Contents lists available at ScienceDirect
Information Systems
journal homepage: www.elsevier.com/locate/is

Characterization of trade-off preferences between non-functional properties

Ulrik Franke[a,*], Federico Ciccozzi[b]

[a] RISE SICS – Swedish Institute of Computer Science, SE-164 29 Kista, Sweden
[b] School of Innovation, Design and Engineering, Mälardalen University, SE-721 23 Västerås, Sweden

A R T I C L E I N F O

Article history:
Received 15 December 2016
Revised 1 June 2017
Accepted 18 July 2017
Available online xxx

Keywords:
Non-functional properties
Decision-making
Trade-offs
Utility functions

A B S T R A C T

Efficient design and evolution of complex software intensive systems rely on the ability to make informed decisions as early as possible in the life cycle. Such informed decisions should take both the intended functional and non-functional properties into account. Especially regarding the latter, it is both necessary to be able to predict properties and to prioritize them according to well-defined criteria. In this paper we focus on the latter problem, that is to say how to make trade-offs between non-functional properties of software intensive systems. We provide an approach based on the elicitation of utility functions from stake-holders and subsequent checks for consistency among these functions. The approach is exploitable through an easy-to-use GUI, which is also presented. Moreover, we describe the setup and the outcome of our two-fold validation based on exploratory elicitations with students and practitioners.

1. Introduction

Software is ubiquitous in our society and most companies in any applicative domain rely on IT for their operations. Digitization and automation are no longer competitive advantages by themselves. Instead, as IT is becoming an irreplaceable asset, proper IT used as a cornerstone of operational excellence is simply essential and expected to be there. A consequence is that decision-makers in any domain face crucial decisions regarding the evolution of their IT portfolios: What should be bought off the shelf? What should be subscribed to as a service? What can be found in open-source communities? What, if anything, should be developed in-house? And, perhaps most importantly from an architectural perspective, how should all these diverse IT components fit together?

This challenge is faced by companies in essentially any domain, from the automotive company deciding on which software to put in the next generation car, to the SCADA system designer outlining the new control system for a power grid or the financial service provider rolling out a new payment system architecture. They all share two wishes: (1) to be able to select the best components throughout their architectures, and (2) to do it in the early phases, before all the details of their intended systems are actually known, in order to limit costs. Indeed, the cost of extracting defects grows as a project progresses and the products are developed – it is in fact much less expensive to correct errors in the concept or design phases, whereas that cost can grow exponentially if corrections are delayed to production and testing phases [1]. Thus, the ability to make informed decisions based on sound reasoning early on in the life cycle is pivotal.

Needless to say, though, it is very difficult to select IT components from several different alternatives, when these alternatives are still on the drawing board. One part of the problem is the estimation of the non-functional properties (hereafter simply "properties") of the future component. How secure will it be? How reliable? How maintainable? This is a classic set of topics that are very interesting in their own right. In this paper, however, we focus on another problem, which remains even when perfect property estimates are achieved: how to make enlightened *trade-offs* between non-functional properties.

To make this problem more concrete, consider the software product quality model defined in the ISO/IEC 25010 standard [2]. According to this standard, system/software product quality consists of eight properties: (i) functional suitability, (ii) performance efficiency, (iii) compatibility, (iv) usability, (v) reliability, (vi) security, (vii) maintainability and (viii) portability. Assuming for the sake of the argument that the estimates problem is solved (which it most certainly is not), this means that each alternative software product – each option on the decision maker's table – can be characterized by an 8-dimensional vector. Also assuming that the properties can all be measured and mapped onto a scale of, say, 0–10, the problem becomes one of selecting between alternatives

* Corresponding author.
E-mail addresses: ulrik.franke@ri.se (U. Franke), federico.ciccozzi@mdh.se (F. Ciccozzi).

http://dx.doi.org/10.1016/j.is.2017.07.004
0306-4379/© 2017 Elsevier Ltd. All rights reserved.

of the form $A = (10, 10, 2, 10, 10, 5, 8, 1)$, $B = (4, 9, 8, 10, 7, 0, 8, 9)$, $C = (7, 8, 7, 4, 7, 2, 7, 0)$.

This is a complex problem. Let us consider some of the possible trade-off choice scenarios. By simply considering an unweighted mean of all properties, A is the best. On the other hand, if portability (last) is the property to maximize, B is the best; if the sum of functional suitability (first) and compatibility (third) shall be maximized, then C is the best.

Only when there is dominance, i.e. one alternative being at least as good as the others in each dimension, and strictly better in at least one, the choice becomes trivial; unfortunately this is not the common case. Although difficult, these choices are pivotal for efficient development and good quality of the resulting product. This paper focuses on trade-offs by providing an approach based on the elicitation of utility functions from stake-holders and subsequent checks for consistency among these functions.

This paper is based on a previous conference publication [3]. While most of the theoretical contents are kept from our conference publication, for this paper we ran a set of empirical elicitations with students and practitioners, which are reported in Section 7. This represents the main original contribution of this paper. Additionally, a few conceptual clarifications and modifications have been made in the theoretical chapters, and the concluding discussion has been updated to reflect the empirical results.

The remainder of this paper is organized as follows. Section 2 briefly reviews some related work in order to put the contribution in context. It is followed by Section 3 which introduces some key concepts, needed to understand the rest of the paper. In Section 4, we introduce the elicitation of preferences with regard to non-functional properties, and in Section 5 we discuss how to ensure the consistency of the preferences thus elicited. Section 6 illustrates the framework devised with an example. Section 7 reports the setup and outcomes of a two-fold empirical elicitation. Section 8 discusses the contribution and Section 9 concludes the paper with a substantial discussion of future work.

2. Related work

There is an abundant literature on decision-making when developing or selecting IT components and services. An early example is King and Schrems' discussion of cost-benefit analysis in developing and operating information systems [4]. From our perspective it is interesting to note that they list five important non-functional properties which have to be taken into account: accuracy, response time, security, reliability, and flexibility. Interestingly, important parts of ISO/IEC 25010 were thus known already in the late 70-ies.

One particular problem that has attracted a lot of attention is the dilemma of in-house development vs. buying commercial off-the-shelf (COTS) products. The problem of identifying appropriate software engineering metrics for evaluating COTS has been studied for a long time [5], as has the problem of setting requirements on such metrics [6]. The actual decision-making is often done using optimization approaches [7,8], in particular when the trade-off is between two properties such as cost and reliability [9].

However, in general these problems are multi-dimensional, as explained in Section 1, and many studies indeed treat them as such. For example, one approach to solve such multi-criteria problems is to prioritize between the objectives in order to resolve inconsistencies, and then solve the resulting problem algorithmically [10]. Another widely used approach is to apply the analytic hierarchy process (AHP) to decompose the problem into sub-problems and resolve differences between stakeholders [11,12]. The kind of analysis most closely related to ours is Pareto analysis, i.e. identifying alternatives that are not dominated by any other alternatives, and then selecting solutions from this so called Pareto front. For example, Neubauer and Stummer first determine Pareto-efficient alternatives and then let the user interactively explore the solution space to find the desired solution [13]. Michanan et al. apply similar analysis to the trade-off problem between power consumption and performance, using actual live performance data [14].

This paper is similar to much of the existing literature in that it takes the multi-dimensionality of the problem seriously, and in that it aims to involve the stakeholders to elicit important information to solve the problem. In particular, it can be seen as an off-shot from the Pareto analysis strand. It differs from the existing literature in that it attempts to discuss the problem of trade-offs between several non-functional properties systematically based on canonical utility functions from the microeconomic literature, allowing for complications like diminishing marginal utility in a way not captured by e.g. AHP or cumulative voting. Österlind et al. have worked in this direction previously [15], but whereas they require the user to manually enter the parameters of utility functions, a core idea in our paper is to elicit these in a user-friendly manner, so that relatively powerful utility models can be built from relatively straight-forward user input.

3. Preference and utility modeling

The preliminaries introduced in this section are standard. A good textbook dealing with these concepts is Varian [16].

Preferences over bundles of goods (or, in our case, non-functional properties of one good – a software system) are comparisons between vectors. $\mathbf{x} \succeq \mathbf{y}$ means that the decision-maker thinks that the bundle $\mathbf{x}$ is at least as good as the bundle $\mathbf{y}$. For the preference relation $\succeq$ to *order* the bundles, it needs to be complete (apply to all $\mathbf{x}$ and $\mathbf{y}$ in the alternatives set X), reflexive ($\mathbf{x} \succeq \mathbf{x}$), and transitive ($\mathbf{x} \succeq \mathbf{y}$ & $\mathbf{y} \succeq \mathbf{z} \Rightarrow \mathbf{x} \succeq \mathbf{z}$). The strict preference $\mathbf{x} \succ \mathbf{y}$ can then be defined to mean not $\mathbf{y} \succeq \mathbf{x}$.

Any preference order that is complete, reflexive, transitive, and continuous (i.e. the preference order is preserved in the limit of a sequence of goods) can be represented by a continuous utility function, i.e. a function $u : X \rightarrow \mathbb{R}$ such that $\mathbf{x} \succ \mathbf{y}$ if and only if $u(\mathbf{x}) > u(\mathbf{y})$. Such functions are convenient to use in modeling and analysis of preferences. However, the assumptions do not always hold. For example, intransitive preferences are readily found experimentally [17].

There are several utility functions proposed in the literature. In the following, we introduce three of the most common, which are all special cases of the more general constant elasticity of substitution (CES) utility function.

One very simple utility function is the following:

$$u(\mathbf{x}) = \mathbf{a}^T \mathbf{x} \tag{1}$$

Here, the utility of the bundle $\mathbf{x}$ of the n goods $x_1, x_2, \ldots, x_n$ is just the sum of these goods, weighted by the coefficients $\mathbf{a} = a_1, a_2, \ldots, a_n$. Under such preferences, the goods are *perfect substitutes*, i.e. the decision-maker is willing to switch between goods at a *fixed ratio*, viz. is indifferent between one unit of x_1 and $\frac{a_1}{a_2}$ units of x_2. Backup tape cartridges of 6 TB from brand A and 3 TB from brand B are a good example of (nearly) perfect substitutes, with decision-makers willing to switch one A for 2 B (if they are similar with respect to e.g. failure rates).

Another very simple utility model is the following:

$$u(\mathbf{x}) = \min\{a_1 x_1, a_2 x_2, \ldots, a_n x_n\} \tag{2}$$

Here, the utility of the bundle $\mathbf{x}$ is the smallest x_i as weighted by a_i. This is called Leontief preferences and the goods are *perfect complements*. Such goods have to be consumed together, so additional units of one good without simultaneous increases in all the others are no better. For a personal computer, a decision-maker