police"
6. Did not depend upon a team approach
7. Not meeting face-to-face

1 The key concern was the lack of customer focus. They felt they were not developing the right product for their customers and were failing to deliver their products to the market on time.

2 The product development cycle for new products was taking too long. The phased structure encouraged sequential, not parallel, thinking. It was difficult to progress from one step to another due to each step requiring sign-off and this was difficult to achieve.

3 Reaching a consensus was very difficult. In order to progress from one phase to another sign-off was required. This was difficult to achieve, as the different departments were not coordinating their efforts, felt no responsibility for delivering the product on time, and consequently felt no pressure to sign-off each phase.

4 One clear objective was to break down the problem of apportioning blame that existed on previous development projects. The engineering director described this as "The Silo Problem" where groups would point the finger of blame at each other for example saying, "It's an engineering problem" or "it's an operations problem".

5 Project managers became "program police". They were not involving themselves in problem solving and were taking no ownership of the projects and were pointing the finger of blame instead.

6 Did not depend upon a team approach. Each department worked on a different stage of the cycle

and there was little or no overlap with other departments.

7 The sales and marketing staff were mainly based in Germany and the UK, the two biggest markets for the company. Some development work was also carried out in another European country. As a result of the geographic distribution the various stakeholders had few opportunities to meet face-to-face. Not meeting face-to-face resulted in poorly communicated and often misunderstood requirements.

4.4. Revised Product Development Cycle

The company decided to revise the product development process to try and alleviate some of the problems encountered in their existing approach. The main idea was to try and incorporate a degree of flexibility into the process.

The product manager explained

> "The whole idea of moving to a different technique was to try and get new products to the market on time and get what the customer asked for which sound like very simple things but it's very hard to do that".

4.5. The Product Realisation Cycle

The new process adopted is called the "Product Realisation Cycle" (PRC). This has two main influences, practices that worked well in the past and external consultants' assessments of the existing process. PRC has a number of distinguishing characteristics that set it apart from the old less flexible process. The PRC is summarised in Table 2.

Table 2. Summarising the "Product Realisation Cycle"

1	Customer Focus	Adopting Software QFD practices including the "Voice of the Customer" and the "House of Quality
2	Cross-Functional Core Teams	A multidisciplinary team has responsibility for and controls each project
3	Set Target Date	The target date is fixed to ensure delivery of project to customers on time
4	Face-to-Face Meetings	Stakeholders meet and agree to deliver value adding customer requirements

4.5.1. Customer Focus using Software QFD

They have adopted two Software QFD practices, namely the "Voice of the Customer" and the "House of Quality". The perception is that both of these have enabled the company to focus on the real customer requirements and prioritise these such that they can decide which value adding features should be included in each release of the software.

4.5.2. Cross Functional Core Teams

Each time a new project commences a small cross-functional core team is established. Each core team is empowered to make decisions. The team usually consists of four people, a combination drawn from Engineering, Operations, Sales, Marketing, Product Management, and Test. The Engineer and Operations person are constants on these core teams. In order to maintain customer focus a representative of either Sales or Marketing ensures that the "Voice of the Customer" is present in the team.

The product manager attributes the success of the core teams to these factors: maintaining a small team and allowing sufficient flexibility in the process, ensure commitment from the beginning and empowering the team to make decisions.

4.5.3. Setting the Target Date

The new Product Realisation Cycle forces the team to be more proactive about setting and hitting the project target date. By making the target date the goal of the team, there was less inclination to function as individual units.

Three factors are uppermost in discussions about the project plan. These are time, resources and the specification. Each project commences by fixing the time period. The team focuses on when the customers need to have the product in their hands and the project is schedule to meet that target date.

4.5.4. Face-to-Face meetings

The new product development process begins with an initial planning meeting conducted face-to-face. The core team will be in attendance and in addition there could be potentially another five people. Bringing the key stakeholders together face-to-face at this stage fosters a common understanding of the project goals. The agenda for the project plan meeting lasts two to three days and sets out the entire scope of the project.

5. Success Factors

5.1. Discussion

Independent Tools commenced the new Product Realisation Cycle (PRC) in 2003. It functions very

differently from the previous sequential process, which was perceived as too inflexible. It operates in a fashion that promotes flexibility and concurrent development of hardware and software. The PRC process is represented in Figure 3.

One of the distinguishing characteristics of the new process is the increased time allocated to product planning and justification. The company now spends more time in the initial planning stages, so that the end product reflects customer needs more precisely.

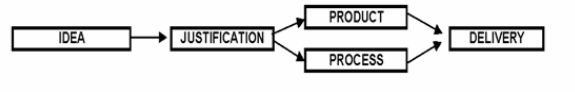


Figure 3. The PRC process

The core team members summarise the benefits of the new Product Realisation Cycle as:

- "Eye-to-eye" communications
- "Voice of the Customer" driven
- The process is tailored to each project
- The new process supports frequent software updates and facilitates achieving time-to-market.
- The "House of Quality" is agreed by all stakeholders and helps prioritise what needs to be done.

5.2. Analysis

In this section we compare the observations of this case study with effective requirements practices identified in the peer-reviewed literature. In particular we consider the findings of a different type of field study in another domain that has ramifications especially for market-oriented situations [30].

Ebert's field study compared hundreds of finished projects in a large telecoms company. He measured requirements effectiveness in terms of project completion dates. His study revealed four key product life-cycle management techniques. The study results showed that when used together, the four techniques could significantly reduce delays.

Of 246 projects evaluated, results showed that performance significantly improves if three or four of the techniques are used. However, in projects where none, one or two only are used then the impact on performance is much less.

The four techniques for better life-cycle management recommended by Ebert [30] are:

1. Install an effective core team for each product release
2. Focus the product life cycle on upstream gate reviews
3. Evaluate requirements from various perspectives
4. Assure dependable portfolio visibility and release implementation

Beginning in 2003, all projects in Independent Tools have been using all four of those techniques recently recommended by Ebert [30]. They particularly understood the need for a multifunctional core team that is fully accountable for the product's success with a clear mandate of ownership of the project.

The company use "Toll Gates" as key review dates in managing each project. Ebert calls these "Gate Reviews" and says that rather than conduct the review as a lengthy meeting, a checklist should be available beforehand so that the team can decide whether to move to the next phase. Independent Tools operate in a similar fashion and have a checklist for each toll gate enabling a "Go" or "No Go" decision very quickly. Another reason for adopting toll gates was that the company found "feature creep" to be a recurring problem in their old product development cycle. Now in the Product Realisation Cycle the "Toll Gates" alleviate this problem.

Evaluating requirements from multiple stakeholders' perspectives is implemented in Independent Tools using the Software QFD practices "Voice of the Customer" and "House of Quality". Software QFD enables the core teams to elicit requirements and justify those that are value adding and prioritise them accordingly. This approach is consistent with Ebert's third practice of evaluating requirements from various perspectives.

Finally, project plans at Independent Tools are directly linked to the requirements. This was a key recommendation of Ebert's study [30]. A key success factor in this case is having all relevant product and project information accessible online. Independent Tools utilise a number of project management tools that facilitate this. As Ebert [30] states, "A Web-based company-wide dashboard facilitates easy and standardized visibility of the project's progress and increases the stakeholders' understanding of project dependencies".

5.3. Main Conclusions

Independent Tools recognized that in order to remain competitive they had to significantly change their Product Development Cycle and that this would be a challenge not only in terms of embracing new

techniques but it involved a change in the culture of the organisation, which can be delicate to manage.

The new Product Development Cycle is still in its infancy and will continue to evolve over time. Having met the challenge of changing the culture of the organization, from being Engineering-centric to being Customer-centric, has not been easy and may have left engineering feeling marginalised. There is a feeling that the "Voice of the Engineer" can be valued alongside the "Voice of the Customer".

In contrast to field studies such as [30], our research is focused on a particular situation. The in-depth nature of the case study approach facilitates close engagement with the participants and the decision makers and allows us to study artefacts and processes at close hand which we hope will lead to a deeper understanding of the phenomenon.

5.4. Future Work

We have presented the initial findings of a study into effective requirements practices in the automotive aftermarket domain. This is only the inception stage of an extended case study. Thus far, we have not defined what we mean by effectiveness but instead have based our findings on the perceptions of the company regarding their own success.

Future work will involve identifying the factors that have an impact on the effectiveness of requirements practices and then measuring the effectiveness of such practices.

6. Acknowledgements

This research has been funded by Science Foundation Ireland through Lero - The Irish Software Engineering Research Centre (Grant no 03/CE2/1303_1) within the University of Limerick, Ireland.

The authors would also like to thank all the interviewees at Independent Tools. Thanks are also due to the anonymous reviewers for numerous helpful comments.

7. References

1. Lauesen, S., *Software Requirements Styles and Techniques*. 2002: Addison-Wesley.
2. Power, N. and T. Moynihan. *A Theory of Requirements Documentation Situated in Practice*. In *Proceedings of the 21st annual international conference on documentation*. 2003. San Francisco, USA: ACM Press.
3. Bolton, D., D. Furber, S. Green, S. Jones, and D. Till, *A Generic Modelling Approach to Requirements Capture in the Domain of Air Traffic Control*. IEE Colloquium on Software in Air Traffic Control Systems: the Future, 1992(153).
4. Cysnerios, L. *Requirements Engineering in Health Care Domain*. In *Tenth International IEEE Conference on Requirements Engineering*. 2002. Los Alamitos, California: IEEE Computer Society Press.
5. Bjorner, D., *Software Engineering 3: Domains, Requirements and Software Design*. 2006, Berlin: Springer.
6. Huemesser, N. and F. Houdek. *Experiences in Managing an Automotive Requirements Engineering Process*. In *Proceedings of the 12th IEEE International Requirements Engineering Conference*. 2004. Kyoto, Japan.
7. Hwong, B.M. and X. Song. *Tailoring the Process for Automotive Software Requirements Engineering*. In *International Automotive Requirements Engineering Workshop (AURE'06)*. 2006. Minneapolis, USA: IEEE Computer Society.
8. AuRE. *International Workshop on Automotive Requirements Engineering*. 2004 [cited 1/11/2006]; Available from: http://www.seto.nanzan-u.ac.jp/~amikio/NISE/AuRE/.
9. AuRE. *International Workshop on Automotive Requirements Engineering*. 2006 [cited 1/11/2006]; Available from: http://www.seto.nanzan-u.ac.jp/~amikio/NISE/AuRE/.
10. Autosar. *Automotive Open System Architecture* 2006 [cited 10/11/2006]; Available from: http://www.autosar.org/find02.php.
11. AAIA. *The Automotive Aftermarket Industry Association* 2006 [cited 26/09/2006]; Available from: http://www.aftermarket.org/home.asp.
12. ResearchandMarkets. *Car Aftermarket in Europe*. 2006 [cited 10/11/2006]; Available from: http://www.researchandmarkets.com/reports/328335.
13. Richardson, I., G. Coleman, N. Power, and K. Ryan. *Does Software Development come under the Total Quality Management Umbrella*. in *Pacific Northwest Software Quality Conference* 2004.

14. Sullivan, L.P., *Quality Function Deployment.* Quality Progress, 1986. 19(6): p. 39-50.
15. Madu, C.M., *House of Quality in a Minute.* Fairfield, CT: Chi Publishers, 1999.
16. Herzwurm, G., S. Schockert, and P. Wolfram. *QFD for Customer-Focused Requirements Engineering*. In *Proceedings of 11th International Requirements Engineering Conference*. 2003: IEEE.
17. Haag, S.E., M.K. Raja, and L.L. Schkade, *Quality Function Deployment: Usage in Software Development.* Communications of the ACM, 1996. 39(1): p. 41-49.
18. Davis, G., A. Geras, and C. Zannier. *Quality Function Deployment for Requirements Engineering*. 2001 20 September 2005 [cited 2006 24th August]; SENG 613: QFD Group]. Available from: http://www.guydavis.ca/seng/seng613/group/qfd.shtml#1.
19. Clausing, D. and J.R. Hauser, *The House of Quality*. Harvard Business Review, 1988. 66(3): p. 63-73.
20. ASI, *Quality Function Deployment*, A.S. Institute, Editor. 1994, ASI Press Dearborn
21. Fortuna, R., *Beyond quality: taking SPC upstream.* Quality Progress, 1988: p. 23-28.
22. Jones, C., *Software Quality: Analysis and Guidelines for Success. Software Assessments, Benchmarks, and Best Practices*. 2000: International Thomson Computer Press.
23. Haag, S.T., *A field study of the use of quality function deployment (QFD) as applied to software development*, in *University of Texas at Arlington.* 1992.
24. Young, R.R., *Effective Requirements Practices*. 2001: Addison-Wesley. 359.
25. QFD-Institute. *Frequently Asked Questions About QFD*. 2006 [cited 10/11/06]; Available from: http://www.qfdi.org/what_is_qfd/faqs_about_qfd.htm.
26. Glinz, M. *Problems and Deficiencies of UML as a Requirements Specification Language*. In *Proceedings of the 10th International Workshop on Software Specification and Design*. 2000. San Diego.
27. Maiden, N. and A. Sutcliffe. *Domain Abstractions in Requirements Engineering: An Exemplar Approach*. In *Proceedings of the 7th Conference on Knowledge-Based Software Engineering*. 1992: IEEE Computer Society Press.
28. Feather, S., S. Fickas, A. Finkelstein, and A. Van Lamswerede, *Requirements and Specification Exemplars.* Automated Software Engineering, 1997. 4(4): p. 419-438.
29. Yin, R.K., *Case Study Research Design and Methods*. Third Edition ed. Applied Social Research Methods Series. Vol. 5. 2003: Sage Publications.
30. Ebert, C., *Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques.* IEEE Software, 2006(May/June): p. 19-25.