Addressing the QoS Drift in Specification Models of Self-Adaptive Service-Based Systems

Romina Torres
Universidad Técnica Federico Santa María
Valparaíso, Chile
romina@inf.utfsm.cl

Nelly Bencomo
INRIA Paris-Rocquencourt
Paris, France
nelly@acm.org

Hernan Astudillo
Universidad Técnica Federico Santa María
Valparaíso, Chile
hernan@inf.utfsm.cl

***Abstract*—Analysts elaborate precise and verifiable specification models, using as inputs non-functional requirements and assumptions drawn from the current environment studied at design time. As most real world applications exist in dynamic environments, recently there has been research efforts towards the design of software systems that use their specification models during runtime. The main idea is that software systems should endeavor to keep their requirements satisfied by adapting their architectural configurations when appropriated. Unfortunately, such specifications models use specific numbers (i.e. values) to specify non-functional constraints (NFCs) and may become rapidly obsolete during runtime given the drastic changes that operational environments can go through. The above may create circumstances when software systems are unaware that their requirements have been violated. To mitigate the obsolescence of specification models we have already proposed to use computing with words (CWW) to represent the NFCs with linguistic values instead of numbers. The "numerical meanings" of these linguistic values are computed from the measurements of non-functional properties (NFPs) gathered by a monitoring infrastructure. This article introduces the concept of "QoS-drift" to represent a significant degree of change in the "numerical meanings" of the linguistic values of the NFPs in the service market. We add to our former proposal a QoS-drift's vigilance unit to update linguistic values only when a QoS-drift is detected. Therefore, the new models are proactive and automatically maintained, what results in a more efficient assessment of run-time requirements' compliance under non-stationary environments. We validate the effectiveness of our approach using (1) a service market of 1500 services with two NFPs, (2) a synthetical QoS-drift and, (3) five systems built by different service compositions. We have detected that four of the five systems experienced requirements violations that would not have been detected without the use of our approach.**

***Index Terms*—concept drift, specification models, requirements@run.time, computing with words, fuzzy logic.**

I. Introduction

Service-based systems (SBSs) [1], which are systems built by compositions of (generally) third-party and distributed services, are under an environment that is intrinsically dynamic and open [2]. Mainly, due to the inherent competition between functionally-equivalent service providers new services may need to be selected during runtime. Therefore, a service composition or configuration (C) selected at design time to implement the requirements (R) of a SBS will not necessarily remain as a valid implementation during runtime. For instance, the services that compose C could drop their non-functional properties levels for which were selected. Thus, SBSs must be capable of adapting to changes dynamically [2].

Self-adaptive systems (SASs) [3] have been proposed to enable systems dynamic adaptation during runtime according the changes in the environment. Specification models used at design time are also used during runtime [4] making systems requirements-aware [5]. One advantage of requirements-aware systems is that they can monitor the requirements' compliance during runtime and trigger an adaptation when appropriated (for instance, replacing a violator configuration C by C').

Monitoring the compliance of the requirements (R) can be done according to Zave and Jackson [6] by monitoring the compliance of the specification model (S) [6]:

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

The specification model is derived by analysts from the requirements and the knowledge domain (K) that they have about the environment. Therefore, it is safe to believe that S will remain as a valid implementation of R if and only if, K does not drastically change from the moment that K was built until the moment the requirements compliance is assessed.

However, due to the facts that (1) specification models depends of K, (2) K is built by analysts from the perceptions that they have about the environment (before runtime), (3) SBSs are under a highly dynamic environment [2] and, (4) perceptual systems are frequently degraded by changing environments [7], specifications model S of SBS may become quickly and frequently obsolete during runtime. Thus, dynamic adaptive systems may not be aware that their requirements have been violated, because technically C still satisfies S however, S is not anymore a valid representation of R.

In the specific case of SBS, non-functional requirements are the main source of obsolescence of specification models. At design time, analysts gather the functional and non-functional requirements (FRs and NFRs respectively) of the system under development and by each FR they determine with stakeholders a set of non-functional constraints (NFCs). NFCs are typically expressed as linguistic terms (e.g. "fast" or "reliable") that map well the common use of natural language of stakeholders [8]. However, in order to make them verifiable, NFCs are typically refined by analysts into a set of precise specifications using numerical ranges of specific metrics [8] [9]. For instance, a requirement like {R: *a fast service to obtain weather forecast*}

could be reified to a specification model $\{S$: *a service capable to forecast weather with a response time less than 150 [ms] is acceptable*$\}$ given that the knowledge domain of analysts was $\{K$: *the average response time of services capable to forecast weather is approximately 200 [ms]*$\}$. However, due to the constant competition between functionally-equivalent services, in general service providers tend to improve their services in terms of non-functional properties (NFPs). Thus, making analysts change their perceptions about the numerical meanings of the NFCs and therefore, making S of SBSs become obsolete.

To mitigate the temporal obsolescence of S during runtime (addressing one kind of uncertainty identified in SASs [10]), we have already proposed [11] to represent S as abstract specification models (S^*) using the *computing with words* (CWW) [12] approach. On the one hand, NFPs are modeled as linguistic variables [13], whose values are linguistic values or concepts (as for example "fast") instead of precise numbers. On the other hand, analysts specify the NFCs of their models with these "words" or linguistic values. The main drawback of our former proposal was its computational cost, since the numerical "meanings" of the NFCs had to be computed every time a specification model was assessed.

Specification models become obsolete only when the domain knowledge drastically changes. Therefore, in this paper, we propose to improve our former proposal by recompute the numerical meanings of the NFCs only when a QoS-drift (concept drift in the non-functional properties) on the service market has been detected.

The reminder of this paper is organized as follows: Section II summarizes the related work needed to elaborate our proposal; Section III introduces the formal mechanism added to the CWW engine to detect QoS-drifts, and present a proposal to mitigate the obsolescence of specification models by using it; Section IV presents a case study; and finally, Section V concludes the paper.

II. Preliminares

Concept drift occurs when the context shifts induce changes in the target concept [14]. The degree of environment's change can be classified as random noise, random trends (gradual changes), random substitutions (abrupt changes), or systematic trends [15]. Concept drift is a notion taken from the machine learning research community (an Artificial Intelligence (AI) sub-area [16]); however, most current techniques assume that data conforms to a stationary distribution. In our case, NFCs depend on the stakeholders' perceptions about the "meanings" of the linguistic values of NFPs. They are typically expressed using natural language. We already proposed [11] to use CWW to represent the specification models. We modeled the NFPs as linguistic variables and their domain values as linguistic values. These "words" are the base to analysts to specify the NFCs of specification models with linguistic values instead of numbers [11].

Traditionally, Fuzzy logic has been used to solve CWW problems [12], where each "word" or linguistic value is modeled as a fuzzy set. These sets are built by the stakeholders' consensus or by using data-driven algorithms like fuzzy c-means [17]. Unfortunately, the sets built at design time do not necessarily remain valid during runtime (due to the inherent trend of improvement of the service market provoked by the competition between functionally-equivalent services). These sets (i.e. their "meanings") tend to (positively) shift through the time, which means that NFCs are temporally uncertain. As far as we know, there is not proposal to address the temporal uncertainty of "words" in CWW. We have found several proposals to reduce the number of "repeated" concepts in clustering methods, for example, where two or more fuzzy sets are merged into one if they represent the same concept [18] according to a similarity index. In the next Section, we adapt that technique to determine the local concept drift of a "word" during a given time, by computing the similarity between two fuzzy sets representing a common "word" at different instances of time.

III. Proposal

Since S_K is a valid representation of R if and only if K has not drifted, this article proposes to expand our former proposal and make systems aware of the *concept drift* in their environment (i.e. when K changes drastically to K').

To detect the QoS-drift we propose to determine temporarily the similarity between the two fuzzy sets that represent the same linguistic value but at different times. We have reused a technique to improve the quality of fuzzy clustering results [18], that as far as we know, has not been previously used to determine temporal similarity.

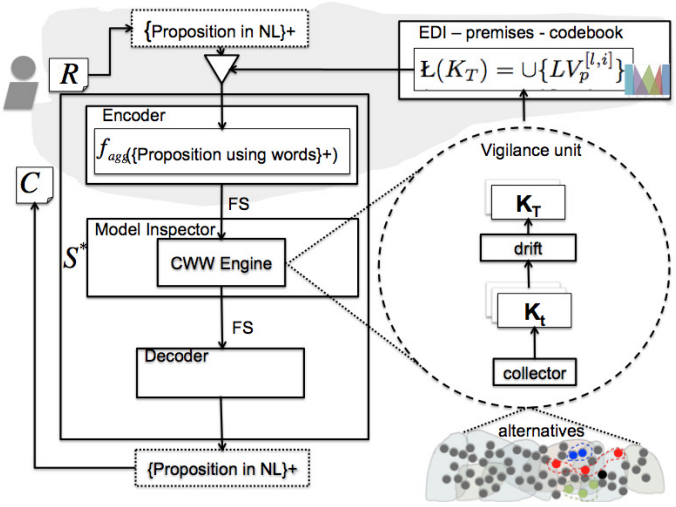

Fig. 1. Architecture of the recommendation framework to support (at runtime) monitoring of requirements' satisfaction by architectural configurations and recommendation of adaptations when appropriate

We model our proposal as a recommendation framework capable to serve several SBSs. Figure 1 shows the main components. Its core component implements the CWW approach (inspired on the perceptual computer [19]), which is constituted by the *encoder* and the *decoder* component, besides of

the *CWW engine*. All of them use a common *codebook* [19], which is in this case, comprised by all the concepts (linguistic values) over each NFP of each functional category of services. The novelty in this article with respect to our former proposal is that we introduce to the *CWW engine* a *vigilance unit* capable to detect the QoS-drifts occurring in the services market, in order to automatically update the "meanings" of the "words" or linguistic term set of the *codebook*.

The following subsections explain each stage, dividing them into those stages needed for the framework setup and those required to serve the SBSs.

A. Framework Setup: Choice of Linguistic Term Set and its Semantics

The linguistic term set with its semantics (Ł) is defined to be used by analysts refining their subjective judgments into linguistic decision models. In order to transform R into a S^*, subjective judgments (typically NFRs) must be expressed as linguistic values using one of the terms of the linguistic term set. Each linguistic term is a "word" with a "meaning". On the one hand, the "word" is a granule or fuzzy set [12], and on the other hand, the "meaning" is the numerical range of a specific metric that changes through time because it is built using the "perceptions" that stakeholders have about the domain. Then, if the environment changes, perceptions should too. Formally for SBSs, let $S = \{s_1, \ldots, s_m, \ldots, s_M\}$ be the set of service alternatives previously present in the environment. Let $Q = \{q_1, \ldots, q_l, \ldots, q_L\}$ be the L quality attributes describing each service in the market of services. Let $Q(s,t) = \{q_1(s,t), \ldots, q_l(s,t), \ldots, q_L(s,t)\}$ be the quantitative measurements of the L attributes of the service s at time t. Let $CS = \{CS_1, \ldots, CS_i, \ldots, CS_I\}$ be the functional taxonomy of services comprised by I functional categories. Let $CS = \{s_1, \ldots, s_j, \ldots, s_{J_I}\}$ be a functionally-equivalent set providing equivalent services in terms of functionality (also known as a functional category) comprised of J_I services.

Let $LV^{[l,i]}$ be the linguistic variable representing specifically the quality attribute l of the service set CS_i. Let $LV^{[l,i]} = \{LV_0^{[l,i]}, \ldots, LV_p^{[l,i]}, \ldots, LV_P^{[l,i]}\}$ be comprised by the P possible linguistic values that it can take (where not necessarily P is the same for each LV). We represent each linguistic value of each linguistic variable $LV_p^{[l,i]} \in$ Ł as a type-1 fuzzy set (or just fuzzy set) [12] (other kind of fuzzy sets could be chosen depending of the specific problem). Since typically service measurements of each non-functional attribute can be arranged in an ordinal scale, we have chosen for SBS's a linguistic symbolic computational model based on ordinal scales and max-min operators [20] (other linguistic computational models are also available [21]). For the sake of simplicity, we assume for each linguistic variable $LV^{[l,i]}$ $P = 5$ and the same linguistic values or "words", as follows: $\{LV_{poor}^{[l,i]}, LV_{fair}^{[l,i]}, LV_{good}^{[l,i]}, LV_{very\ good}^{[l,i]}, LV_{excellent}^{[l,i]}\}$

We also have chosen for each "word" ($LV_p^{[l,i]}$) a membership function of triangular form (Figure 2) defined as a triangular fuzzy number A in the interval $[0, 1]$. Technically, the membership function is the "meaning" of the "word". A can be

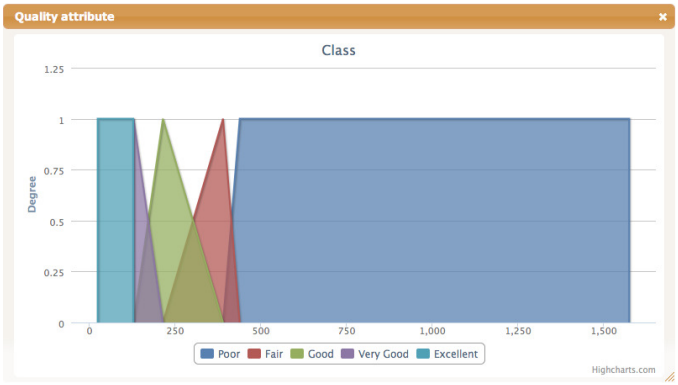

Fig. 2. Linguistic values of a linguistic variable

completely defined by (a_1, c, a_2), whose membership function is defined as follows:

$$\mu_A(x) = \begin{cases} \frac{x-a_1}{c-x_1} & \text{if } a_1 \leq x < c \\ \frac{a_2-x}{a_2-c} & \text{if } c < x \leq a_2 \\ 0 & otherwise \end{cases} \quad (2)$$

where $|(a_2 - a_1)|$ is the support of the fuzzy set and c is its centroid. To define each meaning of each "word" of the language Ł $= \cup\{LV_p^{[l,i]}\}$ K is needed. In other words Ł depends on K, and because K depends on the time, Ł also depends on the time, Ł(K_T) as the top right box of Figure 1 shows.

We have proposed already [22] a data-driven mechanism to obtain, by clustering techniques (fuzzy c-means), the several "meanings" of the linguistic values available in this language (also known as codebook in Perceptual Computing [23], explanatory database instantiated [24] or premises in the CWW community [24]). We chose a data-driven mechanism to set the "meanings" of the linguistic term set due to the high dimensionality of the alternative space and their non-functional attributes [22]. Using the fuzzy c-means, we initially obtained by each "word" the Gaussian membership function representing each linguistic value. Then, for the sake simplicity, we approximate here the three central "words" with a triangular membership function and the two words on the left and right extreme with a semi-trapezoidal function with infinite limit to the right or left respectively (as Figure 2 shows). Therefore, let $L(K_T) = \cup\{LV_p^i\}$ be the language (linguistic term set with its semantic) that will support analysts during later stages on to transform their requirements models R into S^* (a linguistic decision model LDM).

B. Framework Setup: Choice of Linguistic Information Aggregation Operator

Zadeh [24] discusses several linguistic aggregation operators that can help analysts to build their LDMs. In the particular case of SBSs, we have chosen the **disjunction** and **conjunction** linguistic aggregation operators, meaning that when a FR is constrained in a LDM, analysts can aggregate several NFCs using *AND* and *OR* operators. The resolution

process performed inside the CWW engine (at later stage) depends on the type of linguistic aggregation operator; more details can be found in [24] regarding how fuzzy constraints are propagated and how defuzzification is conducted for each kind of aggregation.

C. Framework Runtime: Automatic Synchronization of the "Meanings" of "Words" with the Changing Market

If functionally-equivalent alternatives are competing in terms of non-functional properties, the stakeholders' perceptions about what means each "word" (in this case quality level) will change. In this scenario, granules representing each linguistic value of each linguistic variable will tend to shift through time in a direction where "words" are more demanding. For instance, Figure 7 shows how the "meanings" of the "words" {"poor","fair","good","very good","excellent"} of a specific linguistic variable of a specific functional category have drifted to the left from period of time t_1 to period of time t_2 because this quality attribute in this particular category is of the kind "the smaller, the better", then it is expected that services competing in terms of this attribute tend to improve it by lowering its value.

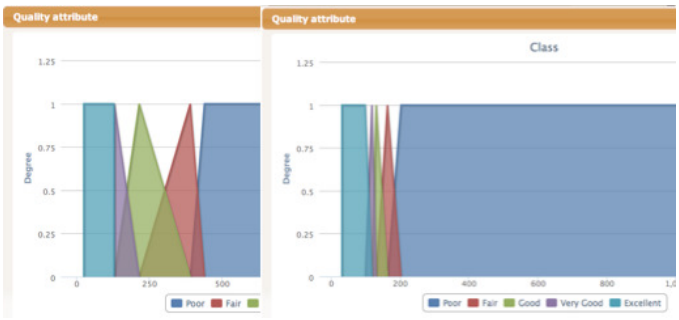

Fig. 3. Local drift of the linguistic values of a non-functional property (*response time*) of a functionally-equivalent set of services

In the center of Figure 1's left side, the *Model Inspector* surrounds the *CWW engine*. The *Model Inspector* allows this framework to support analysts in finding initial configurations or adapting the current ones when requirements are violated during runtime. Inside the *Model Inspector* subsystem, the *vigilance unit* component is constantly aware of the alternatives in the (services) market, gathering instant information K_t about the alternatives' non-functional measurements. This *vigilance unit* builds the domain knowledge K_T (which is comprised by the "meanings" of all the "words") from the perceptions drawn from the accumulated observations of the environment (using fuzzy c-means).

During runtime, this vigilance unit will be periodically verifying that the current knowledge domain K_T is not outdated by monitoring the global drifts on K_T. A global drift occurs if and only if the percentage of granules that have locally drifted between two, not necessarily consecutive, periods of time is greater or equal than a pre-tuned threshold. We define the local drift of a granule as a quantization of how much different are the fuzzy sets representing it at different times.

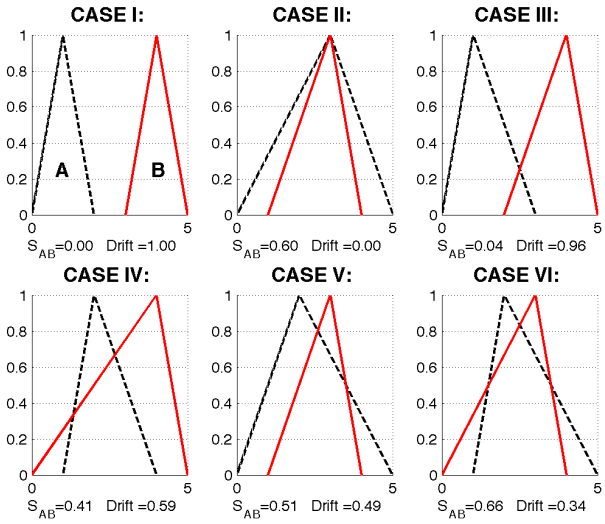

Fig. 4. Six different cases to calculate the similarity of two fuzzy set with triangular shape

For our special case, in which the fuzzy sets have a triangular shape, we reuse the similarity index defined in [18], where authors defined this index to determine (not temporarily) when two fuzzy sets A_t and B_t are too similar and therefore they should be merged into just one:

$$S_{AB} = \frac{M(A \bigcap B)}{M(A) + M(B) - M(A \bigcap B)} \tag{3}$$

where $\bigcap$ and $\bigcup$ denote the intersection and union of fuzzy sets A and B, respectively, $M(\cdot)$ is the size of the fuzzy set, and $0 \leq S_{AB} \leq 1$. We define that a granule has locally drifted if and only if $Drift_{AB} = \kappa(1 - S_{AB})$ where $Drift_{AB} > threshold$ is one of the six cases which are shown in Figure 8, whose formulas can be found in [18]. The variable κ takes the value 1 if c_A is located at the left side of c_B and takes the value -1 in the inverse case. If $c_A = c_B$ then κ is 0. In order to avoid false drifts, κ should not rely only in the value of at a single period of time but should consider the trend over its historical values through the time.

The *vigilance unit* determines that a global drift has occurred by (i) gathering the latest measurements from the alternatives (K_t), (ii) computing offline the new "meanings" of all the "words" according to K_t using a data-driven algorithm and, (iii) determining by each "word" if a local drift has occurred between the fuzzy sets obtained using K_t and the previously existent in K_T and, finally by (iv) determining if the number of local drifts is greater than the defined threshold. When a global drift is confirmed, then, the "meanings" obtained using K_t replace the old outdated "meanings" of K_T.

D. Framework Runtime: Defining the Specification Model S^*

During this phase, the analysts start by refining the R into S^* in order to use it at design or deployment time to select a C that implements S^* and therefore R, as well as during runtime to verify the satisfaction of R by a particular C. In

order to facilitate the understanding of this section, we explain how analysts refine a simple R of a SBS: {A quite fast service and highly reliable capable of sending email}.

As the left side of Figure 1 shows, the LDM is built by transforming R written as a set of propositions in natural language (NL) ($\{propositions\ in\ NL\}+$) into a set of aggregated propositions using "words" ($f_{agg}(\{propositions\ using\ words\}+)$) by encoding it with the available "words" in Ł(K_T). Analysts reify ($\{propositions\ in\ NL\}+$) into ($\{propositions\ using\ words\}+$) by (i) interpreting constraints as "quite fast" as { a response_time at least "very good" } (where "very good" is the p-th linguistic value of the non-functional attribute l of the functional category CS_i, $LV_p^{[l,i]}$). Then, they repeat the same process by each NFC (e.g. "highly reliable" as { a *reliability* at least "excellent" }.

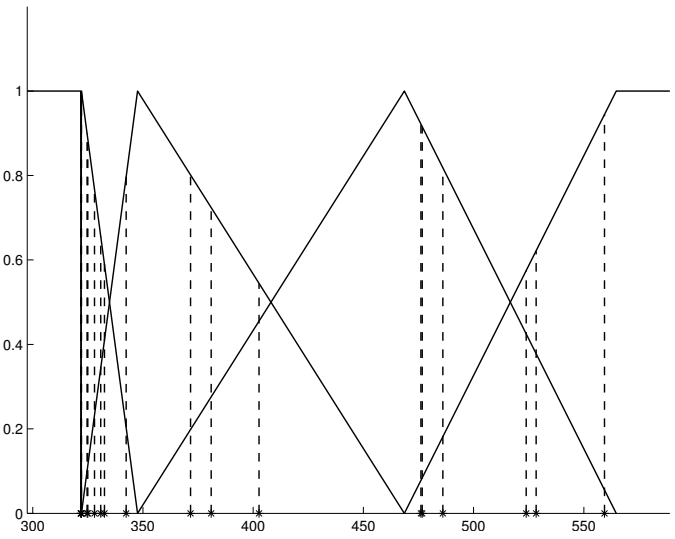

Fig. 5. ${\mu_{excellent}^{[RT,Science]}}^{K_T}(c(C_k, K_{t_u}))$

If there are more than one NFC or FR, analysts assign to each one a relative importance (between "high", "medium", "low") and then aggregate them by using the operators defined in section III-B. Then, the output of the function f_{agg} is processed by the *encoder* component, whose output is S^* which is expressed in terms of the fuzzy sets that represent each "word" or NFC used in the model.

Let C_k be the k^{th} architectural configuration comprised of a composition of O services $\{s_1, \ldots, s_o, \ldots, s_O\}$, with $O \leq X$. The *Model Inspector* receives as a model representation of S^* a fuzzy multi-criteria decision making that can be used by the *CWW engine* to assess architectural configuration alternatives C_k using the measurements obtained at time t but under the conditions of K_T ($S^*(C_k, K_t, K_T)$) as follows:

$$\sum_{x=1}^{X} v_x \left(\sum_{y=1}^{Y_X} w_{yx}\ {\mu_p^{[y,x]}}^{K_T}(c(C_k, K_t)) \right) \mathbb{I}(C_k, FR_x) \quad (4)$$

where X is the number of FRs in the model and Y_X is the number of non-functional properties constrained by each FR (where not necessarily all the FRs will have the same numbers of constraints). Let $v_x \in V = \{v_1, \ldots, v_x, \ldots, v_X\}$ be the relative importance of the FR_x in the model, $w_{yx} \in W_x = \{w_{1x}, \ldots, w_{yx}, \ldots, w_{Yx}\}$ be the relative importance of the non-functional property y for the FR_x in the model. Let $\mathbb{I}(C_k, FR_x)$ be the indicator function that returns 1 if there is a service $s_o \in C_k$ providing the required functionality x. Let $c(C_k, K_t)$ be a function that obtains the measurement values of the service $s_o \in C_k$ at time t. Let $\mu_p^{[y,x]}$ be the membership function of the NFC (the *minimal acceptable class* [1] that a service $s_o \in C$ should provide). Let ${\mu_p^{[y,x]}}^{K_T}(c(C_k, K_t))$ be the membership degree of the measurement of service $s_o \in C_k$ at time t to the (at least) fuzzy set representing the NFC $LV^{[y,x]}$ at time t. The output of the *Model Inspector* is the belonging degree to the type-1 fuzzy sets: *acceptable solution* and *non acceptable solution* . Then, this crispy value is decoded back as a set of proposition in NL, which in this case, is just a recommendation of what alternative to select (at system's deployment time when a configuration C must be selected to implement R) or the assessment of a particular configuration C (at system's runtime when it is necessary to verify the C is still R's compliant).

IV. Case Study

We have built a service dataset from the well-known open catalogue ProgrammableWeb (PW),[2]by selecting only 150 APIs (SOAP services) of 10 different functional categories (provided by the catalogue). Services at PW present high QoS variability because they may be reused by several mashups at once. We have built a collector to gather the data from PW and a certifier component over the SoapUI project[3] to consume the services regularly and obtain measurements for two metrics: *response time* (RT) and *throughput* (Thr).

Due to space constraints, we only show a case study with a SBS and a single requirement R mapped into S^*: {a service to "handle proteins" with a $response\ time$ at least "excellent"}. According to PW, only 19 SOAP services are available to perform several operations with proteins, all of them belonging to the *Science* functional category. Since there is only one constraint, its importance is set to high ("H"). The model $S^*(C_k, K_t, K_T)$ entering the *CWW engine* is

$$\left({\mu_{excellent}^{[RT,Science]}}^{K_T}(c(C_k, K_t)) \right) \mathbb{I}(C_k, FR_{Science})$$

For each $s_o \in Science$ in K_t, there is an architectural configuration C_k (in this case of only one service each) that is a potential candidate to implement S^* (and therefore

[1]Notice that we work with the *minimal acceptable class*: if a configuration C satisfies S^* with a *better class* (e.g. "excellent"), its satisfaction will be maximum. The CWW engine does not return the optimal solution but a set of solutions that satisfy the restrictions; if they have the same satisfaction degree, they cannot be distinguished by mere comparison.

[2]ProgrammableWeb: http://www.programmableweb.com/

[3]SoapUI: http://www.soapui.org/

R). In this case, there are 19 candidate solutions, C_k with $k = 1 \ldots 19$.

At epoch 1 or time t_u, Figure 5 shows the belonging degree of each configuration C_k to each *response time* level using the membership functions computed using the domain knowledge K_T and the numerical measurements provided by K_{t_u}). By inspection, only those services that have a *response time* lesser than 325 [ms] are potential solutions, others are immediately discarded.

Between times t_u and t_v (or epoch $u = 1$ and $v = 6$) we certify the dataset six consecutive times every four hours, with $u << v$. Between each consecutive pair of measurements K_t and K_{t-1} with $u \leq (t-1) < v$ we compute the local drift of each linguistic value in the codebook and its trend (in this case a negative sign of the drift in the case of *response time* indicates that it is improving and vice-versa). The global drift of the environment is computed over all the local drifts of all the available linguistic values. Do notice that a global drift could be an enmasked random noise; if so, to determine that a global drift has occurred, the drift and its trend should be exhibited in several subsequent epochs. Assuming that the only available linguistic values are those that represent the different levels of the *response time* of the category *Science*, Figure 6 shows the computed drift at each epoch. In the first epoch, we can observe that the 25% of the services improve significantly their certified values, indeed more than the 50% of the data shows this improvement. This improvement trend is even more notorious in epoch 4 and 6. At epoch 2 we can see that only less than the 25% of the services have improve themselves in terms of *response time* and at epoch 3 we can see that no one service improve itself, and actually at epoch 5 we can see that all the services have dropped their levels. Then, in order to prevent false global drift we need to evaluate if there is a trend of drift, which in the case of this real dataset and these 24-hours of measurements is not conclusive.

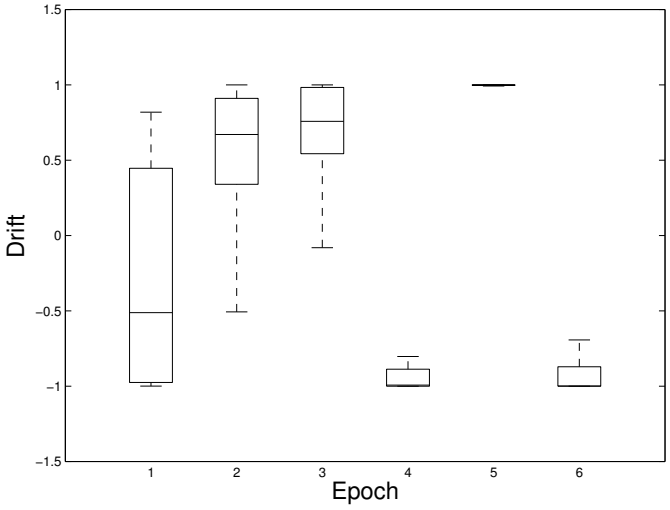

Fig. 6. Drift of *response time* QoS metric of category *Science*

Therefore, in order to continue with our case study we swap

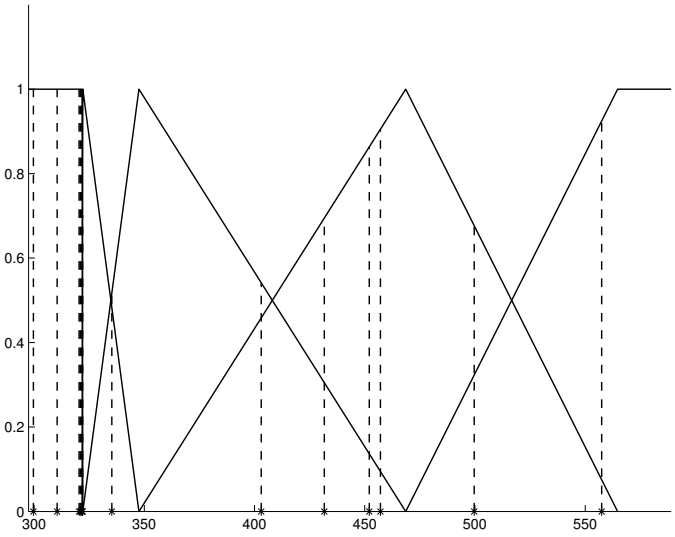

Fig. 7. $\mu_{excellent}^{[RT,Science]^{K_T}}(c(C_k, K_{t_v}))$

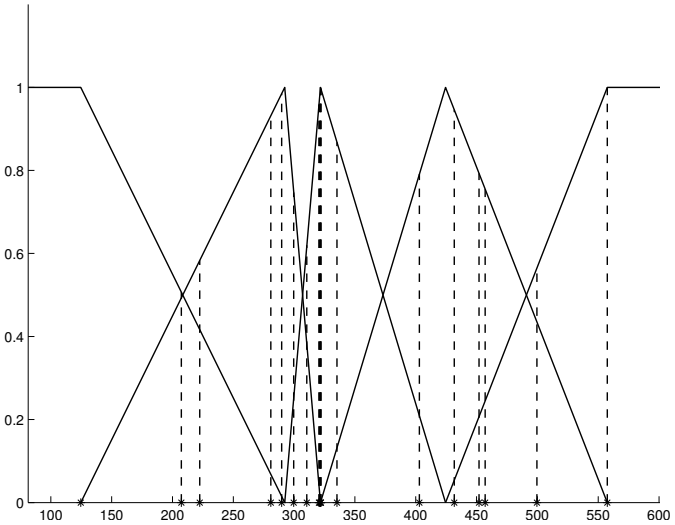

Fig. 8. $\mu_{excellent}^{[RT,Science]^{K'_T}}(c(C_k, K_{t_v}))$

to a synthetical dataset, in which we are sure that a global improvement drift. Comparing Figures 7 and 8 we can see how most of the fuzzy sets have drifted showing a trend of improvement, for instance: { "excellent" from $(0, 322, 325)$ to $(0, 125, 290)$ } and {"very good" from $(322, 325, 350)$ to $(125, 290, 325)$ }.

If, despite of the global drift, we do not update the "meanings" of the "words" we would be evaluating the measurements of the different $s \in C_k$ using the last K_{t_v} as Figure 7 shows, where we can see that still C_k exhibiting a *response time* lesser than 325 are potential candidates, and lesser than 322 are indistinguishable (according to this model) solutions. However, if we do consider the drift, and we update the codebook, then the CWW engine evaluates different each C_k as Figure 8 shows, where now only C_k exhibiting *response time* lesser than 290 [ms] are potential solutions and lesser

than 125 [ms] are indistinguishable valid solutions. If the threshold is set up in 0.5 only the 20% of the acceptable solutions according to K_T are still acceptable, even when K_T has drastically changed from K to K'. Then, if for instance, these alternatives would have been selected alternatives C_k to implement different instances of the R of this section, with our technique we would be able to detect that more than the 80% are violating their R (not because they have drop their QoS but because other improve significantly, making stakeholders' perceptions about the meaning of the NFCs change). If the QoS-drift would be not addressed, these violations would be not detected.

V. Conclusions and Future Work

This paper has introduced an approach to mitigate the temporal obsolescence of specification models by addressing the global drift of non-functional constraints. Specification models are represented as linguistic decision models over a computing-with-words (CWW) architecture, where the meanings of the "words" are synchronized with the environment each time a global QoSdrift is detected. The notion of *QoS-drift* was introduced, based on concept drift.

With our approach, stakeholders are relieved from manually maintaining models. Instead, models are self-maintained -by combining the CWW architecture and a vigilance unit that monitors QoS drifts of the alternatives- to maintain the NFCs in sync with the changing environment.

The approach was validated with a study that has been conducted using a synthetic dataset of 1500 services and 2 QoS attributes. We have applied both our approach and a traditional one, and have made the comparisons between both approaches. The results show that for a given set of changes in the knowledge about the environment, the traditional approach identified five configurations as valid (as they satisfied the original specification) however, our QoS-drift approach detected that only one of them was actually valid.

Ongoing work aims to release a dataset presenting datasets of global QoS drifts to the community, gathered by a service certifying tool over extended periods of time.

This approach allows analysts to use relative concepts to create fixed specifications at design-time, while preserving the flexibility afforded by dynamic changes in the "meaning" of NFCs as specific values. The above allows the system to effectively assess run-time requirements compliance in non-stationary environments.

Acknowledgements

This work was partially funded by UTFSM DGIP (PIIC and 241167), BASAL FB0821, the EU Marie Curie Requirements@runtime and the EU Connect project.

References

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1109/MC.2007.400

[2] L. Baresi, E. Di Nitto, and C. Ghezzi, "Toward open-world software: Issue and challenges," *Computer*, vol. 39, no. 10, pp. 36–43, Oct. 2006.

[3] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intelligent Systems*, vol. 14, no. 3, pp. 54–62, May 1999.

[4] G. Blair, N. Bencomo, and R. B. France, "Models@run.time," *Computer*, vol. 42, pp. 22–27, 2009.

[5] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, "Requirements-aware systems: A research agenda for RE for self-adaptive systems," in *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, ser. RE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 95–103.

[6] P. Zave and M. Jackson, "Four dark corners of requirements engineering," *ACM Transactions on Software Engineering and Methodology*, vol. 6, no. 1, Jan. 1997.

[7] D. Hall, "Automatic parameter regulation of perceptual systems," *Image and Vision Computing*, vol. 24, no. 8, pp. 870 – 881, 2006.

[8] M. Mannion and B. Keepence, "Smart requirements," *SIGSOFT Software Engineering Notes*, vol. 20, no. 2, pp. 42–47, Apr. 1995.

[9] V. R. Basili and H. D. Rombach, "The tame project: towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, vol. 14, no. 6, pp. 758–773, Jun. 1988.

[10] N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7475, pp. 214–238.

[11] R. Torres, N. Bencomo, and H. Astudillo, "Mitigating the obsolescence of quality specifications models in service-based systems," in *Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE*, Chicago, United States, Sep. 2012, pp. 68–76.

[12] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[13] ——, "The concept of a linguistic variable and its application to approximate reasoning - i," *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975.

[14] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996. [Online]. Available: http://dx.doi.org/10.1023/A:1018046501280

[15] L. Kuncheva, "Classifier ensembles for changing environments," in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3077, pp. 1–15.

[16] M. Harman, "The role of artificial intelligence in software engineering," in *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012 First International Workshop on*, june 2012, pp. 1–6.

[17] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[18] G. Leng, X.-J. Zeng, and J. A. Keane, "A hybrid learning algorithm with a similarity-based pruning strategy for self-adaptive neuro-fuzzy systems," *Applied Soft Computing*, vol. 9, no. 4, Sep. 2009.

[19] J. M. Mendel and D. Wu, "Challenges for perceptual computer applications and how they were overcome," *IEEE Computational Intelligence Magazine*, vol. 7, no. 3, pp. 36–47, 2012.

[20] R. Yager, "Aggregation of ordinal information," *Fuzzy Optimization and Decision Making*, vol. 6, no. 3, pp. 199–219, Sep. 2007.

[21] R. M. Rodriguez and L. Martínez, "An analysis of symbolic linguistic computing models in decision making," *International Journal of General Systems*, vol. 42, no. 1, pp. 121–136, 2013.

[22] R. Torres, H. Astudillo, and R. Salas, "Self-adaptive fuzzy QoS-driven web service discovery," in *Proceedings of IEEE Conference on Services Computing*, ser. SCC '11. IEEE Computer Society, 2011, pp. 64–71.

[23] J. Mendel, "The perceptual computer: An architecture for computing with words," in *2001 IEEE International Fuzzy Systems Conference*. IEEE, 2001, pp. 35–38.

[24] L. A. Zadeh, "Fuzzy logic = computing with words," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 2, pp. 103–111, May 1996.