J Intell Manuf (2013) 24:1019–1031
DOI 10.1007/s10845-012-0643-8

Mass customization and personalization software development: a case study eco-design product service system

Tsai Chi Kuo

Received: 13 May 2011 / Accepted: 7 April 2012 / Published online: 24 April 2012

Abstract A large number of customers currently require manufacturers to develop more flexible products to satisfy their own personalization. Due to buyer's market and short product lifecycles, many industries are beginning to shift from mass production to mass customization to satisfy their customer requirements. The purpose of this research is to provide a model for enhancing the software component design in order to improve quality and reusability, as well as to reduce costs based on the concept of mass customization and personalization (MC&P). First, quality function deployment (QFD) was used to find out what the customers think, and to determine the core functions and components of the software. The mass customization technology was then used to aggregate the software modules. Thirdly, the software costs for business strategies, purchasing and renting under the customization model were evaluated. Finally, a real case of Eco design software in PSS was illustrated based on the model.

Keywords Mass customization · Personalization · Quality function deployment · Software renting · Product service system

Introduction

Historically, a huge gap has been found between the general purpose, horizontally oriented, broad market technologies supplied by software vendors and the specialized, vertically oriented, custom solutions developed to automate proprietary business processes. The gap might have been resulted from a mismatch between the diversity of customer requirements and economies of scale in the software industry. Every customer differs from the other in his or her demands and those needs can only be satisfied by a custom solution. However, software vendors who struggle to keep their development costs under control are compelled to respond to the customers' requirements as if they were identical (Greenfield 2007). Mass customization (MC) has been proved as an effective strategy to reduce the above gap. It conduces to standardized goods or services, but at the same time incorporates customization for the final product or service to a certain degree.

MC focuses on the means of efficiently producing and maintaining multiple similar software products, taking into account and making good use of the similarities and differences in customer demands (Krueger 2001). This is a long practiced strategy in the automotive industry. Here, the focus is on creating a single production line out of which a car model that satisfies both mass production and customization is produced.

Two popular tactics are generally used in MC for software and product development. They are modular design and delayed differentiation, respectively. Modular design, a kind of component design, is a form of standardization (Stevenson 2009). Modular could also be applied to production system that is called modular production system (Rogers and Bottaci 1997). Several other studies have proved such a component design useful in developing software systems (Kotonya et al. 2003; Brown and Wallnau 1998; Crnkovic 2001; Crnkovic and Larsson 2002). Delayed differentiation is a postponement tactic which is used to delay the completion of a product until the final demands have become clear. Differentiation of the products takes place in the final stage so as to effectively fulfill customer needs. Delayed differentiation of customers' preferences does not only achieve the differentiation of the end-products, but also helps to reduce the development costs.

T. C. Kuo (✉)
Department of Industrial and Systems Engineering, Chung Yuan Christian University, Chung Li, Taiwan, ROC
e-mail: tckuo@cycu.edu.tw

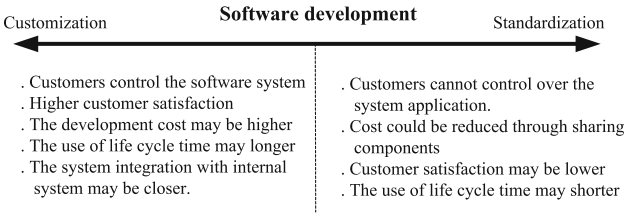

Fig. 1 The software development for standardization and customization

Although the MC technique has been widely used in software development, certain critical problems associated with it have remained unsolved. The first problem is that finding customer requirements is still difficult. Customers are always looking for highly personalized and software and thereby, meeting their unique requirements has become a challenge for the software vendors. This problem often involves quality issues. The second problem is that designing software for reuse is difficult if not impossible. Felice (1998) indicated that engineering software is usually not designed for reusability. However, software developers always seek to standardize their software to enhance reusability and reduce costs. Finally, costs may constitute another problem associating with quality and component reusability. Generally, higher software reusability indicates lower customer satisfaction and lower costs.

In the midst of the world-wide economic downturn, many enterprises are trying to reduce the average cost of their information systems (IS). Instead of purchasing software, there is a trend of enterprises pursuing after renting services on web-based platforms. Most enterprises need an affordable 'total investment' for their complex information systems. Here, the total investment includes hardware, training, installation fees, and maintenance fees. However, irrespective of enterprises' renting or purchasing services, customers still prefer a software system that not only satisfies their requirements, but also lowers their costs. Thus, developing a software system that can both satisfy customer needs and keep the development costs under control has posed a challenge to the enterprises. The above problems could be illustrated in Fig. 1.

The purpose of this research is to provide a model for enhancing the software component design in order to improve quality and reusability, as well as to reduce costs based on the concept of mass customization and personalization (MC&P). First, quality function deployment (QFD) was used to find out what the customers think, and to determine the core functions and components of the software. The MC technology was then used to aggregate the software modules. Thirdly, the software costs for business strategies, purchasing and renting under the customization model were evaluated.

The remainder of the paper has been structured as follows: "Relevant literature review" reviews relevant literature; "Model for software development based on MC&P" details the methodology for the development; "Case study and analysis" presents a description, evaluation and analysis of the two business models; and lastly, "Conclusions" gives a discussion on the research findings.

Relevant literature review

Some of the critical issues in software development encompass getting information on customer demands, using the MC technology to find the core features, and minimizing the development costs. The next three sections include a collection of relevant research. "Software development and QFD" presents a review of former studies on software development and QFD. "Mass customization and personalization for software development" details MC and personalization in software development and "Renting software service" discusses issues relevant to renting software services.

Software development and QFD

Generally, the development of a software system consists of several common, interrelated phases that include system requirement analysis, design coding, testing, deployment, maintenance and improvement (Hoch et al. 2000; Sommerville 2007). Understanding customer needs is usually time consuming, for it requires performing system analyses, developing program prototypes, as well as finalizing systems.

In order to effectively and fully understand customer needs, quality function deployment (QFD) has been used and proven to be powerful in finding core functions of all users (Barnett and Raja 1995; Tan et al. 1998; Haag et al. 1996; Jiao et al. 2007). QFD focuses on improving the quality of both the software development process and the products themselves (Shin and Kim 2000; Shen et al. 2001). It could be viewed as a tool for cost-requirement negotiation. Ramires et al. (2005) since it is understood as a matrix of correlation values between requirements and specifications. In QFD, due to the vagueness and uncertainty existing in the importance attributed to judgement of customer requirements, it is very important to capture the degree of importance of customer requirements. Kwong and Bai (2002) proposed an AHP based on fuzzy scales to determine the importance weights of customer requirements. Figure 2 illustrates the QFD process in software development.

The relationship between customer requirements and software functions are delineated by the QFD matrix. This matrix is used in the following way (Haag et al. 1996; Liu 2000):

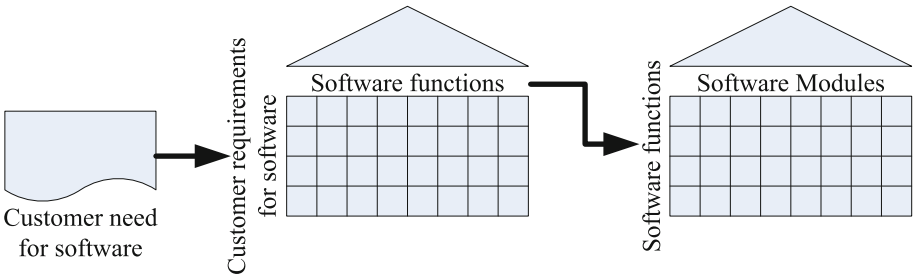

Fig. 2 QFD in software development

(1) User requirements are solicited for relevant stakeholders' use and are placed on the left-hand side;
(2) With the help from the stakeholders, the requirements are converted to technical specifications and are placed on the top;
(3) The stakeholders are then invited to complete the matrix with their perceived correlations;
(4) A list of requirements priorities is defined; and
(5) A list of technical specifications priorities is defined.

Mass customization and personalization for software development

The concept of MC emerged in the late 1980s, and it can be defined either broadly or narrowly (Silveira et al. 2001). From a broad perspective, MC was first devised by Davis (1989) and defined as the ability to provide individually designed products and services to every customer through high-process agility, flexibility, modularity, and reusability. MC technologies have reshaped the approach previously been taken in numerous industries, enabling them to perceive and adapt to latent market niches by developing technical capabilities to meet the diverse needs of target customers (Jiao and Tseng 2000; Krishnapillai and Zeid 2006). Tseng and Jiao (1996) states that the purpose of MC is to identify patterns of customer needs with product families, along with common building blocks of components, subassemblies, and modules with product fulfillment processes. Jose and Tollenaere (2005) indicate that using platforms enables essential family design savings and the ease of manufacturing. Personalization, also referred to as customization or individualization, addresses and meets customer requirements by considering individual needs (Tseng et al. 2010). Unlike customization that emphasizes product differentiation for market segments, product differentiation tends to put its focus on the individual customers. Prahalad and Ramaswamy (2000) researched on distinguishing personalization and customization. According to their research, personalization signifies the co-creation of the experience including actual interactions whereas customization refers to selecting from various existing features.

The technique of MC and personalization for software development focuses on how one can find and share the same core components among software. Generally, a software system consists of software components and software applications. Software components are units of independent design, production and deployment that interact and combine with other components to form an independently functioning software system (Heineman and Councill 2001; Szyperski 2002). A component design is a well-defined unit of software that has a published interface and can be used in conjunction with components to form larger units. Rodríguez et al. (2004) provides a framework for analyzing reusability complexity in component-based systems. They developed a cost model on component composition under the condition that the components were reused either to create a new system or modify existing ones. Moreover, software applications could be tailored to the customer and user needs and their components could be replaced or outdated without affecting the other parts of the application (Machiraju et al. 2000). Issa et al. (2006) used the case patterns to estimate reusability in software systems. The concept of product families was also used for reusability (Bosch 2000). Deelstra et al. (2005) presented a framework of terminology and concepts regarding product derivation. The most important issues are central to finding out the core features (components) shared among software and to postpone varying the components to the last stage of the development. In reference to Fig. 3, a good software system should be applicable to different kinds of software through component sharing.

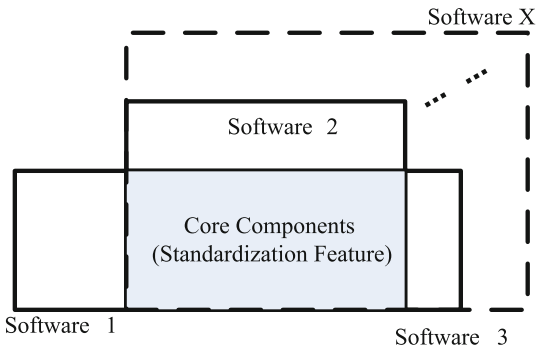

Fig. 3 Different software through components sharing

Renting software service

Business firms have been increasingly turning to application service providers (ASP) to seek solution to their information

system problems. Looking to outsource their IT departments or relevant services, many of them have chosen to rent software services from an independent entity organized to provide such services. As the economy matures, manufacturers are asked to provide more than they were asked to. That is, they have to serve their customers beyond merely the provision of material products.

In an Internet-based economy, the ASP model which allows firms to rent applications has become as an alternative to software purchasing (Tian et al. 2002; Feng 2005). In this way, many services are provided through web-based software. Helander and Jiao (2002) elaborate on the technical aspects of e-product development-enabled MC. They state that the system platform architecture would bring about the development of industrial products for integrating design, manufacturing, and logistics. For instance, HP and HP Financial Services provide a pay-per-use program which is unique in the way that it fulfills the necessary capacity requirements in real time and enables purchasing based on usage level. HP products incorporating this program include notebooks, desktops, servers, printers, and monitors. As part of the utility pricing solutions of HP, pay-per-use helps customers align their costs with their usage (signifying that customers pay less when their usage time is low), and enhance management and allocation of IT resources.

Renting software system could be viewed as a use-oriented product service system (PSS). Under such a system, the service provider retains the ownership of the tangible product of the firm and sells the functions of the product via modified distribution and payment systems. For instance, a firm may actually 'rent' the applications software under some lease-type agreement, under which it is responsible for software installation and subsequent upgrades. By virtue of such an agreement, the firm must also provide a highly skilled expert to troubleshoot computer-networking, systems-administration, and applications-deployment issues (Singh et al. 2004). To implement the use-oriented PSS, the maintenance system is the key issue. Kuo (2011) discusses the different usage concepts of products and analyzes required maintenance and service costs of rented and purchased products in different models so as to help enterprises evaluate product operation and sales models.

Model for software development based on MC&P

Figure 4 shows the QFD MC&P software development model. The model employed in the research consists of three steps. First, QFD was used to grasp the core of customer voices and to formulate modules. The next step involved constructing a system design and evaluating the quality of the modules. At the stage of the system design, component

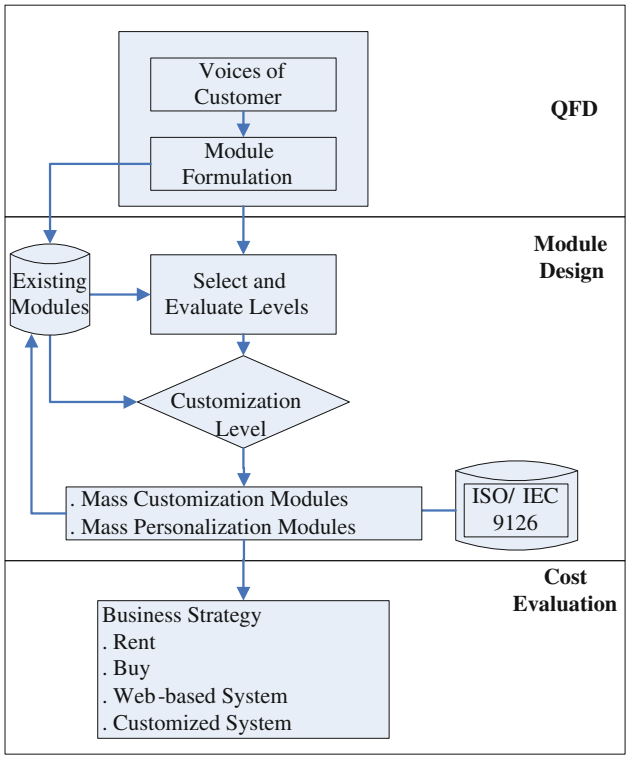

Fig. 4 MC&P model for software development

reusability was evaluated based on ISO/IEC 9126-1 (2001). The evaluation of software development took place last.

Step 1: Quality function deployment (QFD)

In QFD, to produce a customer-oriented product, it is of great importance to gather and analyze the voices of the customers. In this research, QFD was used to find the core customer requirements and transform the demands into a series of software modules.

Customer requirements come from a wide variety of sources: surveys, focus groups, interviews, trade shows, complaints, and expert opinions (Gryna 2001). Thereafter, customer needs were transformed into technical attributes. In reality, customer needs are considered and fulfilled when specified technical attributes are completed. Software QFD is a quantitative method that translates the voices of customers into relevant technical attributes.

Suppose that there are m customer requirements (CRs), denoted by $V_i, i = 1, 2, \ldots, m$. In addition, there are n software functions (SFs) denoted by $\mathrm{SF}_j, j = 1, 2, \ldots, n$. For each V_i, the weighting is evaluated by a group of q experts. Let the weighting for each V_i, be denoted as $w_{ij} (i = 1, 2, \ldots m; j = 1, 2, \ldots, q)$ and the weighting of ith CR as jth expert. Since there are q experts, the weighting values of ith CR for q experts are calculated and averaged based on the harmonic mean.

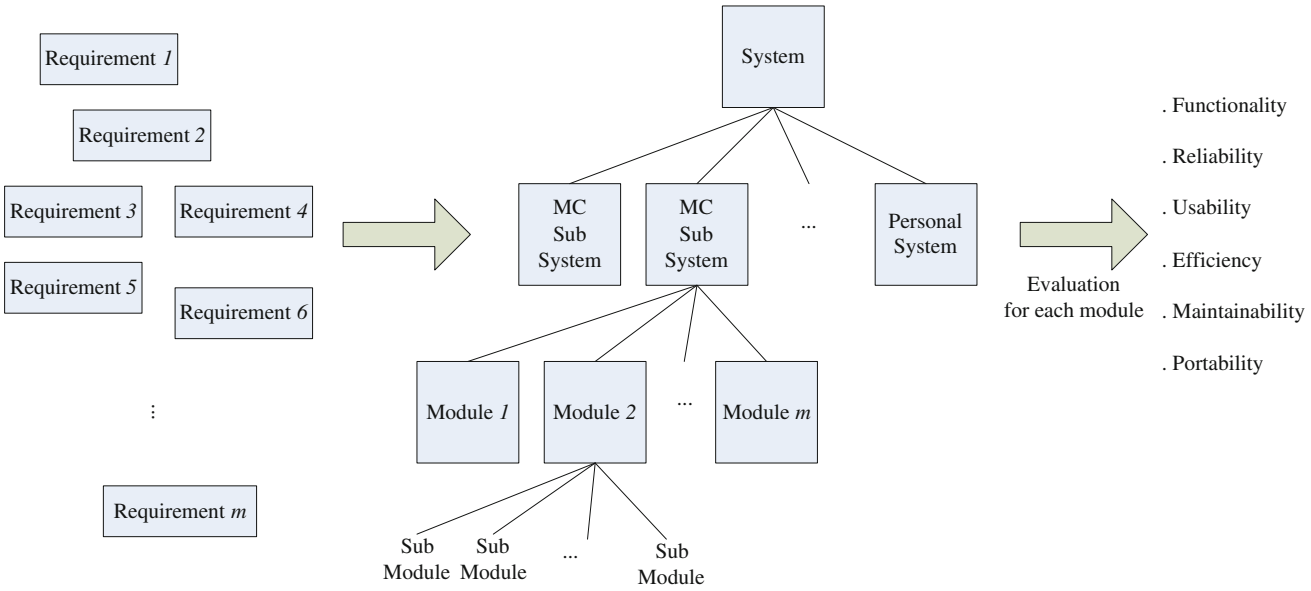

Fig. 5 To aggregate the voices of customer to be modules and sub-modules

The harmonic mean was used in the research to calculate the average of a set of numbers. Typically, it is appropriate for situations when the average of rates is desired. For each W_j,

$$W_j = \frac{q}{\sum_{i=1}^{q} \frac{1}{w_{ij}}} \tag{1}$$

Let U be the relationship matrix between CR and SF with elements U_{ij}, indicating the strength of the impact of jth SF toward fulfilling the ith CR. In the software QFD approach, U_{ij} can be quantified by a selected scale, such as 1, 3, and, 9 to denote weak, medium, and strong need relationships. Based on the construction of HoQ, U_{ij}^*, the normalized U_{ij} can be interpreted as the contribution of the jth SF toward the complete fulfillment of the ith CR when the target of the jth SF is met.

$$U_{ij}^* = \frac{U_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{k} U_{ij}} \tag{2}$$

Step 2: Modular design

After determining the software functions, software developers began to clarify the level of customer requirements and to produce sub-modules or modules. Assume there are k software modules, denoted as F_k, where $k = 1, 2, \ldots f$. For each SF_j will be quantified the relationship with F_k, denoted as V_{ik}. The scale 1, 3, and, 9 were used to denote weak, medium, and strong relationships. SF_j is then to be designed in the module F_k if V_{ik} has the maximum value. (Max $V_{ik,}$ then $\mathrm{SF}_i \subset F_k$)

In addition, when it comes to engaging with customers, it has been found that the customers are always concerned with whether or not the existent system was suitable for their companies. Reusability has been evaluated constantly to determine whether the system should be rebuilt after customer needs were known. To solve the problem, the system was divided into subsystems, modules, or sub-modules. Also, differentiation of products or service and fulfillment of customer requirements were postponed to the final stage. Figure 5 shows the divided system as modules and sub-modules. Satisfying the required variables of the customer by integrating the sub-modules is a critical task for system developers.

To ensure software quality, ISO/IEC 9126 was used to analyze each module. ISO/IEC 9126 is one of the most commonly used software development quality standards. The fundamental objective of ISO/IEC 9126 is to address some of the well known human biases that can adversely affect the delivery and perception of a software development project. These biases include changing priorities after the start of a project or not having any clear definitions of "success". Since MC&P is used to determine most customer needs, it is very important to understand customer satisfaction. ISO/IEC 9126 specifies six characteristics for the external quality of a software product. They are functionality, reliability, usability, efficiency, maintainability, and portability. The six characteristics are listed in Table 1. By using this evaluation, the software enterprise will be able to understand customer satisfaction.

For module or sub-module i, customers evaluated the six ISO/IEC 9126 characteristics on a 1–5 scale. The customer evaluations were used to determine mass customization levels or mass personalization levels.

Table 1 Six characteristics of ISO/IEC 9126-1 (2001) used to evaluate software quality

Characteristics	Sub characteristics	Description	Scale (1–5)	Compliance
Functionality	Suitability	The module/system is suitable to solve our problem		
	Accuracy	The module/system could perform the function properly		
	Interoperability	Client need to operate in different OS		
	Security	Store data can be based on user profile		
Reliability	Maturity	The module/system is ready to use		
	Fault tolerance	Complete a transaction cycle without execution errors		
	Recoverability	Use secondary server if main server fails		
Usability	Understandability	Organize items logically in the interface design		
	Learn ability	Provide online explanations of operation		
	Operability	Provide online help		
	Attractiveness	Alert the new function is available		
Efficiency	Time behavior	It must be used in the future		
	Resource utilization	The system/module must be used		
Maintain ability	Analyzability	Need a expert to operate the system/module		
	Change ability	It could be further developed		
	Stability	It allows mistakes		
	Testability	It does not need to change in the future		
Portability	Adaptability	Export to integrate with other system		
	Install ability	Need to be installed		

$$A_i = \sum_{i=1}^{6} C_i = \sum_{i=1}^{6} W_i \left(\frac{\sum_{j=1}^{n} C_{ij}}{5n} \right) \tag{3}$$

where

A_i is the mass customization level or mass personalization level
C_i: the aggregate value any ith characteristic of ISO/IEC 9126
C_{ij}: the customization level evaluation for any jth criteria in ith characteristic
W_i: the weighting for each ith characteristic, $\sum_{i=1}^{m} W_i = 1$.
n: the no of experts

After calculating A_i, A_i was compared to an upper bond (UB) and a lower bond (LB). If A_i was greater than UB, module i was added to the core system. If A_i was between UB and LB, module i was revised for MC. If A_i was less than LB, module i was revised for mass personalization.

Step 3: Cost evaluation

During cost evaluation, the software developer ($P1$) and the customer ($P2$), engage in negotiations concerning software requirements and software costs. $P1$ can choose between two options: customized software or web-based software. $P2$ can choose between two options: purchase or lease. The options for $P1$ and $P2$ are shown in Table 2.

Determination of the customization level follows that of the cost. It is assumed that costs in all systems comprise three parts:

- C_{core} : The core module of the software system. Generally, it is the fundamental part of a system. Example of them may include the algorithm, the mathematical method, the main database engine, etc.
- C_{Revised} : The modules or solutions are applicable to most enterprises. Example are the solution for one or certain industries, the input/output, or the flow processes.
- $C_{\text{Personalized}}$: Driven by the customer because of the requirements of enterprise culture and organization.

Thus, the cost function is listed as follows:

$$C^{P1}_{\text{ES\_construction}} = C^{\text{ES}}_{\text{Core}} + C^{\text{ES}}_{\text{Revised}} + C^{\text{ES}}_{\text{Personalized}} \tag{4}$$

$$C^{P1}_{\text{WBS\_construction}} = C^{\text{WBS}}_{\text{Core}} + C^{\text{WBS}}_{\text{Revised}} + C^{\text{WBS}}_{\text{Personalized}} \tag{5}$$

where

$C^{P1}_{\text{ES\_construction}}$ It is the cost for $P1$ to construct a onetime embedded system (ES) that includes costs of C_{Core}, C_{Revised}, and $C_{\text{Personalized}}$.

C^{P1}_{WBS} It is the cost for $P1$ to construct a web-based system (WBS) that includes costs of C_{Core}, C_{Revised}, and $C_{\text{Personalized}}$.

The cost equations are denoted as follows, with the notation listed in Table 2. The variables description is also detailed in Table 3.

$$P1^{\text{Buy}}_{\text{ES\_profit}} = C^{P2}_{\text{ES\_pay}} - C^{P1}_{\text{ES\_construction}} \tag{6}$$

$$P2^{\text{Buy}}_{\text{ES\_cost}} = C^{P2}_{\text{ES\_pay}} - C^{P2}_{\text{ES\_annual\_maintenance}} \tag{7}$$

$$P1^{\text{Rent}}_{\text{ES\_profit}} = C^{P2}_{\text{ES\_annual\_rent\_pay}} - C^{P1}_{\text{ES\_construction}} - C^{P1}_{\text{ES\_annual\_maintenance}} \tag{8}$$

$$P2^{\text{Rent}}_{\text{ES\_cost}} = -C^{P2}_{\text{ES\_annual\_rent\_pay}} \tag{9}$$

Table 2 Developer and customer options

$(P1)$	$(P2)$	
	Buy	Rent
Develop customized software modules	$\left(P1^{\text{Buy}}_{\text{ES\_profit}}, P2^{\text{Buy}}_{\text{ES\_cost}}\right)$	$\left(\text{P1}^{\text{Rent}}_{\text{ES\_profit}}, P2^{\text{Rent}}_{\text{ES\_cost}}\right)$
Develop a web-based platform	$\left(P1^{\text{Buy}}_{\text{WBS\_profit}}, P2^{\text{Buy}}_{\text{WBS\_cost}}\right)$	$\left(P1^{\text{Rent}}_{\text{WBS\_profit}}, P2^{\text{Rent}}_{\text{WBS\_cost}}\right)$

ES_profit: the profit to buy an embedded system (ES),
ES_cost: the cost to buy an embedded system
WBS_profit: the profit to lease a web based system, and
WBS_cost: the cost to lease a web based system

Table 3 Mathematic notations

Variables	Description	Variables	Description
$P1^{\text{Buy}}_{\text{ES\_profit}}$	Profit of $P1$ that $P2$ buys an ES	$P1^{\text{Buy}}_{\text{WBS\_profit}}$	Profit of $P1$ that $P2$ pays for WBS by times
$C^{P2}_{\text{ES\_pay}}$	Cost of P_2 to pay for an ES	$C^{P2}_{\text{WBS\_pay\_bytimes}}$	Cost that $P2$ uses WBS by times
$C^{P1}_{\text{ES\_construction}}$	Cost of $P1$ to construct the ES	$C^{P1}_{\text{WBS\_construction}}$	Cost for $P1$ to construct the WBS
$P2^{\text{Buy}}_{\text{ES\_cost}}$	The cost that $P2$ buys an embedded system	$C^{P1}_{\text{WBS\_annual\_maintenance}}$	Cost for $P1$ to maintain WBS annually
$C^{P2}_{\text{ES\_annual\_maintenance}}$	Cost of $P2$ to maintain en ES	$P2^{\text{Buy}}_{\text{WBS\_cost}}$	Cost that $P2$ uses WBS by times
$P1^{\text{Rent}}_{\text{ES\_profit}}$	Profit of $P1$ that $P2$ rent the ES annually	$P1^{\text{Buy}}_{\text{WBS\_profit}}$	Profit of $P1$ that $P2$ use WBS annually
$C^{P2}_{\text{ES\_annual\_rent\_pay}}$	Cost of $P2$ to pay annual rent	$C^{P1}_{\text{WBS\_annual\_rent}}$	Cost for $P2$ to use WBS annually
$C^{P1}_{\text{ES\_annual\_maintenance}}$	Cost of $P1$ for ES annual maintenance	$C^{P1}_{\text{WBS\_annual\_maintenance}}$	Cost of $P1$ to maintain WBS annually
$P2^{\text{Rent}}_{\text{ES\_cost}}$	Cost for $P2$ to rent the ES annually	$P2^{\text{Rent}}_{\text{WBS\_cost}}$	Cost for $P2$ to rent the WBS annually

$$P1^{\text{Buy}}_{\text{WBS\_profit}} = C^{P2}_{\text{WBS\_pay\_bytimes}} - C^{P1}_{\text{WBS\_construction}} - C^{P1}_{\text{WBS\_annual\_maintenance}} \quad (10)$$

$$P2^{\text{Rent}}_{\text{WBS\_profit}} = -C^{P2}_{\text{WBS\_pay\_bytimes}} \quad (11)$$

$$P1^{\text{Rent}}_{\text{WBS\_profit}} = C^{P2}_{\text{WBS\_annual\_rent}} - C^{P1}_{\text{WBS\_construction}} - C^{P1}_{\text{WBS\_annual\_maintenance}} \quad (12)$$

$$P2^{\text{Rent}}_{\text{WBS\_cost}} = -C^{P2}_{\text{WBS\_annual\_rent}} \quad (13)$$

Case study and analysis

Recently, many enterprises are forced to comply with many environmental regulations, such as ROHS, WEEE, and REACH directives, and so on. All these directives require the enterprises to not only introduce a green design system but also to integrate green supply chain management. To comply with these directives, some big enterprises have provided financial and technical supports to integrate their design system and manage their supply chain. However, as for small and medium sized manufactures (SMEs), they are usually hesitant when it comes to introducing a new system in the beginning. The following explains the hesitant attitude: (1) the demands are uncertain; (2) they lack adequate expertise to ensure that the problems will be effectively handled; (3) the installation fees are typically too high for them; and (4) they fear this system would bring about high maintenance costs. Since there exist many requirement uncertainties, these enterprises are faced with a dilemma of whether to buy or to rent a software system.

In this study, a recyclability evaluation software (RES) product service system (PSS) was developed for an SME. The SME wanted a RES tool to integrate eco-design capability into their design flow, to meet customer requirements. Due to its small scale, the company has used a system which was not fully applicable. Since their customers' requirements were not well defined, the company could not determine if their customers' requirements for RES PSS were temporary requirements or long-term requirements. Considering resource and cost issues, two solutions were recommended: SME could choose either to buy or to lease a software service. If the SME opts for purchase, it then needs to develop a comprehensive plan. However, if the company chooses to rent software, the company could meet their customers' requirements temporarily, but the company may need to spend more money to meet future requirement changes. Here is the flow for SME to solve the problem.

Quality function deployment

In the beginning, the software developer $(P1)$ provided a well-defined system for the customer $(P2)$ to evaluate his

or her own requirements. The voices of the customers were collected and analyzed by using the Delphi method. Then, four components were formulated as follows: ($A1$) verification of the input and output format; ($A2$) 3R DB, 3R data calculation engine; ($A3$) the analysis tool for development; and ($A4$) disassembly processes management. Table 4 shows the quality function deployment of the case study. The customer requirements (CRs) have been translated into software functions (SF).

Modular design

All the function requirements were turned into the system modules. Here, the expected function requirements and modules were evaluated by the experts. The users will determine the module could include the functions. The maximum results are shown in Table 5. For example, the function of 'material classification important interface' could either integrate with the modules of 'Verify the Input and outputs format, 3R DB', or 'The analysis tool development'. However, the integration of the function with the module, Verify the Input and outputs format, 3R DB', will be preferred because such integration yields to a higher relation value. The same criteria were also applied to other functions.

Then each module was evaluated based on ISO/IEC 9126.Then, the enterprise could determine their UB and LB percentage of MC&P level. In determining the customization level in this real case, it was assumed that UB and LB were set to 85 and 60 %, respectively, by $P2$. Of course, the higher the UB, the lower cost $P2$ is. A higher UB indicates that a solution that is accepted by most customers is being used by the SME. This will therefore result in a lower cost. A lower LB means that the SME fails to satisfy the module of the software. Figure 6 shows the evaluation results. $A2$ module was 87 %, which was greater than 85 %. Therefore, the enterprise believed the company could use the $A2$ module directly. For modules $A1$ and $A4$, the results fell between 60 and 85 %, signifying that existing modules could only be used after several revisions. The evaluation result for $A3$ module was under 60 %, meaning that the module was unsuitable for the case company. Results are shown in Fig. 6.

After the determination of the modules, the development of sub-modules $A1$ and $A4$ was evaluated. The same criteria were also used to evaluate the sub-modules, and the results are shown in Fig. 5. Because sub-modules $A1.1$, $A1.2$, $A1.3$, and $A1.6$ were evaluated to be greater than 85 %, they could be used for the core system directly. Sub-modules $A1.4$, $A1.5$, and $A1.7$ should be revised according to the enterprise flow because the evaluation fell between 60 and 85 %. According to the results, the other sub-modules $A1.8$ and $A4.2$ should be excluded for system development. Figure 7 shows the results of the evaluation.

Cost evaluation

The determination of modules and sub-modules was followed by the definition of the business strategies. Data were collected from the case company and are summarized in Table 6.

The payoff for the software developer and potential customer was also calculated according to the present value. Therefore, all income or costs should be transformed into the present value shown in Table 7.

Present value = Annual payment $*[(1+r)^n - 1]/[r*(1+r)^n]$

where n is the depreciation year (5 years in this case), and r is the annual interest rate (5 % in this instance).

In Table 7, if $P2$ chooses the 'Buy' option, $P1$ must select the "web-based platform development" option because $P2$ wants to minimize the cost, whereas $P1$ wants to maximize profit. Here is another example: if $P1$ chooses the 'develop customized software' option, $P2$ can then only select the "rent" option. However, by taking this option, $P1$ will not be able to maximize its profits. Though not achieved easily, this option is also considered an efficient solution. The potential reasons are listed as follows:

(1) $P1$ and $P2$ do not show the amount of cost to be paid and accepted in different business models.
(2) Cost ratio of renting and purchasing may be inappropriate for the model.
(3) For a web-based system, system effectiveness should take into account a large number of customers. Therefore, the renting cost for a buyer would be down, whereas renting profit for the developer would have risen.
(4) Usage time also affects the results.

The results of the case study: a web-based eco-design product service system

Table 8 describes the modules and sub-modules for the system. $A1$ and $A4$ were revised according to the customization level of the enterprises. For $A2$ module, the 3R data calculation engine was employed for all users directly.

Conclusions

Though the Internet technology has been rapidly advancing, web-based service models seem to be more suitable for the current enterprises. This new trend enables software devel-

Table 4 The voices of customer are translated to the function requirements

Customer requirement	Desire requirement										
	Weights	Regulation updates	BOM/ Material import interface	Product data&3R export interface	Material classi-fication import interface	Material No. main-tenance interface	Component No. main-tenance interface	WEEE directive main-tenance interface	Disassembly tool main-tenance interface	Report generation for authorization	3R recycling rate database
Regulation management	0.05	1	0	0				9			9
Data input & output manually	0.08	1	9	3	9	9	9	3	9		3
Data input & output automatically	0.12	1	1	1	1	1	1	3			3
Easily to maintain database	0.15	0	0								9
3R data calculation engine	0.20	0	0							3	3
Disassembly planning analysis	0.10	0	1	1	1	1	1	1	1	3	
Recycling planning analysis	0.10	0	0							3	
Disassembly report generation	0.05	1	1	1	1	1	1	1	1		
Recycling report generation	0.05	1	0								3
Security	0.10	0	0							9	
Total score		0.35	0.99	0.51	0.99	0.99	0.99	1.20	0.87	2.10	3.15
Weight (%)		1	3	2	3	3	3	4	3	7	10

Customer requirement	Desire requirement								
	Recycling rate calculation	Recovery rate calculation	Disassembly sequence analysis	Disassembly tool analysis	Disassembly time analysis	Process management	Electronic signature	System integration	Log in/out design
Regulation management		9						3	
Data input & output manually		3	3	1	9			3	3
Data input & output automatically	9	3	9	1	3			3	
Easily to maintain database	9	9		1	9	3		3	
3R data calculation engine	9	3		1	1	3		3	
Disassembly planning analysis			9	1	1	9		3	
Recycling planning analysis								3	
Disassembly report generation								3	
Recycling report generation	3	3	3			3	3	3	
Security							3	3	9
Total score	4.38	3.15	2.37	0.65	2.73	2.10	0.45	3.00	1.14
Weight (%)	14	10	7	2	9	7	1	9	4

Table 5 The modularity analysis for function requirements

	Verify the input and outputs format, 3R DB	3R data calculation engine	The analysis tool development	Disassembly processes management	Maximum
BOM/Material import interface	9[a]				9
Material classification import interface	9[a]				9
Material No. maintenance interface	9[a]				9
Component No. maintenance interface	9[a]				9
WEEE directive maintenance interface	9[a]	1			9
Disassembly tool maintenance interface	9[a]				9
Report generation for authorization				9[a]	9
3R Recycling rate database		9[a]	1		9
Recycling rate calculation		9[a]			9
Recovery rate calculation		9[a]			9
Disassembly sequence analysis		9[a]			9
Disassembly time analysis			9[a]	3	9
Process management		1	1	9[a]	9
System integration	1	3	3	9[a]	9
Log in/out design			3	9[a]	9

[a] The maximum value for each row

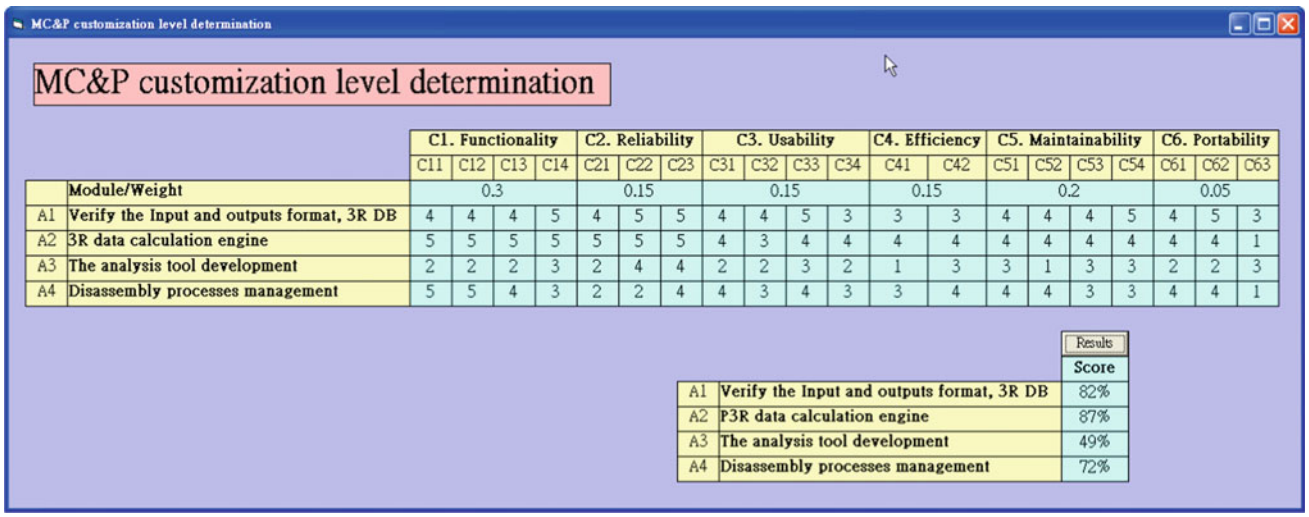

Fig. 6 The evaluation results of the customization level for four modules

opers to provide different business strategies for their customers. In addition, the current enterprises are facing more challenges in internal and external data sharing. Companies need more understanding of their customers' requirements to integrate new information systems into their customers' existing systems. Due to the customer requirements which have remained unclear, enterprises have experienced difficulties in decision making. Thus, they are faced with the question of how much resource (human power and money) they should be put into solving the problem. This study developed a QFD software development model for MC&P, used the QFD software development model to create a RES PSS tool for a SME, and showed that the QFD software development model can be used to create MC&P software products. Although the QFD model has been considered as a mature model, it is seldom used for software quality development.

Also, as it was mentioned previously, the enterprises have to consider how the customers' problems can be solved. Based on the concept of PSS, software developers have come up with two kinds of solutions for the enterprises. That is, they have to decide either to purchase or to rent a software system. This study has developed a model for a software system based on MC&P, as well as a simple and efficient configuration tool that could enable customers to choose how to develop, customize, or personalize their software. This paper also considered the business models, and purchasing and renting options for the customers, who made decisions

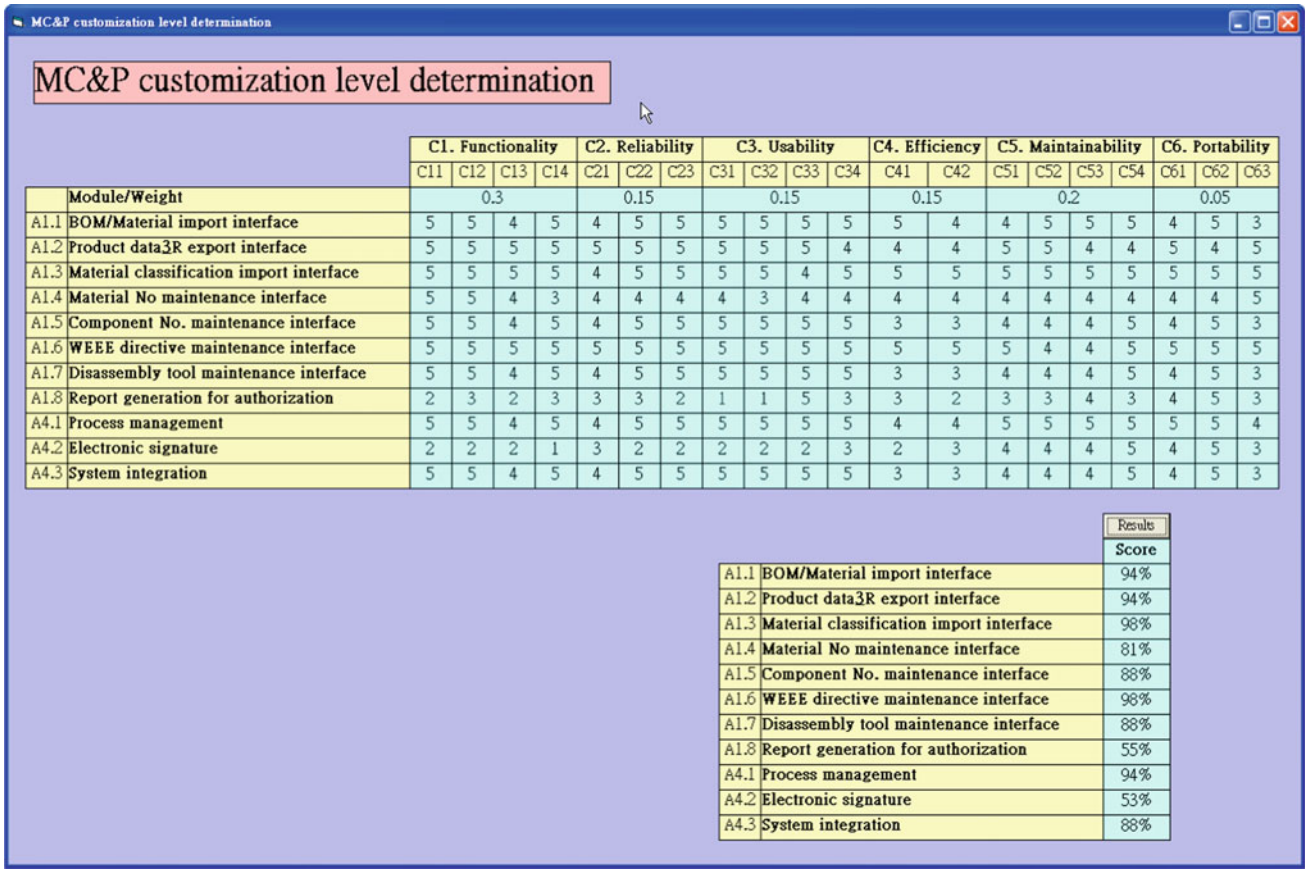

Fig. 7 The evaluation results of the customization level for sub-modules

Table 6 The value for each variable NT Dollars

Variables	Value	Variables	Value
$C^{P2}_{\text{ES\_pay}}$	1,000,000	$C^{P2}_{\text{ES\_annual\_rent\_pay}}$	250,000
$C^{P1}_{\text{ES\_construction}}$	500,000	$C^{P1}_{\text{WBS\_annual\_rent}}$	150,000
$C^{P2}_{\text{ES\_annual\_maintenance}}$	100,000	$C^{P2}_{\text{WBS\_pay\_bytimes}}$	3,000
$C^{P2}_{\text{ES\_annual\_rent\_pay}}$	250,000	$C^{P1}_{\text{WBS\_construction}}$	500,000
$C^{P1}_{\text{ES\_annual\_maintenance}}$	250,000		

Table 7 Profile and payoff for customer and software developer

$P1$ (Software dealer)	P2 (Customer)	
	Buy	Rent
Develop customized software	(500,000, −1,649,422)	(149,422, −1,082,369[a])
Develop a web-based platform	(551,735, −300,000[a])	(798,843, −3,247,108)

[a] The lower cost for different options

relevants to personalized software strategies based on their requirements and budgets.

This study can be viewed as a pilot research for customization and personalization in software development. To researchers who are interested in expanding the present work, it is suggested that they spend some time and efforts investing in strengthening this model. Elements to be considered may include the estimated sales volume of renting or purchasing, and the maintenance fee. Furthermore, as for the configuration tool, the fuzzy method can be used to determine the customization level. In terms of renting or purchasing cost strategies, it is suggested that future studies employ more sophisticated financial methods to obtain superior results.

Table 8 System modules and sub modules

	Start project
Phase I—Function requirement	Software QFD
Phase II—Module development	*A*1. Verify the input and output format, 3R DB
	*A*2. 3R data calculation engine
	*A*3. The analysis tool development
	*A*4. Disassembly processes management
	*A*1. Verify the input and output format, 3R DB
	*A*1.1 BOM/Material import interface
	*A*1.2 Product data&3R export interface
	*A*1.3 Material classification import interface
	*A*1.4 Material No maintenance interface
	*A*1.5 Component No. maintenance interface
	*A*1.6 WEEE directive maintenance interface
	*A*1.7 Disassembly tool maintenance interface
	*A*1.8 Report generation for authorization
Phase II—Subsystem requirements	A2. 3R data calculation engine
	*A*2.1 Recycling rate calculation
	*A*2.2 Recover rate calculation
	*A*3. The analysis tool development
	*A*3.1 Disassembly process generation
	*A*3.2 Disassembly tree generation
	*A*4. Disassembly processes management
	*A*4.1 Process management
	*A*4.2 Electronic signature (certification by third party)
	*A*4.3 System integration

Aknowledgments The author would like to thank the College of Electrical Engineering and Computer Science at Chung Yung Christian University for partially supporting this research under Contract No. CYCUEECS-10001 and the National Science Council of the Republic of China, Taiwan for financially supporting this research under contract NSC 99-2621-M-033-002-MY3.

References

Barnett, W. D., & Raja, M. K. (1995). Application of QFD to the software development process. *International Journal of Quality and Reliability Management, 12*(6), 24–42.

Bosch, J. (2000) *Design and use of software architectures: Adopting and evolving a product line approach*. New York, NY: Pearson Education Addison-Wesley & ACM press

Brown, A., & Wallnau, K. (1998). The current state of CBSE. *IEEE Software, 1998*, 37–46.

Crnkovic, I. (2001). Component-based software engineering: New challenges in software development. *Software Focus, 2*(4), 127–133.

Crnkovic, I., & Larsson, M. (2002). Challenges of component-based development. *Journal of Systems and Software, 61*, 201–212.

Davis, S. (1989). From future perfect: Mass customizing. *Planning Review, 17*(2), 16–21.

Deelstra, S., Sinnema, M., & Bosch, J. (2005). Product derivation in software product families: A case study. *Journal of Systems and Software, 74*(2), 173–194.

Felice, P. D. (1998). Why engineering software is not reusable: Empirical data from an experiment. *Advances in Engineering Software, 29*(2), 151–163.

Feng, S. C. (2005). Preliminary design and manufacturing planning integration using web-based intelligent agents. *Journal of Intelligent Manufacturing, 16*(4–5), 423–437.

Greenfield, J. (2007). Mass customizing solutions. *Methods & Tools, Fall, 15*(3), 27–39.

Gryna, F. M. (2001). *Quality planning and analysis: From product development through use*. New York: McGraw-Hill International Edition.

Haag, S., Raja, M., & Schkade, L. (1996). Quality function deployment usage in software development. *Communications of the ACM, 39*, 41–49.

Heineman, G. T., & Councill, W. T. (2001). *Component-based software engineering: Putting the pieces together*. Boston, MA: Addison-Wesley.

Helander, M. G., & Jiao, J. (2002). Research on e-product development (ePD) for mass customization. *Technovation, 22*(11), 717–724.

Hoch, D. J., Roeding, C. R., Purkert, G., Kindner, S. K., Ralph, M. (2000). *Secrets of software success: Management insights from 100 software firms around the world*. Boston

ISO/IEC 9126-1. (2001). Software engineering—product quality—Part 1: Quality model.

Issa, A., Odeh, M., Coward, D. (2006), Using case patterns to estimate reusability in software systems. *Information and Software Technology, 48*(9),836–845, Publisher: Elsevier.

Jiao, J., & Tseng, M. M. (2000). Understanding product family for mass customization by developing commonality indices. *Journal of Engineering Design, 11*, 225–243.

Jiao, J., Simpson, T. W., & Siddique, Z. (2007). Product family design and platform based product development: A state-of-the-art review. *Journal of Intelligent Manufacturing, 18*(1), 5–29.

Jose, A., & Tollenaere, M. (2005). Modular and platform methods for product family design: Literature analysis. *Journal of Intelligent Manufacturing, 16*(3), 371–390.

Kotonya, G., Sommerville, I., & Hall, S. (2003). Towards a classification model for component based software engineering. In: *Proceedings of the 29th Euromicro Conference* (Vol. 43). Washington, DC: IEEE computer society.

Krishnapillai, R., & Zeid, A. (2006). Mapping product design specification for mass customization. *Journal of Intelligent Manufacturing, 17*(1), 29–43.

Krueger, C. W. (2001). *Software mass customization*. Austin: BigLever Software, Inc.

Kuo, T. C. (2011). Simulation of purchase or rental decision-making based on product service system. *The International Journal of Advanced Manufacturing Technology, 52*(9–12), 1239–1249.

Kwong, C. K., & Bai, H. (2002). A fuzzy AHP approach to the determination of importance weights of customer requirements in quality function deployment. *Journal of Intelligent Manufacturing, 13*(5), 367–377.

Liu, X. F. (2000). Software quality function deployment. *IEEE Potentials, 19*(5), 14–16.

Machiraju, V., Dkkhil, M., Griss, M., & Wurste, K. (2000). *E-service, management requirements*. Alto, CA: HP Laboratories, Palo.

Prahalad, C. K., & Ramaswamy, V. (2000). Co-opting customer competence. *Harvard Business Review, 78*(1), 79–87.

Ramires, J., Antunes, P., & Respício, A. (2005). Software requirements negotiation using the software quality function deployment. *Lecture Notes in Computer Science, 3706*, 308–324.

Rodríguez, I., Núñez, M., & Rubio, F. (2004). A formal framework for analyzing reusability complexity in component-based systems. *Information and Software Technology, 46*(12), 791–804.

Rogers, G. G., & Bottaci, L. (1997). Modular production systems: A new manufacturing paradigm. *Journal of Intelligent Manufacturing, 8*(2), 147–156.

Shen, X. X., Tan, K. C., & Xie, M. (2001). implementation of quality function deployment based on linguistic data. *Journal of Intelligent Manufacturing, 12*(1), 65–75.

Shin, J. S., & Kim, K.-J. (2000). Complexity reduction of a design problem in QFD using decomposition. *Journal of Intelligent Manufacturing, 11*(4), 339–354.

Silveira, G. D., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International Journal of Production Economics, 72*, 1–13.

Singh, C., Shelorb, R., & Jiangc, J. (2004). Gary Kleind, Rental software valuation in IT investment decisions. *Decision Support Systems, 38*, 115–130.

Sommerville, I. (2007). *Software engineering, 8th edn*. New York, NY: Adison-Wesley Longman Publishing CO..

Stevenson, W. L. (2009). *Operations management*. Boston: McGraw-Hill Irwin.

Szyperski, C. (2002). *Component software: Beyond object-oriented programming, 2nd edn*. Boston, MA: Addision-Wesley Professional.

Tan, K. C., Xie, M., & Chia, E. (1998). Quality function deployment and its use in designing information technology systems. *International Journal of Quality and Reliability Management, 15*(6), 634–645.

Tian, G. Y., Yin, G., & Taylor, D. (2002). Internet-based manufacturing: A review and a new infrastructure for distributed intelligent manufacturing. *Journal of Intelligent Manufacturing, 13*(5), 323–338.

Tseng, M., & Jiao, J. (1996). Design for mass customization. *CIRP Annals-Manufacturing Technology, 45*(1), 153–156.

Tseng, M. M., Jiao, R.-J., & Wang, C. (2010). Design for mass personalization. *CIRP Annals-Manufacturing Technology, 59*, 175–178.