Developing Portfolios of Enterprise Applications using Software Product Lines

Andreas Helferich, Georg Herzwurm, Sixten Schockert

Universität Stuttgart,
Chair of Information Systems II (Business Software),
Breitscheidstr. 2c, 70174 Stuttgart, Germany,
{Helferich, Herzwurm, Schockert}@wi.uni-stuttgart.de

Research on Component-Oriented Business Applications has focussed on Specifications of Components and Component Markets. Another approach based on Mass Customization and Software Product Lines is presented here. Mass Customization is a strategic concept used in many industries to offer products that are customer-individual, but based on a limited number of building blocks. Thus, customer value is maximized while developing the systems is still economical. Software Product Lines are a promising approach to allow the adoption of Mass Customization for Software in general and Business Applications in particular. Unfortunately, existing literature on Software Product Lines largely ignores the importance of strategic decisions on the product portfolio. Quality Function Deployment (QFD) is a well-known and successfully used Quality Management method that can help companies identify true customer needs and the features needed to fulfil these needs. This paper demonstrates how QFD can be used for Product Portfolio Planning, thus offering potentially great benefits.

1 Introduction

Existing Enterprise Applications, most prominently Enterprise Resource Planning (ERP) systems, are often large and more or less “one size fits all” systems, requiring extensive customization efforts before they can actually be used in a company (cf. [1], [2]). One possible way to reduce this effort is to adopt and adapt a relatively new competitive strategy already applied in many manufacturing industries, but also in service industries, e.g. the insurance sector: Mass Customization. Mass Customization aims at combining the advantages of two traditionally conflicting paradigms: Mass Production and Individual Customization. Simply speaking, most manufacturing companies adopting Mass Customization try to compose their products 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 (for an in-depth treatment of Mass Customization cf. [3]). Business Components as understood by for example [4] or [5] follow a similar idea, but while their approach focuses on specifying Business Components so that they can be traded on Component Markets, Software Product Lines focus on defining a family of products and their components. Thus, the
components developed for Software Product Lines are developed to be reused within
the Product Line only. These two approaches are in a way competing with each other,
but in no way mutually exclusive: while Software Product Lines do not focus on
components being marketable or developed in different organizations, this is not
explicitly excluded. And knowledge gained in research on Business Component
Specification (e.g. in [1], [4], [6] and [7]) can be beneficial for specifying the
components used in a Software Product Line, since the company developing the
Software Product Line may decide not to develop all components itself. Planning
system families instead of single systems and thereby assuring the components will be
reused (since a fundamental part of Software Product Line Engineering is to identify
components that will be used in several or all members of the product line) is an idea
that can be beneficial to the research on Business Components. This is especially the
case as long as generally accepted standards for Business Components are lacking (cf.
[4] for the need for standards). Software Product Lines have been used in various
kinds of software systems, among them Market Information Systems [8] and
Merchandise Information Systems [9].

Focus of this paper is the strategic planning of the portfolio of products resulting
from Software Product Line Engineering. For this, we propose adapting the well-
known Quality Management method Quality Function Deployment (QFD) for the use
with Software Product Line Engineering since this method has been successfully used
to identify true customer requirements in various industries, among them software
[10], thereby providing the basis for decisions on the Product Portfolio.

Section two briefly outlines Software Product Lines and Software Product Line
Engineering, including the need for Product Portfolio Planning. A brief introduction
to QFD and the authors’ new method for Product Portfolio Planning using QFD make
up section three. Related work from the domains of Software Product Line
Engineering and Quality Function Deployment is presented in chapter four, followed
by the conclusions.

2 Software Product Line Engineering

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” (cf. [11], 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 (cf. [12]).

Different process models exist for the development process of product lines, e.g.
those described in [9], [13] or [14]. Common to them is that the product line
development process is modelled 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 application engineering. Figure 1 shows the complete process.

![Diagram of the product line development process](image)

**Fig. 1. The product line development process (modified from [15])**

### 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 (see [13], pg. 20). A summary of variation points and their modeling and implementation is given in [8] (cf. pp. 13 and pp. 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 (cf. [12]). 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 (cf. [16]).

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 focussing on differences between variable requirements on the individual products. For every single product, those requirements are disregarded which this product does not have to fulfil. 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 (cf. [12], pp. 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 (cf. [13]).

2.3 Product Portfolio Planning

Preceeding domain and application engineering are a rough cost-benefit analysis and the so-called scoping (cf. [12]). During scoping the use of the product line or its products is planned (see [8], p. 44). 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 [17], 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.

But why is Product Portfolio Planning a crucial part if Software Product Lines are to be used for Mass Customization of Enterprise applications? 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 “cannibalising” old products’ sales, i.e. customers buying the new product instead of an existing one). In an advanced stage, this
includes planning for several generations of products, taking into considerations
technology S-curves and technology roadmaps [18].

For a (software) product line, Product Portfolio Planning seeks to answer the
following questions:

- Which products should be members of the product line?
- What technologies should members of the product line utilize?
- Which features/technologies should be common to all members of the product
  line?
- What should be the differences between members of the product line?
- In what direction should the product line and its members evolve?

From a business point of view, the answers are quite easy in theory: there should
be as many different members of a product line as are necessary to satisfy the needs
of the customers in the planned, profitable market segment. The common “core”
consists of all features common to all members of the product line, 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”).

In practice, none of these answers is easy, since customer needs are not that easily
identified and prioritized. Prioritizing customer needs is necessary input for decision-
making, e.g. ease of use and a multitude of functions are conflicting customer needs.
Kano’s Attractive Quality Model [19] 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, therefore not voiced as requirements. Nevertheless
identifying and fulfilling the latter leads to great satisfaction and the willingness to
pay a premium price. [7, therefore identifying them is of great importance [20].
Finally, it is important to notice that customer expectations change over time and
today’s attractive attributes can be tomorrow’s basic attributes [19].

Thus simple market surveys are not sufficient, rather it is important to get a deep
understanding of customer needs and cross-check with technological opportunities
[21]. Especially breakthrough innovations would never be developed if only explicit
customer demands were taken into account since they result from exciting attributes.

Research on software requirements engineering has come to another conclusion:
since software is immaterial in nature, customers have big difficulties expressing their
expectations before using the final product [22].

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 for software, as will be shown in the following.
3 Product Portfolio Planning using Quality Function Deployment

3.1 Software Quality Function Deployment

“QFD provides a systematic but more informal way of communication between customers and developers” [23] 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. 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. The software developers and/or engineers assure that the features can be implemented and that technological breakthrough innovations are not ignored.

The best known instrument of QFD is the so-called House of Quality (HoQ). Generally speaking, the HoQ is the matrix1 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 [24]. QFD has been developed in the Japanese manufacturing industry [25], but 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, “Software […] is valued not for what it is, but for what it does” [26]. 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)” [27], 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)” [27]. 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 [27]. 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 [27]) 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.

The entire QFD process is carried out by a QFD team with representatives of all departments (development, quality management, marketing, sales, service etc.) and is to be extended in several team meetings by the selected typical customer representatives. 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.

1 In QFD, a matrix is basically a table contrasting two factors and their correlations, for the HoQ these are Customer Requirements and Technical Requirements.
These requirements are structured using affinity- and tree diagrams and weighted (e.g. by pair-wise comparison or the Analytic Hierarchy Process [28]) 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.

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 fulfilment 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 [24]. Figure 2 displays an excerpt of a Software HoQ for an email-client including the tables of customer requirements and product functions.

![Software HoQ Table](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© [27]. 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 [29].

### 3.2 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.

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. 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 2 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. The resulting matrix, including all customer requirements and customer segments, including the importance assigned to the requirements is shown in Figure 4.

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: product functions x members of the product line displayed in Figure 5.

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

![Fig. 4. Matrix Customer Requirements x Customer Segments](image-url)
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.

![Matrix Product Functions x Members of the Product Line](image)

The second new (matrix product functions x members of the product line) 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 so-called opportunity algorithm [30] or the algorithm used in [27] 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 [17] and [31]. Figure 6 gives an overview of this part of the process (for reasons of clarity, classic HoQ, design-point analysis and the integration with systems design and implementation are omitted).

Finally, derivation of new products for a Software Product Line and the evolution of the Software Product Lines and its members are facilitated, since the already existing matrices can be used as templates (a similar course of action for agile software development was proposed in [29]). Using the matrices as a starting point leads to reductions in both time-to-market and costs and helps achieving important goals associated with Software Product Lines.
4 Related Work

One important influence for QFD-PPP was PuLSE, a methodology developed at Fraunhofer IESE. Other Software Product Line approaches differ from PuLSE in the later stages, but their treatment of Product Portfolio Planning is similarly short, as [32] discovered while examining requirements engineering for product lines. PuLSE is briefly presented in 4.1.

Other important influences come from the QFD domain, where some reports exist of product variants being taken into account. However, the literature on QFD for product variants is scarce since QFD usually examines the development of one product and not a set of products. Nonetheless, it is possible to use QFD for product lines and existing approaches are presented in chapter 4.2.

4.1 PuLSE

PuLSE (Product Line Software Engineering) consists of several modules. Goal of PuLSE is “the conception and deployment of Software Product Lines within a large variety of enterprise contexts” (cf. [20]). Product Portfolio Planning is considered part of Product Line Scoping which is defined as “the management activity that determines in which life-cycle (…) a certain functionality will be developed.” [17]. Product Portfolio Scoping is one (and logically the first) of three kinds of scoping
The first deals with the definition of the products to be developed, i.e. definition which and how many products shall be developed and the functionality each of them shall have, but is explicitly treated as input [17]. But an activity called Product Line Mapping (PLM) is part of PuLSE. As Schmid points out, PLM is a technical activity, not a decision-making activity [17]. Nonetheless, information is provided and analyzed during PLM: genealogy charts providing a quick overview of current and future members of the product line, and the so-called product map, providing a rather detailed view on the members of the product line, their features, competitor products, and models for analyzing the economic benefits of products or domains, thus aiding in prioritizing the development efforts. QFD-PPP basically adapts the product map and explicitly includes identifying customer segments and managerial decision-making. Thus an integration of QFD-PPP into PuLSE seems natural, thus integrating QFD-PPP in a well-documented and successfully applied methodology for the development of Software Product Lines.

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 [33] 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 planned evolution of the product. 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. 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 [34]. 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 which results in their input being filtered by Marketing and Sales.

Hunt and Walker [35] 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. They use 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…” [35]. 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 [30]. Positioning is then done using the
outcomes and calculated opportunity taking into account competitors’ positions.

Fujita et al. [36] 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. 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. This results
in some product functions not being included in the low- or middle-class model since
they are not fulfilling requirements important for that model. This method simplifies
the question of product portfolio by defining the models first and then assigning the
necessary requirements to the models. But the underlying idea of the importance of a
certain requirement being dependant on the customer (segment), while the correlation
is not dependant on the customer (segment), is important.

Conclusions

It has been demonstrated how Software Product Lines and the underlying idea of
developing system families instead of single systems can be used to allow producers
of Business Applications to develop mass customized products. Also it has been
shown how QFD-PPP can be used to identify different customer groups and their
needs, to systematically derive a product portfolio (i.e. members of a product line)
and derive common and variable product functions including exciting requirements
that the customers would not have come up with on their own. Thus, Business
Applications that are highly customer-individual and at the same time cheaper to
implement, while still allowing adequate profits, seem possible.

Validation of this approach in industrial projects is still lacking, especially the
integration into process models for Software Product Line Engineering. Also required
is further research into the clustering algorithms to be used, the economic evaluation
and into the integration of the QFD results towards later phases of the Software
Product Line Engineering Process (for a method integrating QFD and object-oriented
programming see [37]). As for Software Product Line Engineering in general, tool
support is lacking.

References

   (Ed.): Tagungsband des 5. Workshops Komponentenorientierte betriebliche
2. Gomez, J.M., Krüger, O., Kühne, C., Lübke, D. and C. Rautenstrauch „Developing a
distributed ERP system based on Peer-to-Peer-Networks and Web Services“. In: Fourth
International ICSC Symposium on Engineering of Intelligent Systems (EIS’2004), Funchal
(Portugal), 2004, pp. 55-60


Developing Portfolios of Enterprise Applications using Software Product Lines