Managing requirements@run.time with a linguistic decision making approach*

Romina Torres[1,2] Hernan Astudillo[2]

[1] Universidad Andres Bello, Chile,
romina.torres@unab.cl, http://facultades.unab.cl/ingenieria/
[2] Universidad Técnica Federico Santa María, Chile,
hernan@inf.utfsm.cl, http://www.inf.utfsm.cl

Abstract. In the seminal work about requirements, Zave and Jackson established that if specification models hold the correctness criteria then, they can be used instead of requirements to make decisions (e.g. select an architectural configuration to implement requirements). Unfortunately, during runtime for systems under changing environments, domain assumptions may change and if they are not properly maintained (synchronized) the correctness criteria may becomes not useful to detect when the specification model is not anymore a valid representation of requirements. Thus, requirements may be violated but not properly detected. In order to avoid specification models become obsolete during runtime, we already proposed reify requirements into abstract specification models. In this paper we extend the correctness criteria to requirements@run.time and we propose specifically the linguistic decision making (LDM) models to represent these abstract models. We present an illustrative example of how our approach works. The main contribution of this approach is obtained during runtime, when the false negative rate error on determining when requirements are violated is reduced.

Keywords: correctness criteria, requirements@run.time, linguistic decision making models

1 Introduction

Requirements (R) are the goals that drive the development, the adaptation and the evolution of systems. They are typically comprised by a set of functional requirements (FRs) constrained each one (or the system as a whole) by a set of non-functional ones (NFR). FRs are also known as capabilities. They specify the functions that a system must be able to perform (for instance, FR_1: *"display the heart rate of a patient connected to the monitor"*). NFRs instead are statements of how a system must behave; they are non-functional constraints (NFCs) upon the systems' behavior (for instance NFC_1 over the FR_1: *"with*

* This work is partially supported by projects VirtualMarket (Fondef CA12i10380), UTFSM-DGIP 24.12.50, and CCTVal (Basal FB0821)

a proper response time to a change in the patient's status"). NFRs are by nature ambiguous (e.g. *"... a proper response time to..."*). Indeed, humans often to use linguistic terms to assess qualitative aspects [DVV92]. They are typically expressed using natural language [MK95] (e.g. system's reliability may be classified: "Ideal", "Defective", "Faulty", "Erroneous", "Malfunctioning", "Degraded" and "Failed" [Par97]). Thus, in order to deal with the ambiguity of R, software engineers reify them into precise and verifiable specification models (S) by using a set of domain assumptions (D) built by the observation of the domain. Using the D, software engineers may specify the NFC_1 as *"... must respond to a change in the patient's status within 2 seconds"*. Where D may be comprised of information like *"the approximated average time of this kind of dispositive"*, or *"the needed response time to allow medical doctors attend effectively their patients in case of emergency"*, to name a few. Citing authors Zave and Jackson: {*"The primary role of domain knowledge is to bridge the gap between requirements and specifications"*} [ZJ97]. In their seminal work about requirements, Zave and Jackson established that a software engineer should be able to prove that requirements R hold the following **correctness criteria** [ZJ97]:

$$S, D \vdash R \tag{1}$$

if, and only if, S and D are satisfied and consistent. Thus, instead of using the probably ambiguous R, software engineers may use safety S, which is a precise representation of R (valid meanwhile D are valid and consistent with S). Unfortunately, the correctness criteria of Zave and Jackson [ZJ97] presented on equation 1 becomes invalid when design-time specification models are used during runtime for those systems which are under non-stationary environments. Mainly because through time, perceptual system may be degrade [Hal06]. To humans, the concepts' meanings (like "proper response time" or "ideal reliability") change, mainly because they are built over their perceptions [Zad99]. *"Humans have a remarkable capability to perform a wide variety of physical and mental tasks without any measurements and any computations"*. They ussually use rough perceptions [Zad01].

In order to mitigate the obsolescence of S, we have already proposed to divide the requirements reification process into two stages: reify first R into S^* and then, reify S^* into precise ones each time they are needed to verify the requirements' satisfaction [Tor13,TBA12]. In this work, we formalize S^* as linguistic decision making (LDM) models, we present a framework to allow decision makers to assess models and we extend the correctness criteria to runtime in order to keep using it as a continuous verification mechanism that S are still a valid representation of R. The rest of the article is organized as follows. Section 2 describes the related work. Section 3 introduces the LDM approach. Section 4 presents the reification process and the assessment approach. Section 5 shows an illustrative example. And finally, Section 6 concludes the paper and draws future work.

2 Related Work

The main strategy of managing systems under changing environments has been to bring the design-time models to runtime in order to continuously verify their satisfaction by their current architectural configurations (Cs) [BBF09]. Due to requirements and environment may change, authors like Baresi et al. [BG10] and Epifani et al. [EGM+09] proposed that models should evolve as they do. According to the correctness criteria [ZJ97] this should means that each time D or R change, S should be updated in order to mitigate the probability of using an obsolete S. Under changing environments, changing D are the rule not the exception. If the environment changes (1) D should be updated in order to maintain themselves representative of the environment and, therefore (2) S should be synchronized with D in order to maintain S consistent with the changing D. Thus, if D are not synchronized with the changing environment, then adaptive systems may be not detecting "silent" violations (or violations without symptoms) mainly because S becomes obsolete and inconsistent with the perceptions of humans observing the environment represented by the D. We call "silent" violations, those in which architectural configurations do not drop their quality levels but other functionally-equivalent architectural configurations improve significantly making perceptions about the meaning of their domain assumptions shift.

On the one hand, all those proposals that move precise specification models to runtime [AP07,BCG+10,BG11] may be potentially missing requirements violations because they use directly their design-time models (using precise numbers to specify constraints) to make decisions during runtime (when probably they had already became obsolete due to the inherent competence nature of an environment full of functionally-equivalent alternatives). According to Predrycz et al. [PEP11] *"In practice, exact values of parameters of models are not so common. Normally uncertainty and imprecision arise due to lack of knowledge and incomplete information reflected in system structure, parameters, inputs, and possible bounds"*.

On the other hand, those proposals [CGK+11,FGT12,EGM+09] not using precise numbers to specify non-functional constraints proposed to augment each system with a component responsible for the continuous updating of the models' parameters. They implemented it as a Bayesian estimator, who re-estimated periodically the values of the parameters using the data collected from the running system. However, the main and common drawback of these works is that they represented each model's constraint as a probability distribution. But, according to Zave and Jackson [ZJ97], requirements are humans' statements (which are more familiar to the human reasoning) on the desired phenomena in the world; they are obtained using a consensus process which is based on perceptions. Non-functional constraints like "fast", "not so fast", "not so slow" and "slow" are quite difficult to model as probability distribution when their meanings are context-dependent; thus, their meanings depend on the context of the persons observing the environment and specifying constraints. Thus, constraints' meanings depend on several not deterministic and unmanageable factors. Thus,

rarely, models' constraints, which are on this case non-functional requirements, could have a probabilistic nature as Martínez et al. identified in most real-world problems [MDH+09], where information is not perfect. Even when the *MOSES* framework proposed by Cardellini et al. [CCG+12], did not propose to use probabilistic distribution, they did proposed to use the average statistic estimators, which were recomputed periodically using a data-driven algorithm (making it computationally expensive). However, from the point of view of how specification models were represented, the *MOSES* framework had a poor representation of the constraints (only two classes: under and over the average) which is not enough to express the richness of non-functional constraints.

In [TBA12] we divided the reification process into two parts. The first one, in which software engineers represent their requirements as abstract specification models (S^*) and the second one, in which the abstract models are reified into S (a concrete one).

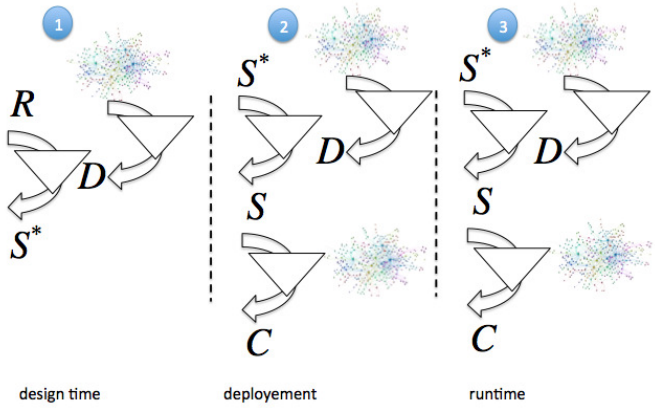

Fig. 1. The reification process from requirements into precise specification models is divided into two consequent phases. The first one occurs when model is built, the second one each time requirements satisfaction needs to be assessed. Proposal originally presented in [TBA12].

Figure 1 summarizes our proposal showing that at design time S^* were built, at deployment time S^* is reified into S using the current D and a C is selected and finally, periodically during runtime, S^* is reified each time is needed to re-evaluate the satisfaction of R by C in order to generate a replacement when properly. Even when in [Tor13] we proposed that architects should specify S^* as linguistic decision making (LDM) models, we did not built them correctly as LDM models because we did not use a linguistic aggregator operator to aggregate the different constraints and we did not use a linguistic decision making approach. We just modeled non-functional properties as linguistic variables [Zad75], where their range of possible values were linguistic values instead

of numbers. We used a computing with words (CWW) machinery [Zad96] but assessing separately the satisfaction of alternative to each constraint to finally aggregate them using the weighted average (WA) operator. Different from our proposal in [TA12] we use in this paper an expert driven instead of a data-driven approach to determine the linguistic values' meanings, extending in this manner, our approach to those areas where data is not available but just humans perceptions.

3 Linguistic decision making approach

Clearly, requirements and therefore specification models are multicriteria decision making models where techniques of multiattribute analysis of alternatives in a fuzzy environment developed on the basis of fuzzy preference modeling are needed. We narrow the scope in this paper to individual decision-making (even when it can be applied to procedures of group decision-making) [PEP11]. It is already accepted that these techniques can lead to different solutions (not deterministic) due to the associated uncertainty.

To solve linguistic decision making problems, the resolution scheme of multicriteria decision making problems has been extended [HH00]. Classically, this resolution scheme has only two phases: (i) an aggregation phase, and (ii) an exploitation phase. In the aggregation phase, experts aggregate their opinions and in the exploitation phase, they rank the alternatives and typically choose one. Linguistic decision resolution schemes [HH00] add two previous and additional stages: (1) the choice of the linguistic term set and its semantics, and (2) the choice of the aggregation operator of linguistic information. In the first stage, experts model each criteria as a linguistic variable whose linguistic values may be defined using for instance: the semantic model [DB88] or the symbolic model [DVV92]. In the second stage, experts define the appropriate aggregation operator of linguistic information for aggregating and combining their opinions about a constraint or aggregate several constraints opinions to get a global assessment [HH00].

An initial proposal of resolution scheme of linguistic decision making problems using CWW [Bon80] had three stages: (1) an encoder that translated words into fuzzy sets, (2) a CWW engine that infer the output as a fuzzy set, and (3) a decoder that retranslated from fuzzy sets into linguistic results. Zadeh [Zad96] formalized the three main stages presented in this initial proposal [Bon80] as: (i) the translation, (ii) the granular computing, and (iii) the retranslation stage. The translation subprocess is the explicitation of the propositions into the Generalized Constraint Language (GCL). The granular computing is the inference subprocess, where constraints are propagated to obtain a constraint on a variable of interest. It uses rules of inference in fuzzy logic [Zad96] to propagate the rules governing the fuzzy constraint propagation process. In the last process, the output of the inference subprocess is retranslated in order to obtain a terminal dataset from which a proposition in natural language may be derived if needed.

It is very important to notice that in the original conceptual structure of CWW, the explanatory database (ED) is considered static, then specifically the granules representing the linguistic values do not change over time, or probably the change is so slow that can be manually updated by humans.

Yager [Yag95] also presented a resolution scheme to make decisions under ignorance using linguistic values. The author considered that in many real problems, the information about the satisfaction associated to different alternatives may be at best expressed in terms of a linguistic scale. The author also considers the two basic stages of the resolution scheme (aggregation and exploitation phase). Each alternative is evaluated in each criteria (if there is available data) and then, depending upon the aggregation function selected by the human making a decision (decision model), alternatives are evaluated and ranked. The author does not address cases in which the meanings of linguistic values could change through time.

Mendel [Men01] proposed the perceptual computer, an architecture for CWW, where perceptions ("words") activate the perceptual computer (Per-C). The Per-C has an encoder and a decoder (similar to the translation and retranslation subprocesses [Zad96]), which needs to transform perceptions into fuzzy sets and numbers into perceptions. Later [Men07], the same author proposed that the internal fuzzy logic system in the Per-C should return fuzzy sets instead of numbers, and only then, these fuzzy sets should be decoded back to perceptions that humans can understand. The Per-C is based on the codebook [Men01], in which every word (the vocabulary) is modeled as an interval type 2 fuzzy set. But independent of the representation of each "word", it is important to notice that Mendel does not discuss either how this architecture addresses or mitigate the change of perceptions that could affect the "meanings" of the words through time (the *codebook*).

4 Proposal

In this Section we present our proposal to support systems on managing their requirements@run.time in order to safety decide when the current architectural configuration should be replaced because requirements are becoming violated. Our proposal consists on to divide the reification process in order to software engineers represent at design time requirements as LDM models (abstract specifications) and to assess during runtime the satisfaction of current architectural configurations to requirements (by using a LDM approach) in order to determine when a replacement is needed. The advantage of using a LDM approach is that obsolescence of S are independently mitigated from the changing D mainly because the LDM approach includes a stage to update the linguistic values' meanings before to assess the models (demanding experts regularly update D to update these meanings). To complete this vision, we extend the current correctness criteria to runtime in order it can be used at any time t:

$$S^*(D(t)), D(t) \vdash R \tag{2}$$

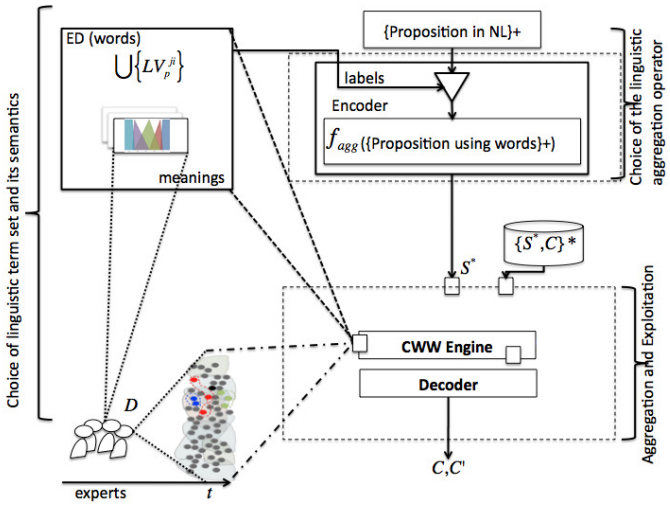

Fig. 2. Proposal to support adaptive systems on managing their requirements@run.time in order to safety decide when the current architectural configuration should be replaced because requirements are becoming violated

where $S^*(\cdot)$ is an abstract specification model, whose reification depends on the available domain assumptions at time t. Thus, the original correctness criteria shown on equation 1 would be a particular case of the extended correctness criteria that we shown on equation 2. In the following subsections we show the stages of our proposal which are shown in Figure 2.

4.1 The choice of the linguistic term set with its semantics (setup stage)

The initial stage of any linguistic resolution scheme is the choice of the linguistic term set and its semantics. In the left side of the Figure 2, we show the Explanatory Database, where the linguistic term set comprised of a set of words is regularly (as the alternatives changes) maintained by humans experts.

Let LV^{ji} be a linguistic variable representing the j-th non-functional property $G^{(j)}$ of the functionally-equivalent set of alternatives capable to provide the functionality i. Let $\{LV_1^{ji}, \ldots, LV_P^{ji}\}$ be the P possible linguistic values of the linguistic variable LV^{ji} according to a selected metric.

Let $ED = \{LV_1^{ji}, ..., LV_p^{ji}, ..., LV_P^{ji}\}$ the linguistic term set or "words" available in the ED (as Figure 2 shows). Each linguistic value LV_p^{ji} is a fuzzy set [Zad65] represented by a fuzzy number.

For instance, assume that for the linguistic variable *response_time* of a set of functionally-equivalent services i, the linguistic term set could be defined as $\{LV_1^{response\_time\ i} : excellent, LV_2^{response\_time\ i} : very\ good, LV_3^{response\_time\ i} : good, LV_4^{response\_time\ i} : fair, LV_5^{response\_time\ i} : poor\}$, where a triangular fuzzy

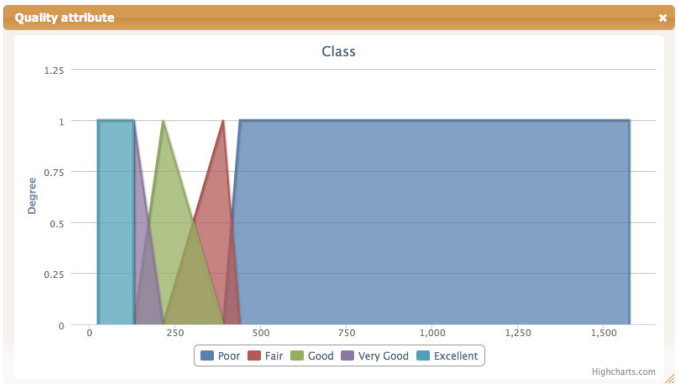

Fig. 3. An illustrative example showing the graphical representation of each "word" of the variable *response_time* using triangular fuzzy numbers.

number representation has been chosen as Figure 3 shows. We delimit this work to an *expert-driven* approach [PEP11], where human experts perform a perception-based process to set up the "words' meanings" (membership functions' parameters) by observing the alternatives first (specifically their properties' values) as the bottom left part of Figure 2 shows.

4.2 The choice of the aggregator operator of linguistic information (setup stage)

To aggregate NFCs we allow the use of the AND and OR operators. Whenever the AND operator is used to aggregate the NFCs, then the selected alternative A_k must simultaneously satisfies all the constraints. On the other hand, if the OR operator is preferred, then the selected alternative must satisfy at least one constraint. We also allow the use of unary ordering-based modifiers "*at least*" ($\mathcal{L}$) and "*at most*" ($\mathcal{M}$) to manage the minimum acceptable level of satisfaction for the user about the non-functional property $G^{(j)}$ (the non-functional constraint). Bodenhofer [Bod08] defined these modifiers as follows. Let l_p be the label of the linguistic value LV_p^{ji} (p fixed), the fuzzy sets *"at least* l_p*"* and *"at most* l_p*"*, abbreviated as $\mathcal{L}(l_p)$ and $\mathcal{M}(l_p)$, respectively, are defined as $\mathcal{L}(l_p)(x) = \sup\{LV_p^{ji}(y)$ such that $y \in \mathcal{X}$ and $y \preceq x$ and $\mathcal{M}(l_p)(x) = \sup\{LV_p^{ji}(y)$ such that $y \in \mathcal{X}$ and $x \preceq y\}$ where $\preceq$ is a crisp ordering on $\mathcal{X}$.

Regarding the operators to aggregate opinions from different decision makers regarding one constraint, we allow the use of linguistic aggregation operator based on linear ordering: the linguistic max operator (LM_1) and the linguistic min operator (LM_2) defined as $LM_1(a_1, a_2, \ldots, a_n) = max_j\{\ a_j\}$ and $LM_2(a_1, a_2, \ldots, a_n) = min_j\{\ a_j\}$ respectively. To deal with an intermediate situation, Yager [Yager88] proposed the ordered weighted averaging (OWA) op-

erator defined as follows

$$\phi(A) = \phi(A_1, ..., A_n) = \sum_{i=1}^{n} w_i \tilde{A}_i \tag{3}$$

which is a mapping $\phi : \mathbb{R}^n \to \mathbb{R}$, where the vector of weights is $W = [w_1, ..., w_n]$ with $0 \leq w_i \leq 1$ and $\sum_{i=1}^{n} w_i = 1$. Moreover the set $\{\tilde{A}_1, ..., \tilde{A}_n\}$ is a prioritized permutation of the collection $\{A_1, ..., A_n\}$, indicating that the element A_i is the i-th greatest element in the set $\{A_1, ..., A_n\}$, i.e., $\tilde{A}_i \geq \tilde{A}_j$ for $i \geq j$. It is important to notice that by properly adjusting the weights we can manage the degree of LM_1 and LM_2 operators.

4.3 The definition of the LDM models (design time)

As right upper side of Figure 2 shows, decision makers (clients) using the labels of the linguistic term set to encode their requirements by encoding their non-functional requirements as non-functional constraints. These constraints were initially defined in natural language

$$\{Proposition\ in\ NL\}+$$

They consist of an aggregation of the constraints written in terms of words extracted from the ED.

$$f_{agg}(x_1, .., x_n) \quad \text{where } x_j \in \mathcal{X}_{G^{(j)}} \text{ and } f_{agg} := \{Proposition\ using\ words\}+ \tag{4}$$

The expression {Proposition using words}+ can be constructed with a context-free grammar to generate the linguistic expression. The elements of the context-free grammar $G_H = (V_n, V_T, I, P)$ are the following:

$$\begin{aligned} V_N &= \{\langle\text{primary term}\rangle, \langle\text{composite term}\rangle, \\ &\quad \langle\text{unary relation}\rangle, \langle\text{conjunction}\rangle, \langle\text{disjunction}\rangle\} \\ V_T &= \{LV_1^{ji}, ..., LV_p^{ji}, ..., LV_P^{ji}, \mathcal{L}, \mathcal{M}, AND, OR, OWA, NOT\} \\ I &\in V_N \end{aligned}$$

The production rules are defined with the Backus-Naur form over the ED (see [RM12]). The brackets symbolize optionality while the symbol "|" stands for alternative elements. The production rules P are the following:

$$\begin{aligned} I &::= \langle\text{primary term}\rangle|\langle\text{composite term}\rangle \\ \langle\text{composite term}\rangle &::= \langle\text{unary relation}\rangle\langle\text{primary term}\rangle| \\ &\quad \langle\text{primary term}\rangle\langle\text{binary relation}\rangle\langle\text{primary term}\rangle| \\ &\quad \langle\text{primary term}\rangle\langle\text{binary relation}\rangle\langle\text{composite term}\rangle \\ \langle\text{primary term}\rangle &::= \langle\text{word}\rangle|\langle\text{unary relation}\rangle\langle\text{primary term}\rangle \\ \langle\text{word}\rangle &::= LV_1^{ji}|...|LV_p^{ji}|...|LV_P^{ji} \\ \langle\text{unary relation}\rangle &::= \mathcal{L}|\mathcal{M}|NOT \\ \langle\text{binary relation}\rangle &::= AND|OR|OWA \end{aligned}$$

4.4 Assessing specification models expressed as LDMs (at deployment or during runtime

As we can see in the right bottom part of Figure 2, the alternative selection decision is part of the aggregation and exploitation phases. In the classical resolution scheme, during the aggregation phase, all the assessment values of the different alternatives are collected from the evaluators. Then, during the exploitation phase, the alternative $\tilde{A}^1$ is selected. Algorithm 1 shows how our framework selects a proper alternative to satisfy the LDM.

Algorithm 1 The *tLDM* framework recommending an alternative during deployment time

Require: Set of alternatives to be assessed $A^1, ..., A^m$. The performance measurements $(x_1^i, ..., x_n^i)$ for each alternative A^i with respect to all the attributes $G^{(1)}, .., G^{(n)}$.
Ensure: The alternative $\tilde{A}^1$ with the highest satisfaction degree (SCORE) is selected.
1: From the ED the membership functions of the relevant $LV_p^{(j)}$, $j = 1..n, p = 1..P$ are obtained (those which are used in the LDM)
2: **for** each alternatives A^i **do**
3: Evaluate $SCORE_{A^i} = f_{agg}(x_1^i, ..., x_n^i)$ using the relevant membership functions.
4: **end for**
5: Rank the alternatives $A^1, ..., A^m$ according to the score $SCORE_{A^i}$ to obtain the permuted preference set $\tilde{A}^1 \succeq \tilde{A}^2 \succeq .... \succeq \tilde{A}^m$.
6: Recommend the alternative $\tilde{A}^1$

In general, the aggregation function f_{agg} (that Algorithm 1 references) must satisfy the following properties: (i) if $a > b$ then $f_{agg}(w, a) \geq f_{agg}(w, b)$; (ii) $f_{agg}(w, a)$ is monotone in w; (iii) $f_{agg}(0, a) = ID$, where ID is the identity element; and (iv) $f_{agg}(1, a) = a$ (see [HH00] for further details of these properties).

Afterwards, once the alternative was recommended, it is stored with its LDM, as Figure 2 shows $\{S^*, \tilde{A}^1\}$ (where $\tilde{A}^1 = C$). The aim of storing the duple is to periodically monitor during runtime that the selected alternative still satisfies the LDM, and if not, a new alternative should be recommended. An alternative may cease to satisfy a model basically by two causes:

- the alternative drops its quality of service or,
- other functionally-equivalent alternatives have improved enough to make experts observing them to change of perception about the meaning of each word (they become more exigent)

5 Illustrative example

"Golden age" is an internet-of-things application that monitors different aspects of their subscribers using sensors (e.g. heart rate, GPS, to name a few) in order to minimize the time between their patients start suffering an episode and that their

families are notified or/and the "golden age" ambulance assists them. Our client requires that the application "Golden age" calls automatically to the patient's relatives using a highly available and fast telephony service This call will use a pre-recorded notification asking confirmation or rejection by pressing a telephone number button. Depending on how severe is the emergency, "Golden age" may notify all relatives at once and assigns immediately the ambulance or it may notify relatives one by one until one confirms that he/she can assist the patient.

The service market has approximately 1300 services of 59 different functional categories. They have been crawled from programmableweb[3], an open web catalog of APIs where only SOAP services have been considered. We consider two quality attributes represented by the following metrics: *response_time* and *availability*. Both metrics have been initially certified, whose values were observed by the experts to set up initially the parameters of the membership functions of each linguistic value of the ED. Let $\mathcal{A}$ be the set of 27 from 1300 alternatives capable to provide $\{Telephony\}$, which is comprised on this example by

$$\mathcal{A} = \{Alianza, CallfireVoiceBroadcast, \ldots, Voxbone\}.$$

Let $G = \{G^{(1)} = response\_time, G^{(2)} = availability\}$ be the two attributes under consideration. "Golden age"'s software engineers have agreed to use five linguistic values ($P = 5$) for each linguistic variable. For the *response_time* the linguistic values are $\{very\_fast, fast, medium, slow, very\_slow\}$, meanwhile for the *availability* the linguistic labels are $\{poor, fair, good, very\_good, excellent\}$. Software engineers also agree upon the constraints that *"the selected alternative should have at least a very_fast* response_time *and at least an excellent* availability*"*, in other words:

$$LDM = \mathcal{L}(LV^{response\_time\ telephony})\ AND\ \mathcal{L}(LV^{availability\ telephony}_{excellent})$$

The fuzzy numbers at setup time are: $\mu^{response\_time}_{very\_fast}(a = (0, 1), b = (700, 1), c = (970, 0))$, $\mu^{availability}_{excellent}(a = (91, 0), b = (94, 1), c = (100, 1))$.

At deployment time, the *Adaptive framework* obtains the ranked set of alternatives and recommends to its client the one with highest rank: $\{\tilde{A}^1 = \{CallfireVoiceBroadcast = 1.0\} \succ \tilde{A}^2 = \{CallfireHostedCallCenter = 0.5\} \succ \tilde{A}^3 = \{AngelOutBound = 0.5\} \succ \tilde{A}^4 = \{AcrossCommunications = 0.5\} \succ \tilde{A}^5 = \{PushBug = 0.5\} \succ \tilde{A}^6 = \{TelenorClicktoCall = 0.5\} \succ \tilde{A}^7 = \{DIDWW = 0.5\} \succ \ldots\}$ Assuming that "Golden age" accepts to use the $CallfireVoiceBroadcast$ service, then the pair $\{LDM, \tilde{A}^1\}$ is stored in order to monitor violation symptoms. The alternative $\tilde{A}^1 = \{CallfireVoiceBroadcast = 1.0\}$ has a *response_time* of 661 milliseconds and an *availability* of 96%.

After ten iterations, the experts that are observing the world have changing their perceptions about the word "very_fast" *response_time* of the functional category $Telephony$. The linguistic value "very_fast" of the *response_time* under observation has drifted with a positive trend: $\mu^{response\_time}_{very\_fast}(a = (0, 1), b = (540, 1), c = (601, 0))$. In this example, reader must assume that those services

[3] http://www.programmableweb.com

already selected have been "frozen"; thus, $\tilde{A}^1 = \{CallfireVoiceBroadcast\}$ still has a *response_time* of 661 milliseconds. Even that, because regularly experts are observing the alternatives, they updated the linguistic values' meanings and therefore when models are assessed, they are against of the latest updated meanings. Therefore, in this chase it is detected that the contract $\tilde{A}^1$ has degraded its satisfaction on at least a 50% and therefore a replacement is needed.

6 Conclusions

In this paper we have shown that by using a linguistic decision making approach to manage requirements@run.time, we are capable to detect "silent" requirements violations caused by the changes on the environment. As future work, we will prepare experiments with humans to validate empirically our results.

References

[AP07] Ardagna, D., Pernici, B. Adaptive service composition in flexible processes. *IEEE Transactions Software Engineering*, 33(6):369–384, June 2007.

[BBF09] Blair, G., Bencomo, N., France, R. Models@run.time. *Computer*, 42:22–27, 2009.

[BCG+10] Baresi, L., Caporuscio, M., Ghezzi, C., Guinea, S. Model-driven management of services. In *8th IEEE European Conference on Web Services (ECOWS)*, pages 147 –154, dec. 2010.

[BJP12] Bianculli, D., Jazayeri, M., Pezzè, M. *Matinée with Carlo Ghezzi: From Programming Languages to Software Engineering.* CreateSpace, 2012.

[BG10] Baresi, L., Ghezzi, C. The disappearing boundary between development-time and run-time. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 17–22, New York, NY, USA, 2010. ACM.

[BG11] Baresi, L., Guinea, S. Self-supervising BPEL processes. *IEEE Transactions on Software Engineering*, 37:247–263, 2011.

[Bod08] Bodenhofer, U. Ordering of fuzzy sets based on fuzzy orderings. part i: The basic approach. *Mathware & Soft Computing*, (15):201–218, 2008.

[Bon80] Bonissone, P. A fuzzy sets based linguistic approach: Theory and applications. In *Proceedings of the 12th conference on Winter simulation*, WSC '80, pages 99–111, Piscataway, NJ, USA, 1980. IEEE Press.

[CCG+12] Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Lo Presti, F., Mirandola, R. MOSES: A framework for QoS driven runtime adaptation of service-oriented systems. *IEEE Transactions Software Engineering*, 38(5):1138–1159, 2012.

[CGK+11] Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., Tamburrelli, G. Dynamic QoS management and optimization in service-based systems. *IEEE Transactions Software Engineering*, 37(3):387–409, 2011.

[CGK+12] Calinescu, R., Ghezzi, C., Kwiatkowska, M., Mirandola, R. Self-adaptive software needs quantitative verification at runtime. *Communications on ACM*, 55(9):69–77, September 2012.

[DB88] Degani, R., Bortolan, G. The problem of linguistic approximation in clinical decision making. International Journal of Approximate Reasoning 2 (2) (1988) 143 – 162.

[DVV92] Delgado, M., Verdegay, J., Vila, M. Linguistic decision-making models. *International Journal of Intelligent Systems*, 7:479 –492, 1992.

[EGM+09] Epifani, I., Ghezzi, C., Mirandola, R., Tamburrelli, G. Model evolution by run-time parameter adaptation. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 111–121, Washington, DC, USA, 2009. IEEE Computer Society.

[FGT12] Filieri, A., Ghezzi, C.,Tamburrelli, G. A formal approach to adaptive software: continuous assurance of non-functional requirements. *Formal Aspects of Computing*, 24:163–186, 2012. 10.1007/s00165-011-0207-2.

[Hal06] Hall, D. Automatic parameter regulation of perceptual systems. *Image and Vision Computing*, 24(8):870 – 881, 2006.

[HH00] Herrera, F., Herrera-Viedma, E. Linguistic decision analysis: steps for solving decision problems under linguistic information. *Fuzzy Sets and Systems*, 115(1):67–82, 2000.

[MK95] Mannion, M., Keepence, B. Smart requirements. *SIGSOFT Software Engineering Notes*, 20(2):42–47, April 1995.

[Mar10] Martínez, L. Computing with words in linguistic decision making: Analysis of linguistic computing models. *International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, 5–8, 15-16 Nov. 2010.

[MDH+09] Martinez, L., Ruan, D., Herrera, F., Herrera-Viedma, E., and Wang, P. Linguistic decision making: Tools and applications. *Information Sciences*, 179(14):2297 – 2298, 2009.

[Men01] Mendel, J. The perceptual computer: An architecture for computing with words. In *2001 IEEE International Fuzzy Systems Conference*, pages 35–38. IEEE, 2001.

[Men07] Mendel, J. Computing with words and its relationships with fuzzistics. *Information Sciences*, 177(4):988–1006, February 2007.

[PEP11] Pedrycz, W., Ekel, P., Parreiras, R.. *Fuzzy Multicriteria Decision-Making: Models, Methods and Applications*. John Wiley and Sons, Ltd., 2011.

[Par97] Parhami, B. Defect, fault, error,..., or failure? *Reliability, IEEE Transactions on*, 46(4):450–451, 1997.

[RM12] Rodriguez, R., Martinez, L., Herrera, F. Hesitant fuzzy linguistic term sets for decision making. Transactions on Fuzzy Systems 20 (1) (2012) 109–119.

[TA12] , Torres, R., Astudillo,H. Market-aware requirements. *A*nais do WER12 - Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, April 2012.

[Tor13] Torres, R. Mitigating the obsolescence of specification models of service-based systems. *35th International Conference on Software Engineering*, ICSE 2013: 1462–1464, San Francisco, CA, USA, May 18–26, 2013.

[TBA12] Torres, R., Bencomo, N., Astudillo, H. Mitigating the obsolescence of quality specifications models in service-based systems. *Second IEEE International Workshop on Model-Driven Requirements Engineering*, MoDRE 2012: 68–76, Chicago, IL, USA, September 24, 2012.

[Yager88] Yager, R. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions System Man Cybernetics*, 18(1): 183–190, 1988.

[Yag95] Yager, R. An approach to ordinal decision making. *International Journal of Approximate Reasoning*, 12(3–4):237 – 261, 1995.

[Zad65] Zadeh, L. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[Zad75] Zadeh, L. The concept of a linguistic variable and its application to approximate reasoning - i. *Information Sciences*, 8(3):199–249, 1975.

[Zad96] Zadeh, L. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103–111, May 1996.

[Zad99] Zadeh, L. From computing with numbers to computing with words-from manipulation of measurements to manipulation of perceptions. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 46(1):105119, 1999

[Zad01] Zadeh, L. A new direction in AI: Toward a computational theory of perceptions. *Artificial Intelligence Magazine*, 22(1):73–84, 2001.

[ZJ97] Zave, P., Jackson, M. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology*, 6(1):1–30, January 1997.