Market-aware Requirements*

Romina Torres and Hernan Astudillo

Universidad Técnica Federico Santa María,
Departamento de Informática, Valparaíso, Chile,
{romina,hernan}@inf.utfsm.cl,
http://www.inf.utfsm.cl

Abstract. Traditionally, non-functional requirements (NFRs) are specified as measurable entities to permit evaluation satisfaction; however, NFR specifications quickly become obsolete because (1) NFRs are expressed in numbers, (2) architects specify them using the correct values at design time, and/or (3) providers are constantly improving their offer, in terms of functionality and quality of service (QoS). The computing-with-words approach has already been proposed to replace numerical NFR specifications, where natural language words denote fuzzy quality levels; unfortunately, current proposals provide only for design-time, stakeholder-defined translation of words as numerical ranges. We propose a mechanism to automatically and dynamically determine current numerical ranges of the fuzzy quality levels from the available data, without human intervention, whenever changes to component QoS specifications. Our main contribution is allowing architects to specify their requirements using words only once (at design time), and whenever providers change components QoS characteristics, automatically update those requirements to the new market view, enabling market-aware requirements. The approach was validated by measuring the number of times that necessarily a requirement had to be rewritten at runtime in order to get new operationalizations which replace the now older ones. We use a set of ten complex requirements, a dataset of 1500 actual Web services with precise measurements for nine QoS aspects, and a simulated offering variability. A Web-based prototype is also made available.

Keywords: Non-Functional Requirements, computing with words, fuzzy sets, fuzzy c-means

1 Introduction

The capability of systems for runtime adaptation to unforeseen environment conditions or changing requirements is becoming an expected feature for the next modern systems. Even more, it is expected that this capability be provided by the system itself (self-adaptive systems). Currently, there are several proposed solutions addressing partially this challenge. Recent work has focused

* This work was partially funded by FONDEF (grant D08i1155), UTFSM DGIP (grant DGIP 241167 and PIIC) and BASAL FB0821.

in addressing the self-adaptability from the requirements phase, where most of them represent requirements with a goal-based model ([YLLML08], [MPP08], [GSBHC08] [LL04]) which allows to represent functional and non-functional requirements (as goals and softgoals, respectively), domain and goal dependencies, among others.

This work proposes market-aware non functional requirements, where customers specify at design-time the required softgoals for the system under development using words (quality levels), and each time the market changes at runtime new operationalizations are recommended (probably because current operationalizations do not satisfy softgoals anymore). Our main contribution is to simplify the requirement maintenance model at runtime avoiding to customers (stakeholders) to map each time from quality expressed in words to quantified quality according to the new market view.

The rest of the article is organized as follows: Section 2 motivates the problem; Section 3 describes related proposals; Section 4 introduces basic background required to understand the proposed approach; Section 5 presents the approach; Section 6 describes a case study and its discussion; and Section 7 concludes the paper and draws future works.

2 Motivation

Given the ever increasing proliferation of available Web services in open Web catalogs in the Internet, we can safely assume that for each functionality type there is a set of Web services that could be potentially interchangeable [WSBCB10]; furthermore, there may be competition among their providers to make them distinguishable by their quality of service since they are functionally equivalent. Traditionally service providers provide precise measurement numbers for each quality aspect, but for stakeholders is easier to specify quality constraints with natural language words that denote the expected service level (e.g. "highly reliable component", "fast response time"). Thus, QoS requirements must be typically expressed as numbers to make them measurable and verifiable; these numbers are typically obtained (perhaps after iterating) with relevant stakeholders. Fuzzy Requirements [Li98] have been used in the past as a tool to capture client needs in terms of words and to map them into design specifications, also representing conflicts among them with a fuzzy multi-criteria decision making technique.

This approach works well at design time: stakeholders can use a snapshot of component QoS specifications to approximate the numbers that better represent their preferences. But this approach breaks at runtime, when components' QoS specifications are rapidly changing: (1) it is difficult for humans to constantly react to changes in market offerings, and (2) it is unfeasible to bring together stakeholders every time requirements needs to be restated. For instance, if stakeholders indicate that a component needs to send email with an "extremely acceptable" latency, an examination of prevailing market data may allow to decide (after some iterations) that "extremely acceptable" means "at most 200 millisec-

onds and ideally less than 100 milliseconds" (see left side of Figure 1); i.e. within this range stakeholder satisfaction increases as latency gets closer to 100 milliseconds (see right side of Figure 1). Softgoals depend of subjective perceptions of customers. Thus, this softgoal specification will probably become obsolete when the market changes.

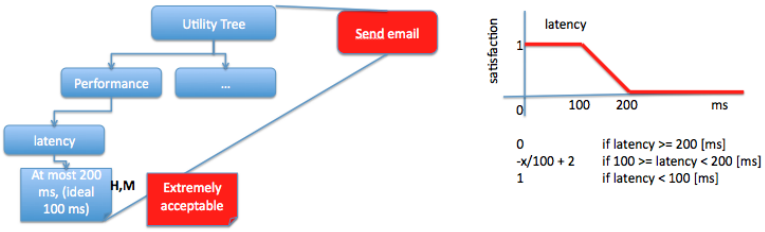

Fig. 1. utility tree

At runtime, when components providers are continuously competing several of them could change and probably improve their QoS specifications. Then, even when the numerical specification still is satisfied, the market has changed in such way that for those that before exhibited a "extremely acceptable" latency, they are not anymore, because other components' QoS improved in such a way that the stakeholders' perception change. Then, they could, by example, determine that now the "extremely acceptable" class ranges between 50 and 120 ms. Then, the current operationalization could become also obsolete (for instance if the current operationalization exhibits a 102 ms of latency).

3 Related Work

Several authors [YLLML08], [MPP08], [GSBHC08], [LL04] have used a goal-based approach as base to model requirements for self-adaptive systems because is very suitable specially because allows designers to model the tradeoff between NFRs, but all of them are prepared for foreseen adaptations where each violation as well as the alternatives to apply are enumerated at design time. Even more, not all of them addressed soft goals (NFRs) [WMYM09]. Our approach also use a reduced goal-based model, where not functional requirements are represented as fuzzy softgoals which are automatically adjusted each time the QoS component specifications change avoiding at runtime to rewrite them in order to align them with the stakeholders' expectations changes (because the context improvements).

Whittle et al. [WSBCB10] proposed RELAX, a language to address the uncertainty in the specification of requirements of self-adaptive systems, which allows developers to identify by specific special constructs (modal, temporal or ordinal operators) which requirements could be relaxed at runtime when the environment changes. Different from our work, they address the environmental

uncertainty as the doubt of to know at runtime if all the components which satisfy the requirements will be available, and let the system to adapt itself to an scenario where not all the needed parts are available but at least the critical ones are. Even more, RELAX could be seen as a mean to establish the boundaries of adaptive behavior, then it could be a complementary approach to be considered. The ultimate goal of authors is challenging, they intent to enable systems "requirements reflection" [BWSFL10] which means, systems are capable to reason about theirs own requirements and goals at run-time.

Cheng et al. [CSBW09] show how can be used to reason about uncertainty identifying its impact on the goal hierarchy. For instance KAOS is a goal-oriented software requirements capturing approach in requirements engineering. Originally, it did not consider adaptation, then Baresi et al. [BP10] extend it by include adaptive goals, which allows it to support the specification of "when" and "what" should be the adaptation. Besides, authors proposed a runtime infrastructure [BGP08] which constantly monitors the conditions to trigger adaptations if it is necessary. Baresi et al. [BPS10] formalized this model as FLAGS (Fuzzy Live Adaptive Goals for Self-adaptive systems) which represents requirements as runtime entities, distinguishing between crisp and fuzzy goals whose satisfaction is represented through fuzzy constraints. Unfortunately, they use the preferences numbers of the stakeholders to define the membership function boundaries, then, if the component QoS measurements change then these membership functions become obsolete because they were calculated using an older snapshot of the market where the numbers specified by the stakeholders made sense, but it could be that in current's market no longer has. Then, in order to align fuzzy goals with the changing expectations of stakeholders should be asked often what lead to a costly process.

Horkoff et al. [HY08] developed a backwards, qualitative, interactive analysis procedure for the i* Framework by express the goal model analysis as a SAT problem [SGM04] allowing represent them as a satisfaction problem which can be evaluated and actually is ran in an iterative fashion process, but allowing the interaction of the stakeholders at each stage in order to adjust the model. The our only critique to this work is that the softgoals adjustments rely on the stakeholders interaction, then if this interaction is slower than the QoS component specifications changes or for humans not possible to be assimilated, then the adjustments are not enough and then the model becomes obsolete against the component market. Then, we think our approach could improve this proposal by reducing the expensive and constantly participation of stakeholders based on the assumption that the market (QoS component specifications and components themselves) is rapidly changing. Same case with the work of Giorgini et al. [GMN02], we pointed that their approach can be improved by let customers use words instead of precise numbers, where these words are, in a backup process, mapped into precise numbers. Then, examples as "highly reliable systems" presented by these authors, when they are decomposed into observable goals do not need to be traduced into 99.9% of reliability, because "highly reliable" is a goal which is dependently of the current conditions of the market, and opera-

tionalizations should self-organize around the several quality levels of the aspect "reliability" and the ranges of each level should be calculated automatically from the available offer (over the time) instead of use stakeholders inputs (which implies a manual or a separated tool-support process in order to understand the offer in order to reach agreements which should be the precise numbers to use).

Silva et. al [SLRM11] proposed "Awareness Requirement" (AwReqs), a new type of requirement that can refer to other requirements or domain assumptions and their success or failure at runtime, and how they can be elicited and formalized, offering besides a requirements monitoring framework to support them. For instance, for the AwReq: "the ambulance response time of less than 8 min should be 95%". Compared to our work is similar to use quantifiable softgoals (" response time of less than 8 min" mapped to our language is "response time of at most 8 min") and using a threshold of 0.95 for the utility satisfaction function. But again, they are using precise numbers to specify these quality constraints, meaning these AwReqs specifications will become also obsolete. Same considerations for the work of Letier et al. [LL04], which proposed an approach that allow to specify partial degrees of goal satisfaction by quantifying the impact of alternative designs on goal-based systems using a probabilistic approach based on numbers at design time.

Serrano et al. [SSL11] implemented a propagation simulator which supports Requirements Engineering (RE) community for dealing with NFRs and a reasoning engine which is capable to deal with unpredictable situations that changes overtime. Regarding the requirements part, as in previous work with service quality perceptions (for instance [MGD03]), they basically modeled the goal-based runtime model as a fuzzy multi-criteria decision making problem, where again, the fuzzy levels boundaries of each fuzzy variable (which represents a softgoal) is not clear if they are arbitrarily assigned by the programmer or are elicited from stakeholders. Both cases, are not good enough to support the variability representation capability need it to be able to represent the constant changes in the QoS' components specifications at runtime.

Pimentel et al. [PCF11], on the other hand, proposed the FAST (failure handling for Autonomic Systems) framework which facilitates the definition of tolerance policies and their monitoring in order to evaluate them at runtime and trigger an adaptation if a failure was caught. There are only two available adaptation strategies: to replace the failed components (components which are violating any tolerance policy rule of the system) or ignore them. We present this work in this section because they also use words or levels to make decisions instead of crispy numbers, allowing requirements to adapt themselves to the new conditions of the market. But again, it seems instead of use from the context data the ranges values of the different levels, they use stakeholders defined values (which as we established in our work they becomes rapidly obsolete). Another difference with our work is that they use a binary classification instead of graduated one (for instance, our approach allows us to tell that a service with a "availability" of 90% is "extremely acceptable" with a membership of 0.7 but it

is "very acceptable" with a membership of 1, instead of to just say it is classified on this one or on the other).

4 Background techniques

In classical set theory, an object either is or is not member of a set, but membership in a fuzzy set is a graded, where e.g. an object x may belong to a fuzzy set A in a certain degree between 0 and 1 (values closer to 1 indicate higher degrees of membership). Fuzzy sets were proposed by Zadeh [Za68]. Formally, a fuzzy set A is defined by a set of ordered pairs, $A = \{(x, \mu_{A(x)}), x \in U, \mu_{A(x)} \in [0, 1]\}$, where $\mu_{A(x)}$ is a function (piecewise continuous or discrete) called *membership function*, that specifies the grade or degree to which any element x in U belonging to the fuzzy set A. A fuzzy value is a fuzzy set which membership function is continuous and defined over the real numbers. There are several forms to represent fuzzy values; the triangular and trapezoidal shapes are most popular (see [TAS11]). Depending of the type of the fuzzy number, we can need different parameters in order to define it: a triangular fuzzy value can be defined by a triplet (a, b, c), but a trapezoidal fuzzy value can be defined by four parameters (a, b, c, d). Typically these parameter values are decided upon by a group of experts or by consensus of the users.

On the other hand, linguistic variables can also be defined by fuzzy sets. A linguistic variable refers to the possible states are fuzzy sets assigned to relevant linguistic terms (e.g., "important", "acceptable", "not fast", etc.). Because at runtime, when QoS' components specifications are rapidly changing (1) for humans is difficult to frequently manage different snapshots of the market, and (2) also it would be expensive to bring together stakeholders each time requirements needs to be aligned with these changes, we proposed to replace stakeholders at runtime by an automatic method to identify the classes in the distribution of the data. This should be more robust than the classical manner because the Web service market is currently expanding and changing and users and even experts may not know much about the subject that they are clustering, or may not be familiar with the clustering system or the reality that they knew have completly changed.

The objective of fuzzy clustering methods is to divide a given dataset into a set of clusters based on similarity. In classical cluster analysis each datum must be assigned to exactly one cluster. Fuzzy cluster analysis relaxes this requirement by allowing membership degrees, thus offering the opportunity to deal with data that belong to more than one cluster at the same time. Each cluster is represented by a center and additional information about the shape of the cluster. Each data point can belongs to several clusters with different degrees of membership which are computed from the distances of the data point to the cluster centers.

The Fuzzy C-Means (FCM) clustering algorithm (see [PPKB05] for details) is applied to each quality attribute. After applying the FCM algorithm, the set of cluster prototypes, $\mathbf{V} = \{v_1, \ldots, v_c\}$ and the partition matrix $\mathbf{U}$ are available. The value $u_{ij} \in \mathbf{U}$ of the fuzzy partition matrix determine the membership

degree at which the attribute of th j-th Web service s_j fulfill the i-th linguistic term. The membership degrees of the dataset to the corresponding clusters are obtained by minimizing iteratively the objective function:

$$J_m(U, V; WS) = \sum_{j=1}^{m} \sum_{i=1}^{c} u_{ij}^m d^2(s_j, v_i) \tag{1}$$

The minimization process with respect to u_{ij} and v_i is done separately and necessary conditions for a minimum yields update equations for both:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d^2(s_j, v_i)}{d^2(s_j, v_k)} \right)^{\frac{1}{m-1}}} \quad \mathbf{v}_i = \frac{\sum_{j=1}^{n} u_{ij}^m s_j}{\sum_{j=1}^{n} u_{ij}^m}. \tag{2}$$

Due to imposed probabilistic constraints to minimize the objective function, the membership matrices are non-convex discrete fuzzy sets. In [TAS11] we used a variant of the fuzzy c-means algorithm [LCH03] that prevents the generation of non-convex fuzzy terms and ensures that the all the classes are correctly mapped to their domains.

5 Market-aware requirements

In this section we explain how market-aware requirements can be achieved by specifying non-functional requirements using words (quality levels) and applying fuzzy c-means to the sensed data from the market in order to allow services to self-organize around these levels in order to, at runtime, using the same specifications (at design time) obtain updated operationalizations recommendations.

5.1 Specifying market-aware requirements

To enable architects in finding an appropriate set of Web service candidates that satisfy their requirements, we propose a manual subprocess that must be performed in order to map requirements into market-aware-requirements.

First, architects specify functional requirements (FRs) that need to be satisfied, selecting from the functional categories list or using keywords. Then, for each FR, they specify a set of NFRs that the solution must address. Using this initial subset of NFRs, they are refined until obtaining a prioritized set of required QoS aspects. For each aspect that needs to be addressed, they specify the minimal acceptable class that the aspect must satisfy (e.g. response time should be at least "very acceptable"), as well as its relative importance vis-a-vis other aspects. This subprocess could be performed with the help of the stakeholders.

In Figure 1, an architect is trying to find a service capable to send email. He is concerned with the performance of the Web service; using the utility tree, refines the performance into latency QoS aspect and others (not considered in this example for the sake of simplicity). This is a step toward refining NFRs

to be concrete enough for prioritization (similar to softgoals). Then, typically, he specifies a concrete value range in order to make the aspect quantifiable and verifiable. But, as we mentioned before in Section 2, setting precise (crispy) values is an expensive task because architects and stakeholders need to be aware of the current market offerings. In the example in Figure 1, instead of defining crispy (market-dependent) numbers, they specify the minimal acceptable class, in this case denoted with the "extremely acceptable" word.

Regarding the prioritization of the aspects, we reuse the ideas of goal-based models using relative rankings: High(H), Medium(M) and Low (L).

5.2 Mapping market-aware specifications into a model@runtime

Each market-aware requirement is mapped into a runtime entity that must be monitored to determine whether the current operationalization still satisfies it; if it does not, this triggers a recommendation to change runtime representation according to the updated market view (i.e. a new operationalization) without manual intervention and without update quality constraints. Since requirements have been specified using words, for each requirement or requirement set, we can calculate its fuzzy multi-criteria utility function, which is defuzzified into two classes: "acceptable solution" and "not acceptable solution".

Inspired on previous work [Sc01] where a Multi-Attribute Utility Theory (MAUT) is used for estimating user's interests, we derive an utility function for each request, which is an aggregation function. In our case this is a weighted addition of the functional requirements' satisfaction, and for each functional requirement another weighted function of the evaluation of each relevant value dimension (in this case non-functional constraints) is added. Formally, for each functional requirement $FR = \{FR_1, ..., FR_N\}$, let $Q = \{q_1, ..., 1_J\}$ be a set of J sub-non-functional requirements. $MAC = \{MAC_1, ..., MAC_J\}$ is the minimal acceptable label class for each element of the set NFR, explicitly expressed by the customer or derived from the desired and reserved values ($D = \{D_1, ..., D_J\}$ and $R = \{R_1, ..., R_J\}$). Let $C = \{C_1, ..., C_M\}$ be the set of M functional-equivalent components that satisfy the specific FR_n; and $CC_m = \{c_{m1}, ..., c_{mJ}\}$, the current values that the component m provides for each non-functional attribute. Let $W = \{w_1, ..., w_J\}$ be the set of importance's weights assigned by the customer to each non-functional requirement. The weight is calculated assigning to each label ({H,M,L}) a predefined value ({1,3,5} respectively) normalized by the sum of all the values assigned to the different aspects chosen by the evaluator. These relative importance labels were specified by the evaluators in the previous stage. The utility function for each request is shown at equation 3.

$$U = \sum_{i=1}^{n'} v_i \cdot \sum_{j=1}^{m} w_j \cdot q_{jk} \tag{3}$$

where n' is the number of functional sub-requirements which comprised the request and q_{jk} is the degree of membership of the quality aspect j of the service k

to the required minimal quality class for this particular aspect j for the particular sub-requirement i. Quality classes are represented as fuzzy sets. Further details can be found in [TAS11].

6 Validation

6.1 Case Study

The case study dataset consists of a collection of 1500 Web services (operative at least until October 2011), based on the QWS Dataset [AM10] [1] which originally included 2507 actual Web service descriptors with nine QoS measurements. The quality aspects with their metric ids are *response time*(1), *availability*(2), *throuhgput*(3), *successability*(4), *reliability*(5), *compliance*(6), *best practices*(7), *latency*(8) and *documentation*(9).

To emulate the market changes, we have created two new market variations from the original dataset, where QoS' component specifications are improved in the first snapshot a random percentage between 0% and 30%, and in the second snapshot a random percentage between 30% and 50%. All of these modifications are applied to all services excepting a black list of service identifiers that are not modified because they were suggested in the original or in the second market snapshot; we did this to show that using the original word-expressed QoS specifications at design time (specified as market-aware requirements) is good enough to obtain updated suggestions that better reflect updated market conditions. Table 1 shows the ten complex requests that we use to assess our approach where basically, each sub functional requirement of the request is formed by the relative importance to others, the functional category which is assigned in our system, and then a list of the non-functional requirements which must be addressed, and for each one a triplet which represents the quality aspect id, the minimal acceptable class and the relative importance.

In Table 2 we show how the requirements specifications become obsolete as the market evolves (100% of the requests) meaning if the quality levels are not updated as we proposed those requirements becomes obsolete as well their suggested operationalizations. If we use market-aware requirements even when the market evolves, using the same requirements specifications, clients do not need to specify again them and obtain new operationalizations if the satisfaction degradation is too high. For the 100% of the requests, using the same requirements specified at design time, we obtained at runtime new operationalizations which outperforms the obsolete ones. Then, using our approach the number of times that we rewrite the requirements was 0.

6.2 Prototype implementation and dataset

A prototype system was developed to asses our approach [2], which allows architects of service-based systems (1) specifying a set of functional goals; (2) ranking

[1] http://www.uoguelph.ca/q̃mahmoud/qws
[2] http://cc.toeska.cl/cww

Table 1. Test Cases Design

id	FR(weight)	{(category_id){metric_id, mac, weight}+}+
R1	latitude longitude map (H)	{(12954){1,EA,H}{2,EA,H}{3,EA,H}}
	country zip(H)	{(13338){3,EA,H}{5,EA,H}{7,EA,H}{8,EA,H}}
R2	file size image(H)	{(12915) {1,EA,H}{2,EA,H}{3,EA,H}}
R3	sequence protein(H)	{(13100){1,EA,H}{3,EA,H}{8,EA,H}}
R4	sale, product)(H)	{(12958){7,EA,H}{6,EA,H}{8,EA,H}}
R5	job progress (H)	{(13131){1,EA,H}{1,EA,H}{7,EA,H}}
	email(H)	{(13206){1,EA,H}}
R6	phone info(H)	{(12943){1,EA,H}{3,EA,H}{7,EA,H}}
	fax text (H)	{(13129){1,EA,H}{2,EA,H}}
R7	lookup email(H)	{(12979) {1,EA,H}{5,EA,H}}
	weight package delivery (H)	{(12940){1,EA,H}{2,EA,H}{3,EA,H}}
R8	currency country(H)	{(12994){1,EA,H}{2,EA,H}{7,EA,H}}
	zip country(H)	{(13005){1,EA,H}{5,EA,H}}
R9	subscription(H)	{(12973){1,EA,H}}
	delivery(H)	{(12946){1,EA,H}}
	zip map(H)	{(13005){1,EA,H}}
R10	notification subscription(H)	{(13387){1,EA,H}}
	email(H)	{(13206) {1,EA,H}}

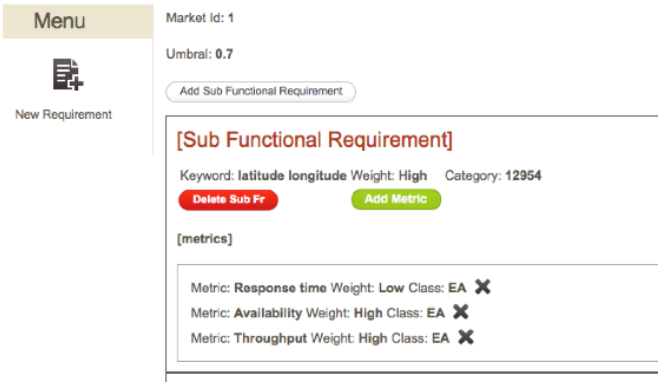

Fig. 2. The prototype Tool interface

Table 2. Test cases Execution - For each test case, first column shows id of request, second the operationalization suggested at design time with the specification given by stakeholders by observing the snapshot at design time, third and fourth columns shows how the operationalization becomes obsolete as the market evolves.

id	operationalization	design-time	(1st)runtime	(2nd)runtime
R1	84193574,4869688	(0.875)	(0.545)	(0.27)
R2	26914897	(1)	(0.404)	(0)
R3	15103218	(1)	(0.667)	(0.501)
R4	66537772	(1)	(0.33)	(0.33)
R5	43546302,15103218	(1)	(0.833)	(0.833)
R6	78667380,23013627	(1)	(0.833)	(0.333)
R7	191615174,115114747	(1)	(0.580)	(0.449)
R8	112791393,179771826	(1)	(0.438)	(0.137)
R9	191615174,162199213,145056169	(1)	(0.976)	(0.76)
R10	179326172,167233563	(1)	(0.5)	(0.5)

functional goals; (3) specifying for each functional goal its non-functional requirements (softgoals) using words; (4) ranking the softgoals of each functional goal; and finally (5) finding a set of operationalization services that satisfies the request. Also, suggestions change as the service market evolves, recommending new operationalization sets (if appropriate) based on updated market conditions.The prototype system contains four main components to enable component composition. *Taxonomy builder* which uses the components' descriptor files to create a complete hierarchical taxonomy using Formal Concept Analysis (FCA) and Galois Lattices (see [TAS11] for more details). *Non-Functional Clusterizator builder*, which using fuzzy c-means clustering, for each market snapshot, functional category and quality aspect, five fuzzy clusters of service are iteratively built (this component is built in Python over Peach [3]). The *API* which is a public Application Programmer Interface which exposes all the functionality implemented as Web services. The *Presentation Layer* which was built using Ruby on Rails, it consumes the services exposed by the API.

6.3 Calculating the current market view

The infrastructure to provide time after time the current market view is composed of several components. The *functional crawler* component which collects from different Web-based catalogs the Web service descriptors. The *QoS certifier* component, based on the endpoint list sent by the previous component, runs a benchmark tool over the Web services in order to gather the QoS measurements. The*functional clustering* component, based on the Web service descriptors, it clusters the Web services according to their functionality (this can be replaced by the categories provided by the catalog). And the *QoS-fuzzy clustering* which for each functional cluster and for each quality aspect it cluster into c classes

[3] http://code.google.com/p/peach/

(quality levels). The ranges of each class depend of the services which belong to the classes.

To this end, we define five cluster of QoS or classes: "poorly acceptable" (PA), "almost acceptable" (AA), "acceptable" (A), "very acceptable" (VA), and "extremely acceptable" (EA). These classes are fuzzy numbers even when the measurements are crispy. Each fuzzy number has a membership function that is limited and continuous and in this case we have choose a triangular shape, represented by a triangular fuzzy number A with membership function $\mu_A(x)$ which is defined on R.

$$\mu_A(x) = \begin{cases} \frac{x-a_1}{a_M-a1} & \text{if } a_1 \leq x \leq a_M \\ \frac{x-a_2}{a_M-a_2} & \text{if } a_M \leq x \leq a_2 \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $[a_1, a_2]$ is the supporting interval and the point $(a_M, 1)$ is the peak. We have two types of quality aspects, those that we want to minimize and those that we want to maximize. Therefore, for different aspects we will have individuals that have a great belonging grade to the "EA" with a lower value for the metric measurement against those that will be completely opposite, where the individual with greater values are classified into that class. For instance, this happens when we are comparing response time against the reliability quality aspect.

As we mentioned before, for each Web service functional cluster and for each QoS aspect we apply the modified fuzzy c-means clustering to organize the Web service around of the five QoS classes mentioned before, obtaining the center of each class as well their supporting interval based on the crispy measurements obtained by the QoS crawler. The advantage to use Fuzzy logic is that each QoS measurement is classified in at least one class with a belonging degree. For instance, a specific weather Web service could be classified into the "EA" class with a belonging degree of 1 as well it could be classified into the "VA" class with a belonging degree of 0.8.

7 Conclusions and further work

This article has described an approach to support architects to specify NFRs with words at design time which are automatically expressed in numerical terms whenever providers change components QoS characteristics, thus enabling market-aware requirements. Current fuzzy-based techniques are expert and/or consensus-based, and therefore too fragile, expensive, non-scalable, and non- self-adaptive. We address market dynamism by using each time a modified fuzzy c-means module, which allows service providers to automatically be organized around QoS levels. This approach is not only useful for adaptive systems, but also for any kind of service-based system where there is strong competition (several continuously-improving providers for the same requirements). Our approach allows at run-time not-obsolete improvements using the specifications made at design time. An extra advantage of our approach is that architects can specify their QoS constraints with words, without really knowing what are the current best quality

ranges. Most approaches revisited at related work shared a closed-world assumption, where stakeholders can decide the better numbers for each variable because they know the current QoS components specifications. However, in practice architects cannot retain all market information and also be aware of its continuous changes, and even if they could (say, with proper tools), it would still be unfeasible to gather stakeholders to determine the proper number every time a softgoals' hidden number ranges must be adjusted. On the other hand, none of the related work address the same problem as we are, specifically the problem faced by architects who are designing, composing, deploying and maintaining service-based systems against a constantly changing (mainly improving) QoS component specification offerings, which make obsolete at runtime requirements specified at design time. Then current works demand that architects, besides maintaining system, also maintain the system goals to keep them aligned with the implicit stakeholder expectations (and also changing due to market changes).

As future work we are considering to apply market-aware requirements to self-adaptive systems as a model robust enough to drive the adaptation, allowing by properly tuning the parameters, obtain new recommendations when the current ones become obsolete.

References

[AM10] Al-Masri E., Mahmoud Q. WSB: a broker-centric framework for quality-driven Web service discovery, in Software: Practice and Experience, vol. 40. John Wiley and Sons, Ltd., (2010) 917–941.

[BGF09] Bencomo N., Blair G., France R. Guest editor's introduction: Models@run.time. IEEE Computer, (2009).

[BGP08] Baresi L., Guinea S., Pasquale L. Integrated and Composable Super- vision of BPEL Processes. In: Proc. of the 6th Int. Conf. of Serv. Oriented Computing (2008). 614–619.

[BP10] Baresi L., Pasquale L. Adaptive Goals for Self-Adaptive Service Compositions. In Proceedings of the 2010 IEEE International Conference on Web Services (ICWS '10). IEEE Computer Society, Washington, DC, USA. (2010) 353–360.

[BPS10] Baresi L., Pasquale L., Spoletini P. Fuzzy Goals for Requirements-Driven Adaptation. In Proceedings of the 2010 18th IEEE International Requirements Engineering Conference (RE '10). IEEE Computer Society, Washington, DC, USA. (2010) 125–134.

[BWSFL10] Bencomo N., Whittle J., Sawyer P., Finkelstein A., Letier E. Requirements reflection: requirements as runtime entities. ICSE (2010). 199–202.

[CSBW09] Cheng B., Sawyer P., Bencomo N., Whittle J. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: MoDELS'09: Proceedings of the 11th international conference on model driven engineering languages and systems. Springer, Berlin. (2009)

[GMN02] Giorgini P., Mylopoulos J., Nicchiarelli E., Sebastiani R. Formal Reasoning Techniques for Goal Models. Journal Data Semantics 1. (2003). 1–20

[GSBHC08] Goldsby H, Sawyer P, Bencomo N, Hughes D, Cheng B. Goal-based modeling of dynamically adaptive system requirements. In: 15th annual IEEE international conference on the engineering of computer based systems (ECBS) (2008)

[HY08] Horkoff J., Yu E. Qualitative, Interactive, Backwards Analysis of i * Models. iStar 2008 (2008) 43–46

[KM07] Kramer J., Magee J. Self-Managed Systems: an Architectural Challenge. In 2007 Future of Software Engineering (FOSE '07). IEEE Computer Society, Washington, DC, USA. (2007). 259–268

[LCH03] Liao T., Celmins A., Hammell R. 2003. A fuzzy c-means variant for the generation of fuzzy term sets. Journal Fuzzy Sets and Systems. 135, 2 (2003), 241–257.

[Li98] Liu X. Fuzzy Requirements. IEEE Potentials. (1998). 24–26

[LL04] Letier E., van Lamsweerde, A. Reasoning about partial goal satisfaction for requirements and design engineering. In SIGSOFT'04/FSE-12: 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering, (2004) 53–62.

[MGD03] Chen M., Tzeng G., Ding C. Fuzzy MCDM approach to select service provider. The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03. 1(2003) 572–577

[MPP08] Morandini M, Penserini L, Perini A Modelling self-adaptivity: a goal-oriented approach. In: Proceedings of second IEEE international conference on self-adaptive and self-organizing systems (SASO), (2008) 469–470

[PCF11] Pimentel J., Castro J., Franch X. Specification of Failure-Handling Requirements as Policy Rules on Self-Adaptive Systems. WER 2011

[PPKB05] Pal N., Pal K., Keller J., Bezdek J. A Possibilistic Fuzzy c-Means Clustering Algorithm. IEEE Transactions on Fuzzy Systems, vol. 13, number 4. (2005) 517–530

[Sc01] Schaefer, R. Rules for Using Multi-Attribute Utility Theory for Estimating a User's Interests. In Proc. ABIS Worksh. Adaptivity und Benutzermodellierung in interaktiven Softwaresystemen, Dortmund, Germany, October (2001).

[SGM04] Sebastiani R., Giorgini P., Mylopoulos J.: Simple and Minimum Cost Satisfiability for Goal Models. CAiSE'04. Advanced Information Systems Engineering. LNCS. (2004). 675–693

[SLRM11] Silva V., Lapouchnian A., Robinson W., Mylopoulos, J. Awareness requirements for adaptive systems. In Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '11). ACM, New York, NY, USA, (2011). 60–69.

[SSL11] Serrano M., Serrano M., do Prado S., Leite J. Dealing with softgoals at runtime: A fuzzy logic approach, Requirements@Run.Time (RE@RunTime), 2nd International Workshop. (2011)

[TAS11] Torres R., Astudillo H., Salas R. 2011. Self-Adaptive Fuzzy QoS-Driven Web Service Discovery. In Proceedings of the 2011 IEEE International Conference on Services Computing (SCC '11). IEEE Computer Society, Washington, DC, USA. (2011) 64–71.

[WMYM09] Wang Y., Mcilraith S., Yu Y., Mylopoulos J. Monitoring and Diagnosing Software Requirements. Automated Software Engineering, 16(1). (2009) 3–35.

[WSBCB10] Whittle J., Sawyer P., Bencomo N., Cheng B., Bruel, J. RELAX: a language to address uncertainty in self-adaptive systems requirement. Requirements Engineering, Springer London, ISSN 0947-3602. (2010)

[YLLML08] Yijun Y, Lapouchnian A, Liaskos S, Mylopoulos J, Leite J. From goals to high-variability software design, vol 4994. Springer, Berlin (2008)

[Za68] Zadeh L. Fuzzy Algorithms. Information and Control, vol.12, number 2. (1968) 94–102