Expert Systems with Applications 37 (2010) 3415–3426

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Fuzzy quality function deployment based methodology for acquiring enterprise software selection requirements

Ceyda Güngör Şen *, Hayri Baraçlı

Department of Industrial Engineering, Yildiz Technical University, Yildiz, Istanbul 34349, Turkey

ARTICLE INFO

Keywords:
Software selection process
Software requirements analysis
Non-functional requirements
Fuzzy QFD
ERP

ABSTRACT

In many software acquisition methods, functional software requirements are resolved, but non-functional requirements are more or less deliberately put aside. A large body of research exists on the necessity of handling specific non-functional requirements as major drivers in the software development process. However, prior research does not provide adequate support for managing non-functional requirements in the software selection process, and suggests a unique technique and methodology for identifying the selection criteria. This paper presents a fuzzy quality function deployment approach for determining which of the non-functional requirements reported by earlier studies are important to a company's software selection decision based on and integrated with its functional requirements. The solution provided in this study not only assists decision makers in acquiring software requirements and defining selection criteria, but also supports determining the relative importance of these criteria. An actual case in Audio Electronics of Turkey's electronic industry demonstrates the feasibility of applying the proposed framework in practice.

1. Introduction

During the acquisition of large-scale software systems, effective and efficient management of user requirements is one of the most crucial issues (Karlsson, 1997). Software that lacks the appropriate functionality (what the system does) and non-functionality (how the system behaves with respect to some observable attributes like performance, reliability, etc.) carries the risk of failing to meet user needs. In many software acquisition methods functional requirements are resolved, but non-functional requirements (NFRs) are more or less deliberately put aside (Karlsson, 1997). Representing functions in terms of inputs, processes and outputs may be relatively straightforward. However, representing and modelling NFRs such as usability or reliability may be less straightforward, since such requirements are often inherently problematic.

NFRs play an important role for software architecture decisions because the selection of the main architecture structure is mainly driven by these requirements (Kruchten, 2003). There exist a lot of software architecture design methods discussing the necessity of handling specific NFRs as major drivers (Hofmeister et al., 2007). On the other hand, an increasing trend of purchasing enterprise software packages (e.g. enterprise resource planning (ERP) systems, customer relationship management (CRM) systems, and supply chain management (SCM) systems) has also highlighted the need to take NFRs seriously in the software selection process. In this situation, requirements need to be elicited from the end-user for commercially available software, and issues including the representation of the requirements, their prioritisation, level of detail, and the relation to functional requirements become more complex. Since the end-user has no influence on the functionality provided by enterprise software packages, there is little use in providing detailed functional requirements. It is more important to identify the constraints that software must meet and the overall quality of the product.

Much research has addressed the different sets of NFRs (Erol & Ferrell, 2003; Illa, Franch, & Pastor, 2000; Kunda & Brooks, 1999; Ochs, Pfahl, Chrobok, & Nothhelfer-Kolb, 2000; Standards 9126 (1991); Wei, Chien, & Wang, 2005; Wei & Wang, 2004), what criteria are used in the software selection process, and also what criteria are the most important for firms (Baki & Çakar, 2005; Bernroider & Koch, 2001; Chau, 1995; Keil & Tiwana, 2006; Kontio et al., 1995; Maiden & Ncube A.M., 1999; Montazemi, Cameron, & Gupta, 1996). However, these empirical investigations do not provide adequate support for managing NFRs, and suggest a unique technique or methodology for identifying the selection criteria in the software selection process. We have recently developed an integrated decision support system dealing with qualitative and quantitative criteria for enterprise software selection (Şen, Baraçlı, Şen, & Başlıgil, 2009). In this paper, we focus on the second step of this six steps methodology, identifying the criteria for enterprise software selection. The first goal of this paper is to identify possible

* Corresponding author. Tel.: +90 212 3832873; fax: +90 212 2585928.
E-mail address: cgungor@yildiz.edu.tr (C.G. Şen).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.10.006

NFRs reported by earlier studies. The second goal is to present a systematic procedure for determining which of these are important to a company's software selection decision in accordance with its functional requirements. We develop a fuzzy quality function deployment (QFD) approach that focuses on translating functional requirements formed with linguistic variables into non-functional criteria. The solution proposed in this study not only assists decision makers in defining non-functional selection criteria on the basis of company's own conditionality, but also supports determining the relative importance of these criteria. Consequently, the paper presents the end-user's perspective rather than that of organizations that are developing enterprise software packages (suppliers).

The rest of the paper is organized as follows: Section 2 describes briefly the QFD process and its use in software requirements analysis. In Section 3 we present, in detail, the proposed fuzzy QFD approach. An actual case in Audio Electronics of Turkey's electronic industry is described to demonstrate the proposed approach in practice.

2. Using QFD in software requirements analysis

QFD is a comprehensive quality tool specifically aimed at satisfying customers' requirements. It was originated in the late 1960s to early 1970s, in Japan, by Prof. Yoji Akao. The QFD process involves four phases: (1) product planning: house of quality (2) product design (3) process planning, and (4) process control. A matrix represents each phase of the QFD process. These matrices explicitly relate the data produced in one stage of the process to the decisions that must be made at next process stage (Akao, 1990). In the house of quality matrix, the customer's requirements in their own words (WHATs) as well as various stages of product planning, engineering, and manufacturing are determined and translated into design requirements (HOWs) or proposed performance characteristics of the product (Vanegas & Labib, 2001). The majority of QFD applications stop with the completion of this first matrix. Many companies, such as Volvo, have found that a great deal of benefit can be achieved from just completing the first matrix (Han, Chen, Ebrahimpour, & Sodhi, 2001). Similarly, Cox (Cox, 1992) indicates that no more than 5% of companies go beyond the first matrix. For this reason, in this study we only focus on the first matrix of the QFD structure. However, the proposed approach may be easily extended to the remaining QFD stages.

Companies attempting to implement QFD have reported a variety of benefits and also problems with the method. Several attempts have been made to overcome the difficulties in carrying out the QFD process. One of these trends is the application of the fuzzy set theory to determine priorities of WHATs and HOWs in QFD (Bahrami, 1994; Kho & Ho, 1996; Masud & Dean, 1993; Sohn & Choi, 2001; Vanegas & Labib, 2001). The goal of applying fuzzy set theory to QFD is to translate vague and imprecise customer inputs into exact data. An overview of the particular fuzzy-logic applications in QFD is given in Chan and Wu (2002).

A requirements analysis of a software system is often considered to be one of the most crucial steps in the software acquisition process (Liu, 1998), in which statements describing the functions and characteristics of the forthcoming software system should be developed and agreed upon. In order to achieve high user-satisfaction and high feasibility, the software requirements should be carefully specified and analyzed. As the basis of QFD is to obtain and translate the requirements of a customer into a set of detailed design specifications, it is one of the techniques that can overcome problems that occur in the software requirements analysis process (Barnett & Raja, 1995). The software requirements are divided into two separate categories: functional and non-functional (Karlsson, 1997). The functional requirements are the core of the statement, describing the functions of the software system that are expected by the stakeholders. Functional requirements typically describe the relationships between all valid (and invalid) inputs to the software system and the corresponding outputs of the software system. Traditionally, features of a system that are not covered by its functional description have been called NFRs (Bosch & Molin, 1999; Buschmann, Meunier, Rohnet, Sommerland, & Stal, 1996; Karlsson, 1997). These requirements are notorious for being difficult to elicit, express, quantify and test. They may contradict as well as complement each other. The interdependencies and trade-offs that exist between them should be considered, and a preference of one requirement over another in case of conflict should be defined (Liu, 1998; Mylopoulos, Chung, & Nixon, 1992). Especially in NFRs analysis, QFD is a widely used technique that helps to establish the relative importance between attributes and their values (Bot, Lung, & Farrell, 1996; Duenas, De Oliveira, & De La Puente, 1998).

The custom environment allows the user to specify his/her requirements prior to development, and the software is then developed to meet those needs. The usefulness of QFD in software development has been proven (Barnett & Raja, 1995; Büyüközkan & Feyzioğlu, 2005; Haag, Raja, & Schkade, 1996; Herzwurm & Schockert, 2003; Karlsson, 1997; Liu, 2001; Ramires, Antunes, & Respício, 2005). On the other hand, in the package environment, software acquisition is fundamentally different in that the user must select from an array of packages which have already been developed, with the goal of finding one that most closely meets his/her needs (Keil & Tiwana, 2006). The role of NFRs becomes more important in the software selection process because software packages already have their functionality built-in. Despite many and frequent successful applications of QFD in custom environments, few studies have been conducted on the usability of QFD in the selection of large-scale software systems. Erol and Ferrell (2003) use fuzzy QFD to convert qualitative information into quantitative parameters and then combine this data with other quantitative data to parameterize a multi-objective mathematical programming model. Their methodology was applied to a simplified version of the ERP purchasing problem. However, their methodology does not assist decision makers in how to obtain qualitative information from the end-user and how to prioritise NFRs. With the proposed fuzzy QFD approach in this study, we go one step back in Erol and Ferrell's methodology and focus more clearly on the functional and non-functional user requirements. We illustrate that QFD is also useful in the software selection process for managing NFRs. Section 3 is devoted to the details of the proposed approach.

3. The proposed fuzzy quality function deployment (QFD) methodology

A fuzzy QFD-based methodology, which consists of seven steps, is proposed to determine enterprise software selection criteria through the use of fuzzy set theory and a random experiment-based decision making approach. This methodology is presented step by step below and illustrated in Fig. 1.

3.1. Form a cross-functional team and conduct BPR

Software selection criteria formulation affects several functions in the organization; therefore, such a decision should be made according to the consensus of a cross-functional team of decision makers with various points of views and who represent different services of the company. Hence, the first step of our approach is to form a team that consists of all related managers and supervisors of a company, such as a general manager, production manager,