The use of Quality-Function Deployment (QFD) for customer-focused Product Development

Author(s): Andreas Helferich, Georg Herzwurm, Sixten Schockert

Abstract
In today’s competitive business environment, it is extremely important to offer customers exactly the products they want. Mass Customization promises to enable companies to offer a large variety of products while still being able to manage the complexity caused by this increased number of products. But offering a large range of variants does not necessarily mean increased profits, as many manufacturing companies had to notice in the early 1990ies. Therefore, value-based product development methods are necessary to ascertain a product portfolio that optimally satisfies customer demands and at the same time restricts the number of products offered. This paper describes how the well-established product development and quality management method Quality Function Deployment (QFD) can be used to identify customer segments, and using the importance ascribed to certain customer requirements as input, the right product variants needed to satisfy the customer segments identified are defined.

Keywords
Quality Function Deployment, Product Portfolio, Value-based Product Development
The use of Quality-Function Deployment (QFD) for customer-focused Product Development

1 Introduction

One of the main reasons for adopting Mass Customization is the possibility of fast, economical and high quality development of a wide range of products.

But adopting a Mass Customization approach does not guarantee success: as many manufacturing companies learned in the 1990ies, offering too many products leads to substantial complexity costs, endangering profits (Kaplan, 1995). As Blecker et al. (2005) show, there are many definitions of Mass Customization. Ours is based on the notion of products composed in a largely modular way, allowing the customer to choose between a large variety of options that are based on a limited number of building blocks, thus simultaneously allowing for customer-orientation and costs’ efficiency.

Even though this procedure significantly reduces complexity as Piller (2006) demonstrated, every additional product introduces some complexity and additional cost. Therefore, carefully planning the available products, thus restricting the combinations actually offered, is of great importance.

Focus of this paper is the strategic planning of the portfolio of products resulting from Mass Customization. For this, we propose adapting the well-known Quality Management method Quality Function Deployment (QFD) for the use with Mass Customization since this method has been successfully used to identify true customer requirements in various industries (Chan & Wu, 2002), thereby providing the basis for decisions on the Product Portfolio.

While QFD has been applied in various industries and our method (called QFD-PPP, Quality Function Deployment – Product Portfolio Planning) could theoretically also be used in any industry, we have developed it with Software Products in mind, so that a few peculiarities of software have been taken into account. For example, the distinction between product function and quality element has to be made: a product function is a “functional characteristic feature of the product, usually not measurable (creates perceptible output)” (Herzwurm, Schockert & Mellis, 2000), while a quality element is a “Non-functional characteristic feature of the product, possibly measurable during development and before delivery (does not create perceptible output)” (Herzwurm et al., 2000, p. 28). Since we had Software in mind when developing the method, it is only logical that the example presented in this paper is from that domain.

The remainder of this paper is structured as follows: first, the approach called Software Product Lines is presented and briefly explained, then the importance of Product Portfolio Planning is explained. In section four then, we give a short introduction to QFD, explain how value-based product development using QFD works in general and explain our method QFD-PPP. After this, an example is given to demonstrate the application of QFD-PPP., before the conclusions wrap up the paper.
2 Software Mass Customization using Software Product Lines

There are several approaches towards Software Mass Customization, maybe the most important being Software Product Lines (Bettin, 2004). The term “Software Product Line” implies that different products of one domain (also referred to as problem space or application range, e. g. operating systems for mobile telephones or software support of the sales department) are viewed as a family and not as single products. According to the Software Engineering Institute, Software Product Lines are defined as “set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way” (Clements &Northrop 2002, p. 5). The components of a Software Product Line are the product line architecture and the individual products which are part of the product line. The product line architecture describes the individual products, their common components and - at least in outlines - the differences between the products of the family (Bosch, 2000). Different process models exist for the development process of product lines, e. g. those described in Bayer et al. (1999), Weiss and Lai (1999) or Muthig (2002). Common to them is that the product line development process is modeled along the structure of a product line. Just as the product line consists of product line architecture and product line members, the development process also consists of the process of the development of the product line architecture and the development process of product line members. The development of the product line architecture is called domain engineering and the development of product line members is called application engineering. Figure 1 shows the complete process.

![Figure 1: The Software Product Line Engineering Process](image)

2.1 Domain Engineering

Domain engineering as the first major part of Software Product Line Engineering consists of three steps: domain analysis, architectural design and domain implementation. During domain analysis, the analysis of the application scope of the product line that started with the scoping is continued and a requirements analysis is carried out for the complete product line. Common
features among and differences between the products are defined and the so-called variation points are fixed. Variation points are those system parts where the products differ from one another (Weiss & Lai, p. 20). A summary of variation points and their modeling and implementation is given in (Böckle, Knauber, Pohl and Schmid, 2004, p. 13 and p. 109).

Following domain analysis, the product line architecture is designed. The product line architecture provides the framework for reusable components. This framework describes visible properties of the components and the relations between them (Bosch, 2000). Reusable components are designed in the last step of domain engineering, during domain implementation. These components represent the base for the products of the product line. Together with test cases or scenarios, documentation and models they form the so-called core assets (Clements & Northrop, 2002).

### 2.2 Application Engineering

After Domain Engineering is finished, the members of the Software Product Line are developed in the second main part of Software Product Line Engineering called Application Engineering. During application engineering, the individual products are implemented according to the results of scoping and domain engineering. Three phases can be distinguished: system analysis, system design and system implementation.

During system analysis the requirements on the respective product gathered during domain analysis are further particularized, especially focusing on differences between variable requirements on the individual products. For every single product, those requirements are disregarded which this product does not have to fulfill. Then, the architecture of this product is derived from the product line architecture. The following steps are carried out: architecture pruning, architecture extension, conflict resolution, and architecture assessment (Bosch, 2000, p. 262). Next, product-specific components are implemented, using the possibilities of core asset varieties and all product specific components. Finally, the adapted core assets are tested and integrated into the designed product (Weiss & Lai, 1999).

### 2.3 Scoping

Preceeding domain and application engineering are a rough cost-benefit analysis and the so-called scoping (Bosch, 2000). During scoping the use of the product line or its products is planned (Böckle et al., 2004). One important aspect of this is the separation between requirements common to all products and variable requirements. Variable requirements are not demanded for all products of a product line in the same way. For example, all requirements which depend on the used hardware platform are variable requirements. In the context of enterprise applications, one example is the user interface. A user can alternatively access the system using a local client, web browser or UMTS mobile phone. According to Schmid (2003), scoping consists of three different tasks: Product Portfolio Scoping, Domain Scoping and Application Scoping. Product Portfolio Scoping deals with defining the Product Portfolio and will be called Product Portfolio Planning in the remainder of this paper.

### 3 The Importance of Product Portfolio Planning

Not only for Software Product Lines, but for all kinds of products, Product Portfolio Planning is a management activity closely associated with product development. Integrating information
about technical innovations, market demand, cultural and legal developments, Product Portfolio Planning tries to develop a portfolio of products that optimally satisfies customer demands (thereby leading to increased sales) and at the same time restricts the number of products offered (thereby reducing costs and the risk of new products “cannibalizing” old products’ sales, i.e. customers buying the new product instead of an existing one). In an advanced stage, this includes planning for several product generations, taking into account technology S-curves and technology roadmaps (Ulrich & Eppinger, 2000).

For a Mass Customized product, Product Portfolio Planning seeks to answer the following questions:

- Which combinations of modules should be offered?
- What technologies should be utilized?
- Which features/technologies should be common to all combinations?
- What should be the differences between different combinations?
- Which features of the product are most important to the customers?
- In what direction should the Mass Customized product evolve?

From a business point of view, the answers are quite easy in theory: all combinations that are necessary to satisfy the needs of the customers in the planned, profitable market segment should be offered. The common “core” consists of all features common to combinations. The differences result directly from the different needs of different customers in this market segment. And the technology used is the one best satisfying customer needs (including the need “reasonable price”). The importance and the suggested evolution can be obtained by asking your customers the right questions.

In practice, none of these answers is easy, since customer needs are not easily identified and prioritized. The latter is necessary since some customer needs are conflicting, e.g. ease of use and a multitude of functions. Kano’s Attractive Quality Model (Kano et al., 1986) provides some insight why even the customers themselves have problems stating their true needs. According to the model, customer needs can be classified into the three categories: Must-be or Basic Attributes, One-dimensional or Performance Attributes, and Attractive or Exciting Attributes. And according to Kano, only Performance attributes are voiced by the customer since he takes Basic Attributes for granted and Exciting Attributes are neither required nor expected by the customer. But nevertheless identifying and fulfilling the latter leads to great satisfaction and the willingness to pay a premium price (Sauerwein et al., 1996). Finally, it is important to notice that customer expectations change over time and today’s attractive attributes can be tomorrow’s basic attributes (Kano et al., 1986).

Thus asking (potential) customers to fill out a questionnaire is not sufficient, rather it is important to get a deep understanding of customer needs and cross-check with technological opportunities (Aasland et al., 2000). Especially breakthrough innovations would never be developed if only explicit customer demands were taken into account since they result from exciting attributes.

Quality Function Deployment can be used to answer the questions that are part of Product Portfolio Planning and overcome the problems associated with identifying customer requirements, as will be shown in the following.
4 Value-based Product Portfolio Planning using QFD

4.1 Quality Function Deployment

QFD has been developed in the Japanese manufacturing industry (Akao, 1990), but used in a large number of industries, including financial services, health services and the software industry. 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. For an in-depth description of QFD see Cohen (1995). One important feature of QFD is the separation of customer requirement (the actual need) and technical solution (the features of the product). This is done in order to assure that the final product’s features are not determined by the technically possible but by the fitness for use, i.e. the features the customers demand.

Even though QFD was not specifically designed to be used to develop software, it can easily be adapted towards software development if two differences are considered: first, the software production process is basically a duplication process and implementation is largely determined by the system design, especially the system architecture. Therefore, the effort has to be directed mainly into the earlier stages. Secondly, as Zultner (1994) put it, “Software […] is valued not for what it is, but for what it does” . Thus, the distinction between product function and quality element has to be made: a product function is a “functional characteristic feature of the product, usually not measurable (creates perceptible output)” (Herzwurm et al., 2000), while a Quality Element is a “Non-functional characteristic feature of the product, possibly measurable during development and before delivery (does not create perceptible output)” (Herzwurm et al., 2000).

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 et al., 2000). 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. 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 (Herzwurm et al., 2000)) 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.

“QFD provides a systematic but more informal way of communication between customers and developers” (Herzwurm et al., 2003), compared to traditional ways of formalizing and specifying product requirements. A project team consisting of customer representatives, developers/engineers and a moderator who is an expert in QFD works together during the whole QFD process. The software developers and/or engineers assure that the features can be implemented and that technological breakthrough innovations are not ignored, while the customer representatives assure the value-based prioritization of the development effort.

Substituting a customer survey, one of the first meetings tries to ascertain customer needs and to classify them in the Voice of the Customer Table. These requirements are structured using affinity- and tree diagrams and weighted (e.g. by pair-wise comparison or the Analytic Hierarchy Process (Saaty, 1995)) by as many members of the customer groups as possible under control of the customer representatives. The weights of the different groups are then used to calculate the average weight by calculating the average of the weights assigned by the customer groups weighted with the importance of the groups, resulting in value-based priorities for the requirements. We call this “value-based”, even though there is no financial value computed, but
it seems safe to assume that the importance ascribed to a certain requirement is strongly related to the financial value that the solution(s) to this requirement have for the customer. Target Costing does basically exactly this: using conjoint analysis, the price of the product and the weight of the different product components is determined, leading to the so-called allowable costs per component (Horvath, 1993).

If a new release of an existing product is developed, the customer representatives will evaluate them according to the level of satisfaction with the current fulfillment of the requirements (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 ineffective because customers cannot evaluate the competition’s products as well. Thus, representatives of competing products’ customers would have to be consulted for such a comparison to be effective.

The second major input is the Voice of the Engineer Table, compiled by the QFD team, among them particularly developers, that includes the potential product functions. The classic HoQ also uses measurable quality elements. These are derived from the requirements by the developers. The relationships between product functions and customer requirements in both prioritization matrices are identified together with the customer representatives. Analyzing the effects that one product function has on the other product functions leads to the roof of the HoQ (Cohen, 1995). Figure 2 displays an excerpt of a Software HoQ for an email-client including the tables of customer requirements and product functions.

![Figure 2: Software-HoQ for an email client (adapted from [14]).](image)

Figure 3 gives an overview of the whole software development process using PriFo QFD. This approach has been used to develop the Calendar function in SAP R/3© (Herzwurm et al., 2000). A variation of PriFo QFD called Continuous QFD (C-QFD) using templates and iterative development cycles has been used for electronic and mobile business systems (Herzwurm et al., 2002).
4.2 QFD and product variation

There are a few examples in literature where QFD was used to define product variants. These will be presented in the following paragraphs, before explaining why these examples fall short of realizing the full potential of applying QFD for Software Product Lines.

Hoffmann and Berger (2000) extend the House of Quality by using more than one target value per feature: they use specification classes (high, mid and low) for each product instead of one simple target value and indicate the evolution of the product using arrows (e.g. the lower class product starts with low value for feature F14, but the plan for the next generation is to improve F14 to medium value). Additionally, they include information on cost reduction potential and features offered by competitors. This approach is not suitable for a large number of products since it gets too complex. Also it is not clear how they distinguish between the needs of different customers or how they identify different customer groups.

Cheng et al. (2002) use QFD to derive a new product from an existing product platform as well as to develop a new product platform, and finally to differentiate common modules from variable module. Their approach is primarily based on checking whether a certain feature is part of the core functionality or not, and close cooperation between Marketing, Sales and Engineering. While this approach stresses the need to cross-check customer input with technological input, identification of customer groups and their needs seems to depend on Marketing. Additionally, “real” (existing and potential) customers are not included in the cross-checking process. The results of their input are being filtered by Marketing and Sales.

Hunt and Walker (2003) focus on what they call the fuzzy front-end of strategy i.e. the questions how to obtain a sustainable position in the market and which markets to operate in.
QFD to gain a deep understanding of the marketplace, identify strategic outcomes (equivalent to customer requirements) and predictive metrics (equiv. to product functions) and identify what they call natural segments, i.e. customer segments that “share the same perceptions about outcomes, and more importantly who can be expected to prefer the same products or services…” (Hunt & Walker, 2003). Interesting about this method is the way they identify and use the natural segments: the identification is done using statistical clustering methods, to focus they concentrate on those outcomes that are at the same time important and where the customer satisfaction is currently rather low (Ulwick, 2002). Positioning is then done by using the outcomes and calculated opportunity and taking into account competitors’ positions. The link to identification of common and variable customer requirements is missing here, but the focus of this paper is strategy, not product development.

Fujita et al.(2003) extend QFD with a so-called variety table, where the customer functions are further analyzed with regard to customer expectations for a high-class, a mid-market and a low-class model (small, medium and large refrigerators for the Japanese market). Thus, in the HoQ, the weights of the customer requirements are different according to the model, while the correlations between customer requirements and product functions are the same for all models. Product functions achieving a high score in fulfilling requirements that have no value for the low-class model but low scores in those requirements important for that model, are identified as variable requirements only needed for the high-class and maybe – depending on the importance of the respective requirements for the mid-market model – for the mid-market model. Additionally, they present a way to perform cost-worth analysis. This method simplifies the question of product portfolio by defining the models first (in the given example to three models, but theoretically, the number could be higher) and then assigning the necessary requirements to the models. But the underlying idea of the importance of a certain requirement depending on the customer (segment) is important.

4.3 QFD-PPP

Our approach to Product Portfolio Planning makes extensive use of QFD while at the same time introducing two new matrices. First of all, the Voice of the Customer (VoC) is collected by asking existing and potential customers about the requirements they have for the product line. Once these answers are collected, they are analyzed and sorted before asking the customers to assign priorities to all requirements. Once these priorities are assigned, customer segments are derived based on these priorities using cluster analysis. Thus, unlike in PriFo QFD, there is no weighting of customer groups as this is only necessary to come up with common priorities. Another difference to PriFo QFD is the identification of customer groups not by attributes of the customer (e.g. job title or role description) but by statistical analysis. As a result of this, the differences in importance are used to identify product variants, as will be shown.

Figure 4 shows the table of customer segments and the matrix including all customer requirements and customer segments in the upper right corner. The figure also gives an overview of this process (for reasons of clarity, classic HoQ, design-point analysis and the integration with systems design and implementation are omitted).

The next step is to bring together developers, software architects and selected customers (based on the clusters identified) to build the Software House of Quality. Explicitly including the Voice of the Engineer in the form of product functions is important to identify exciting attributes according to the Kano model, i.e. e. software characteristics that customers themselves would not
have come up with. Since a product function’s level of fulfilling a customer requirement is independent from the weight assigned to the requirement, there is only one SW-HoQ for all the members of one product line. But since the weights of the customer requirements depend on the customer segments, the weight of the product functions does so either. The Software-HoQ in Figure 1 equals the Software-HoQ for one of the customer groups (including the weights), e.g. attorneys used to dictate letters who would therefore being able to dictate emails, too.

As indicated in Figure 4, the members of the product line are identified using the simple rule one member of the product line per customer segment. Core and variable features are identified by comparing the weight of the product functions for the different customer segments. This is visualized in the second new matrix called Product Portfolio Matrix shown in the lower right corner.

The software developers and software architects perform the next step evaluating different software architectures and technologies taking into account necessary quality attributes and product functions. This is also done by using matrices (Classic HoQ for the quality attributes, Software HoQ for product functions), where the roof is intensively used to analyze the impact that different architectural or technological elements have on each other. The results of this analysis are used to decide on the software architecture and the technologies to be used for prototypes.

These prototypes are then presented to the customers, thereby demonstrating exciting features the software developers and software architects came up with and the proposed solutions to the requirements voiced by the customers. Showing all customers all prototypes, some of the customers will decide to include some features they previously hadn’t assigned value to, maybe
drop some features they requested. This discussion is based on the product functions, not the original customer requirements and their weights. Only when large changes are asked for the customer requirements will be re-evaluated.

The Product Portfolio Matrix helps prioritizing the variants. Inputs are the expected costs for the product functions and the expected revenue a product will achieve. The second depends on the size of the potential market, the products currently available on the market and the customer satisfaction with these products and the advantage the member of the product line have over these products. Ulwick’s (2002) so-called opportunity algorithm or the algorithm used in (Herzwurm et al., 2000) can be used as indicators here. Both algorithms use the importance of a feature and the customers’ satisfaction with the current solutions provided by own and competitors’ products to identify features where improvements provide a competitive advantage. A more detailed economic assessment is presented in (Schmid, 2003) and (Clements et al., 2005).

Finally, derivation of new product variants and the evolution of the Mass Customized product are facilitated, since the already existing matrices can be used as. Using the matrices as a starting point leads to reductions in both time-to-market and costs and helps achieving important goals associated with Mass Customization.

5 Example: Merchandise Information System

The following example is insofar fictitious, as the project that is the basis for this example was not targeting to develop a mass-customized merchandise information system, but to select a standard software package. But during the project, it became obvious, that the members of the retailer cooperative could actually be separated into three groups:

- The largest group (~80% of the members) has only one store and sells goods directly to consumers and companies,
- The second group has 2-4 stores each and sells goods directly to consumers and companies,
- The last group has 5+ stores each and does also wholesale, delivering the goods to other retailers by truck, in addition to selling goods directly to consumers and companies in the stores.

But there were also other features that distinguished the retailers from each other: some offered a wide variety of services, others had a lot of international customers, some sold the goods online, and larger companies placed more emphasis on reporting and forecasting.

All in all, the requirements that these companies had on the merchandise information system were different in many ways, so that many of them would have been better off if there had been a mass-customized merchandise information system, where some companies would have chosen a web shop, others the more advanced reporting and forecasting, again others the tour planning module so that their trucks are used to capacity.

Figure 5 shows the Software-HoQ for the first group (companies with one store each), the Software-HoQ for the other groups would be different only in the Weight in % (and of course the relative and absolute importance, as they are calculated using the weight in %). The customer requirements shown are actually groups of requirements, the group **Supplier-side functions** for
example comprises requirements in the areas of purchasing, disposition, goods received, and goods-receipt-based invoice verification.

Figure 6 shows the differences between the weights ascribed by the different groups.

<table>
<thead>
<tr>
<th>Customer Requirements</th>
<th>Weight in %</th>
<th>Product Functions</th>
<th>Support for collective invoices</th>
<th>Support for distributed warehouses</th>
<th>Support for tour planning</th>
<th>Support for temporary purchase prices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Administration tasks</td>
<td>30</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Corporate Management</td>
<td>25</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Inventory Management</td>
<td>18</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier-side Functions</td>
<td>16</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer-side Functions</td>
<td>11</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

| relative Importance          | 19%         | 15%               | 18%                           | 10%                              | 23%                      | 14%                                   |
| absolute Importance          | 5.94        | 4.82              | 5.76                          | 3.36                             | 7.26                     | 4.44                                  |
| Rank                         | 2           | 4                 | 3                             | 6                                | 1                        | 5                                     |

Figure 5: Software-HoQ for the first group

<table>
<thead>
<tr>
<th>Customer Requirements</th>
<th>Customer Segments (in %)</th>
<th>Customer Variant #1</th>
<th>Customer Variant #2</th>
<th>Customer Variant #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Administration Tasks</td>
<td>30.2</td>
<td>28.1</td>
<td>24.7</td>
<td></td>
</tr>
<tr>
<td>Corporate Management</td>
<td>24.8</td>
<td>24.3</td>
<td>30.1</td>
<td></td>
</tr>
<tr>
<td>Inventory Management</td>
<td>18.1</td>
<td>16.8</td>
<td>17.8</td>
<td></td>
</tr>
<tr>
<td>Supplier-side Functions</td>
<td>15.9</td>
<td>17.2</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>Customer-side Functions</td>
<td>11.3</td>
<td>13.6</td>
<td>10.1</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- : most important group of customer requirements
- : 2nd most important group of customer requirements

Figure 6: Matrix Customer Requirements x Customer Segments
The Product Portfolio Matrix in Figure 7 shows the different target values for the different product variants as well as two competing products: product variant #3, belonging to the group with five and more stores and own trucks delivering goods to other retailers and their own stores, needs support for distributed warehouses (while #1 does not, since the corresponding retailers have only one store each) as well as support for tour planning.

This matrix allows for all kinds of decisions, for example, support for collective invoices is a basic attribute according Kano’s model, i.e. needed but not an important factor for differentiation from the competition, whereas for example the support for tour planning is a variable module that not all variants need to have, even though it is quite important to customer segment #3. Whether developing and offering two versions of the Customer Relationship module is sensible, needs to be calculated using the expected development cost. Maybe all three product variants should get the better version of the module (as indicated by “fulfilment level 75%”).

<table>
<thead>
<tr>
<th>Product Functions</th>
<th>Product Variant #1</th>
<th>Product Variant #2</th>
<th>Product Variant #3</th>
<th>Competitor A</th>
<th>Competitor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for collective invoices</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Support for distributed warehouses</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Financial Accounting module</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Support for tour planning</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Support for temporary purchase prices</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Customer Relationship module</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Legend:
- ●: fulfillment level 100%
- ●: fulfillment level 75%
- ●: fulfillment level 50%
- ●: fulfillment level 25%
- ○: fulfillment level 0%

Figure 7: Product Portfolio Matrix

6 Conclusions

It has been demonstrated how QFD-PPP can be used to identify different customer segments and their needs, to derive a product portfolio (i.e. members of a product line) systematically and derive common and variable product functions including exciting requirements that the customers would not have come up with. By identifying customer and the importance that these segments ascribe to different requirements, Value-Based Product Development is made possible.

Since QFD-PPP has been developed for the use with mass-customized Software Products (using Software Product Line Engineering) in mind, there will be some changes necessary before QFD-PPP can be used for physical products like cars or shoes. Additionally, validation of this approach in industrial projects is still lacking, but the general feasibility has been demonstrated using a fictional example (based on a real-world project where standard software package was selected). Also required is further research into the clustering algorithms to be used, and the connection with Target Costing seems promising, but has not yet been sufficiently researched either.
References


