ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 11 (2018) 272–281

An architecture based on computing with words to support runtime reconfiguration decisions of service-based systems

Romina Torres [1] , Rodrigo Salas [2] , Nelly Bencomo [3] , Hernan Astudillo [4]

[1] *Faculty of Engineering, Universidad Andres Bello, Viña del Mar, Chile*

E-mail: romina.torres@unab.cl

[2] *Biomedical Engineering School, Universidad de Valparaiso, Valparaíso, Chile*

E-mail: rodrigo.salas@uv.cl

[3] *School of Engineering and Applied Science, Aston University Birmingham, UK*

E-mail: n.bencomo@aston.ac.uk

[4] *Informatics Department, Universidad Técnica Federico Santa María, Valparaíso, Chile*

E-mail: hernan@inf.utfsm.cl

Received 20 October 2017

Accepted 27 October 2017

Abstract

Service-based systems (SBSs) need to be reconfigured when there is evidence that the selected Web services configurations no further satisfy the specifications models and, thus the decision-related models will need to be updated accordingly. However, such updates need to be performed at the right pace. On the one hand, if the updates are not quickly enough, the reconfigurations that are required may not be detected due to the obsolescence of the specification models used at runtime, which were specified at design-time. On the other hand, the other extreme is to promote premature reconfiguration decisions that are based on models that may be highly sensitive to environmental fluctuations and which may affect the stability of these systems. To deal with the required trade-offs of this situation, this paper proposes the use of linguistic decision-making (LDM) models to represent specification models of SBSs and a dynamic computing-with-words (CWW) architecture to dynamically assess the models by using a multi-period multi-attribute decision making (MP-MADM) approach. The proposed solution allows systems under dynamic environments to offer improved system stability by better managing the trade-off between the potential obsolescence of the specification models, and the required dynamic sensitivity and update of these models.

Keywords: service-based systems, quality-of-service, linguistic decision making models, computing with words

1. Introduction

Service-based Systems (SBSs) are built by composing distributed and heterogeneous services that are capable of partially or fully satisfying their functional and non-functional requirements [1]. Most SBSs depend on external third parties services. In contrast to software components [2], these services are

out of the control of the systems integrators' jurisdiction [3]: they are deployed on provider-site, they are not exclusive, they may serve several clients at the same time and, therefore, they may change in uncertain and non-predictable ways.

Due to the proliferation of services, non-functional requirements have become crucial in the service selection process. Today, the problem of SBSs has changed from finding a service that is capable of satisfying a functional requirement to finding which one should be selected from several functional-equivalents. Therefore, services are selected according to how well they satisfy the non-functional constraints (NFCs) of the specification model.

At design time, the non-functional requirements are transformed into concrete and precise NFCs using ranges of numerical values. These numbers are provided by experts whose perceptions are shaped by their own skills, experience, and/or level of knowledge about the domain (the current characteristics of the alternatives). These models are used at design and deployment time to select the services.

Pre-runtime verification of the configurations' satisfaction of the specification models cannot give the desired guarantees that are needed post-deployment[3] because runtime changes are inherent[4]. For example, during runtime, a previously selected service may become no longer the right alternative to be used because: (1) it has dropped its quality-of-service (QoS) [4]; or, and even more difficult to detect, (2) other functional equivalents became better alternatives than the selected option, making experts and users' perceptions change the meaning (range of values) of a constraint (e.g. which services are "fast"). Thus, even when the selected service still satisfies the model, it is no longer a valid alternative. Therefore, proposals to assess concrete specifications models (using crisp numbers) under dynamic and changing environments (with fluctuations, outliers and/or random trends) may miss the required reconfigurations because the models, and precisely the NFCs' meanings, may already be obsolete.

In the specific case of SBSs, they need continuous verification to check that the current service configurations still satisfy the specification models because the available knowledge about the service market (and specifically about the services' QoS) before deployment was either incomplete and uncertain or it may have changed during execution.

To address this issue, in general, it has been proposed that specification models should evolve as requirements or environments evolve [9,10] by synchronizing the models' parameters during runtime [11,12,6,10,7]. Satisfaction to these models should be continuously verified [6,8,5]. Several different implementations have been developed using the previous concept. For instance, the MOSES framework [11] modeled changing aspects as average statistic estimators, while the KAMI framework [21] uses Markov chains to periodically recompute the parameters' values and predict violations.

However, under the dynamic and changing environments in the service market, when the obsolescence issue is addressed, the stability of the SBSs may be compromised because they will tend to perform premature reconfiguration decisions due to the oscillating QoS's behavior in the service market. In extreme cases, configurations that had previously been discarded may rapidly become valid again, which means that the cost of reconfiguration was unnecessary.

We have previously proposed in our ongoing work that SBSs' owners should represent specification models as linguistic decision making (LDM) models to specifically represent the constraints over services' QoS as constraints over linguistic values instead of precise numbers [13,14,22]. The models' satisfaction are frequently assessed during runtime by a *CWW engine* that addresses the models' obsolescence. Unfortunately, under dynamic and changing environments, with fluctuations and/or outliers, SBSs are too sensitive to fluctuations giving place to premature reconfiguration decisions and this affects the stability of these systems.

In this work, to enable us to address both issues at the same time (i.e. the obsolescence of specification models and the high sensitivity for reconfiguration of SBSs under dynamic environments), we complement our previous proposal with a CWW engine using a multi-period multi-attribute decision making (MP-MADM) resolution approach[35] to as-

sess reconfigurations against of specification models, which evaluates and aggregates models' satisfaction in several periods in order to determine when a reconfiguration is really needed.

The rest of the paper is organized as follows. In Section 2, we review the LDM models and the CWW architecture as the computational basis in LDM processes. In Section 3, we present our proposal. In Section 4, we first introduce an example and then we present our experiment to show how well our proposal under dynamic and changing environments mitigates the degradation of the stability of systems by reducing the number of premature decisions while at the same time ameliorating the problem of obsolescent specifications of models. Finally, in Section 5 we conclude the paper.

2. Background

Multiple-attribute decision making (MADM) has been widely and successfully used to support decision making in multiple areas. The Multi-period multi-attribute decision making (MP-MADM) approach is an extension of the MADM where the decision should also be taken by using the historical data. Typically, most decisions that are made in the real world take place in an environment in which the goals and constraints are not known with precision [26] and, therefore, the problem cannot be precisely represented using crisp values [27]. Typically, these problems involve human perceptions using linguistic broad constructions (e.g. "nice", "a lot", "a few", "comfortable", to name a few). LDM models [28] have been successfully used to solve ill-structured decision problems in a wide range of practical problems, such as personnel evaluation, online auctions, venture capital supply chain management and medical diagnostics. However, these applications present two challenges: (1) they cannot be solved with the classical tools of decision theory [29,30]; and, (2) they exist under uncertain environments[31,32,33,34]. In order to meet these MADM challenges, a four-stage linguistic resolution scheme [17] has been proposed: the selection of the linguistic term set and its semantic, the selection of the aggregation operator of linguistic information, and the aggregation and exploitation phases.

CWW [15] has been applied as the computation basis in LDM processes [16]. It proposes a methodology of reasoning, computation and decision making in which "words" from natural language are used. Several CWW-based architectures have been proposed [15,18,19,20]. Its main components are an explanatory database (*ED*), a *CWW engine*, an *encoder* and a *decoder*.

3. Proposed Solution

Figure 1 shows the dynamic CWW architecture to support SBSs' owners in their reconfiguration decisions during runtime under changing environments. Complementary to the basic components of a CWW architecture, this proposal needs an additional component—the *collector*—, which is responsible for periodically monitoring the QoS values of the services in the marketplace and collecting the QoS measurements.

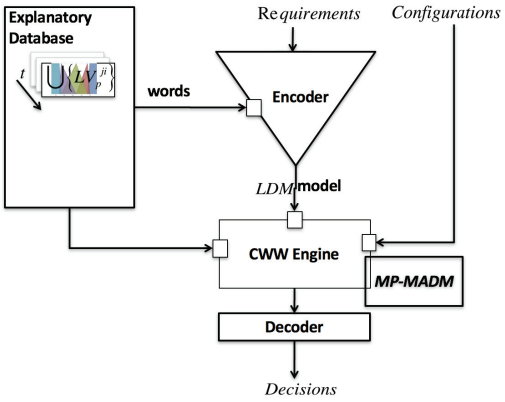

Fig. 1. A dynamic CWW architecture to support SBSs' owners in performing reconfiguration decisions during runtime. The architecture is composed by the *ED* (upper-left), the *encoder* (upper-right), the *CWW engine* with a decoder (lower-right). The *collector* is a key part that continuously senses the service market so that it can measure the QoS.

The left side of Figure 1 shows the *ED*, which is composed of the set $\cup\{LV_p^j\}$, where each term LV_p^j is a linguistic value of the $j-th$ linguistic variable LV^j. Linguistic values are represented using a linguistic evaluation scale S, where each value is represented by a word s_p or with a membership function μ_p^j. In the case of SBSs, a linguistic variable is, for instance, the response time of services capable of performing a certain functionality; in which case,

their linguistic values could be "worst", "normal', 'and "best".

The *encoder* in the center of Figure 1 supports humans in the process of reification of the requirements into specification models, which are represented as LDM models by using the available "words" (i.e. linguistic values) provided by the *ED*. This component supports the first two phases of the linguistic resolution scheme.

We reuse a grammar that was proposed in our previous work [22] to support SBSs' owners in building the LDM models, which is as follows:

$$
\begin{array}{rl}
I ::= & \langle\text{primary term}\rangle|\langle\text{composite term}\rangle \\
\langle\text{composite term}\rangle ::= & \langle\text{unary relation}\rangle\langle\text{primary term}\rangle| \\
& \langle\text{primary term}\rangle\langle\text{binary relation}\rangle\langle\text{primary term}\rangle| \\
& \langle\text{primary term}\rangle\langle\text{binary relation}\rangle\langle\text{composite term}\rangle \\
\langle\text{primary term}\rangle ::= & \langle\text{word}\rangle|\langle\text{unary relation}\rangle\langle\text{primary term}\rangle \\
\langle\text{word}\rangle ::= & LV_1^j|...|LV_p^j|...|LV_P^j \\
\langle\text{unary relation}\rangle ::= & \mathscr{L}|\mathscr{M}|NOT \\
\langle\text{binary relation}\rangle ::= & AND|OR|LWA|LOWA
\end{array}
$$

The "*at least*" ($\mathscr{L}$) and "*at most*" ($\mathscr{M}$) are unary ordering-based modifiers [23]. Given the label s_q associated to the linguistic value LV_p^j (p fixed), the fuzzy sets *"at least* LV_p^j*"* and *"at most* LV_p^j*"* (abbreviated as $\mathscr{L}(s_q)$ and $\mathscr{M}(s_q)$) are defined as follows:

$$
\begin{aligned}
\mathscr{L}(s_q)(x) &= \sup\{\mu_p^j(y) \text{ such that } y \in \mathscr{X} \text{ and } y \preceq x\} \\
\mathscr{M}(s_q)(x) &= \sup\{\mu_p^j(y) \text{ such that } y \in \mathscr{X} \text{ and } x \preceq y\}
\end{aligned}
$$

where $\preceq$ is a crisp ordering on $\mathscr{X}$.

The LDM model is a set of aggregated constraints that are written in terms of words (which are extracted from the *ED*) and the operators. $\delta_j()$ is a function denoting the resulting LDM for the constraint related to the $j-th$ QoS measurement, where the function δ_j evaluates the level of compliance to the $j-th$ constraint.

Based on the monitored data, the *ED* component updates the meanings of the linguistic terms (words) at current time t. At setup time, these values are obtained from the first sample. The *collector* component is either implemented manually (i.e. by humans experts) or automatically. The *ED* periodically recomputes the parameters of the membership functions of each linguistic values according to the available data. For instance, quantiles or the fuzzy c-means algorithm can be used, as in our previous work [24].

During runtime, the *CWW engine* component receives both the LDM model and the configuration of services that are currently in use and the available set of configurations alternatives. Based on this information, the *CWW engine* ranks the alternatives including the configuration in use by using the MP-MADM approach. In the MP-MADM approach[35], the *time-based fuzzy assessment matrix*, $\mathbf{R}$, is constructed using the time sequence of membership functions $\{\mu_1^j(t),...,\mu_P^j(t)\}_{t=1..\tau}$ and the time sequence of QoS measurements for each alternative $\mathscr{A}_i$; that is, $\{x_{i1}(t),...,x_{iJ}(t)\}_{t=1..\tau}$. Let r_{ij}^t be a fuzzy value that represents the assessment of the level of compliance of the alternative $\mathscr{A}_i$ to the constraint δ_j over the $j-th$ QoS attribute at time t; $\mathbf{R}^t = (r_{ij}^t)_{m\times n}$ is a matrix of size $m \times n$ of these fuzzy values, Δ is the size of a time window and τ the current time. Thus, the *time-based* fuzzy assessment matrix is given by the following equation:

$$
\mathbf{R} = \mathbf{R^t}_{[\tau,\tau-\Delta+1]} = (r_{ij}^t)_{m\times n\times\Delta} \tag{1}
$$

Afterwards, the temporal assessments of $\mathbf{R}$ are aggregated using the dynamic weighted average [35] (*DWA*) operator of equation (2). The temporal aggregated assessment at time τ, with a time-window of size Δ, of the j-th QoS attribute of the alternative $\mathscr{A}_i$ is given by:

$$
\begin{aligned}
a_{ij} &= DWA(r_{ij}^\tau,\ldots,r_{ij}^{\tau-\Delta+1}) \\
&= \omega^j(\tau)r_{ij}^\tau \oplus \cdots \oplus \omega^j(\tau-\Delta+1)r_{ij}^{\tau-\Delta+1}
\end{aligned} \tag{2}
$$

where the temporal weight is constructed using the basic unit-interval monotonic (*BUM*) function

$$
\omega^j(t) = \frac{e^{\frac{t-\tau+\Delta}{\Delta}}(1-e^{-\frac{1}{\Delta}})}{e-1} \tag{3}
$$

A *BUM* function is defined as the function $f : [0,1] \to [0,1]$ where $f(0) = 0, f(1) = 1, f(x) \geqslant f(y)$ if $x > y$ [25].

To obtain an ordered ranked list of the provided alternatives, the linguistic weighted average operator (*LWA*) is used to compute the final score of each one:

$$\begin{aligned} SCORE(\mathscr{A}_i) &= LWA(a_{i1},\ldots,a_{ij},\ldots a_{iJ}) \\ &= W_1 a_{i1} \oplus \cdots \oplus W_J a_{iJ} \end{aligned} \quad (4)$$

The greater the score $SCORE(\mathscr{A}_i)$ is, the better the alternative $\mathscr{A}_i$ will be. A ranking order of the alternatives $\widetilde{\mathscr{A}_1} \succ \widetilde{\mathscr{A}_2} \succ .... \succ \widetilde{\mathscr{A}_n}$ is then generated. Based on both the score and on the output of *decoder* component, the SBSs' owners will decide whether to change or maintain the alternative. If the satisfaction of the current configuration in use is lower than a threshold ρ, then the best alternative $\widetilde{\mathscr{A}_1}$ of this ranked list is suggested by the *decoder* component to the SBS's owner as a required reconfiguration.

4. Results and Evaluation

To illustrate how our proposed solution can be used in a real application, we have designed an Internet-of-things application that is called *Golden Age*. This service-based system monitors different aspects of their patients at home (e.g. heart rate, current location, to name a few), notifying their relatives when appropriate. To notify, *Golden Age* needs a web service that is capable of sending messages (SMS), with at least *good* performance and *good* availability. The *availability* attribute is expressed as a percentage of uptime in a given period of time. At the beginning, the *performance* is subdivided into both the *response_time* and *throughput*. For this study, we have assumed that the *SMS* functional requirement needed by *Golden age* is implemented by the Web service *TextAnywhere SMS*.

In this study, we have considered a subset of the services that are listed in the QWS dataset.* The providers of the dataset have collected 5,000 web services while offering various measurements and they provide a subset of 365 real web service implementations. The majority of the web services offered were obtained from public sources on the Web. The dataset specifically consists of 365 Web services. Each web service presents a set of (9) nine Quality of Web Service (QWS) attributes that have been measured using commercial benchmark tools.

This dataset is partially used to feed the *ED*, replacing in this experiment the *collector* component. A total of 33 alternatives out of 2567 possible services were identified as being able to provide the required $\{SMS\}$ functionality. For instance, we have the following alternatives set $\mathscr{A} = \{TextAnywhereSMS, ..., SmsGatewayService\}$, with the following QoS measurements: (1) *response time*: is the time taken to send a request and receive a response (in milliseconds); (2) *availability*: is the number of successful invocations/total invocations (percentage); and (3) *throughput*: is the total number of invocations for a given period of time (percentage).

Assuming that the SBS's owner is concerned with $G = \{G^{(1)} = response\_time, G^{(2)} = availability, G^{(3)} = throughput\}$ to assess the *performance* and *availability* quality concerns. The linguistic variables under consideration are: $\{LV^{response\_time\ SMS}, LV^{availability\ SMS}, LV^{throughput\ SMS}\}$. *Golden Age*'s owners have agreed to use five linguistic values for each linguistic variable. For *availability* and *throughput*, the linguistic values are $\{poor, fair, good, very\_good, excellent\}$. Meanwhile, for *response_time* the linguistic values are $\{very\_fast, fast, medium, slow, very\_slow\}$.

Moreover, SBS's owners have decided that attributes are equally important ($W^{response\_time} = \frac{1}{3}$, $W^{availability} = \frac{1}{3}$, $W^{throughput} = \frac{1}{3}$) and they have agreed that the selected alternative should satisfy the following assertion: "The selected alternative should have at least a *fast response_time*, at least a *very_good availability* and at least *very_good throughput*."

The *LDM* is constructed as follows. First, we identify the minimum level of quality required for each attribute.

- $LV^{response\_time\ SMS}$: $\{very\_fast;\ \underline{fast};\ medium;\ slow;\ very\_slow\}$,
- $LV^{availability\ SMS}$: $\{poor;\ fair;\ good;\ \underline{very\_good};\ excellent\}$ and,
- $LV^{throughput\ SMS}$: $\{poor;\ fair;\ good;\ \underline{very\_good};\ excellent\}$.

* The data set can be obtained from `http://www.uoguelph.ca/~qmahmoud/qws/` and was released in 2010

Later, the requirements using natural language are reified into a LDM using the Backus-Naur form, as follows:

$$\begin{aligned} LDM &:= LWA(\mathscr{L}(LV_{fast}^{response\_time\ SMS}), \\ &\quad LWA(\mathscr{L}(LV_{very\_good}^{availability\ SMS}), \\ &\qquad \mathscr{L}(LV_{very\_good}^{throughput\ SMS}))) \\ &= W^{response\_time} \cdot \mathscr{L}(LV_{fast}^{response\_time\ SMS}) \\ &\quad \oplus W^{availability} \cdot \mathscr{L}(LV_{very\_good}^{availability\ SMS}) \\ &\quad \oplus W^{throughput} \cdot \mathscr{L}(LV_{very\_good}^{throughput\ SMS}) \end{aligned}$$

The satisfaction degree of the alternative $\mathscr{A}_i$ to the LDM model is computed as

$$\begin{aligned} SCORE(\mathscr{A}_i) &= \tfrac{1}{3}\delta_1(x_{i1}(\tau-\delta+1),\ldots,x_{i1}(\tau)) \\ &\quad \oplus \tfrac{1}{3}\delta_2(x_{i2}(\tau-\delta+1),\ldots,x_{i2}(\tau)) \\ &\quad \oplus \tfrac{1}{3}\delta_3(x_{i3}(\tau-\delta+1),\ldots,x_{i3}(\tau)) \end{aligned} \quad (5)$$

where $\delta_1 = \mathscr{L}(LV_{fast}^{response\_time\ SMS})$, $\delta_2 = \mathscr{L}(LV_{very\_good}^{availability\ SMS})$ and $\delta_3 = \mathscr{L}(LV_{very\_good}^{throughput\ SMS})$. $x_{ij}(t)$ corresponds to the evaluated metric of the attribute j ($j \in \{response\_time, availability, throughput\}$) of the alternative $\mathscr{A}_i$ at time t.

The *SMS* functional requirement of *Golden age* is implemented by the Web service *TextAnywhere SMS*; therefore, $C = \{TextAnywhereSMS\}$.

Simulation experiments

This section describes how the synthetic data has been generated, starting from the measurements of *SMS* services registered in the QWS dataset. The QWS dataset will be considered as made at design time (period $t = 1$) data points. We have considered five periods of time. For the periods ranging from $t = 2$ until $t = 5$, we have simulated an autoregressive process ($X_t = X_{t-1} + \varepsilon_t$) for the *response_time*, *availability* and *throughput* measurements, where we have incorporated additive Gaussian noise ($\varepsilon_t \sim \mathscr{N}(0, \sigma_\varepsilon^2)$). Figures 2, 3 and 4 shows the line chart of the logarithm of response time (in ms), availability (in percentage) and throughput (in invokes per second), respectively. From figure 2 to 4 we have arbitrarily highlighted in colors some services so that we can better appreciate and track their dynamical behavior. In the figures it can be seen that the web services present a high variability in their quality of service when measured at design time ($t = 1$). In addition, and due to the synthetic noise we have introduced, the performance of web services changes over time. Consequently, the QoS can improve or worsen with respect to the functional-equivalent. However, in this case it is only a random behavior that does not necessarily correspond to a trend but rather corresponds to noise. Therefore, we expect that our proposal does not over-react by generating unnecessary reconfigurations.

In this proposal, the linguistic terms are obtained with the quantile information of all 33 alternatives of *SMS* services of the entire database. Figure 5, 6 and 7 show the EDs, at different periods of time, of the Response Time, availability and throughput variables, respectively. Although the resulting time series are random fluctuations with very high variability, the linguistic terms of the ED are slightly disturbed due to the aggregation factor. In the figures, it can be seen that the linguistic terms did not suffer major changes through time. This corresponds to an expected behavior because the web services were only affected with random noise and should not present a change in the concept regarding to new trends.

We have compared the MADM model with the following three approaches:

1. The $MADM_{design\_time}$ computes the score with the ED obtained at design time;
2. The $MADM_{current\_time}$ computes the score with the ED obtained at current time; and,
3. The $MP-MADM$ correspond to our proposal and it computes the score with the historical ED stored since design time.

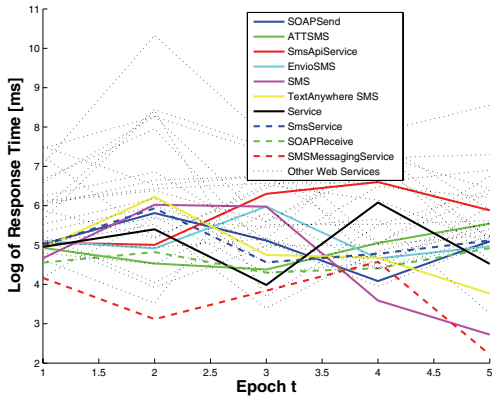

Fig. 2. QoS measurements of the logarithm of the *response time* of the web services obtained with the collector component

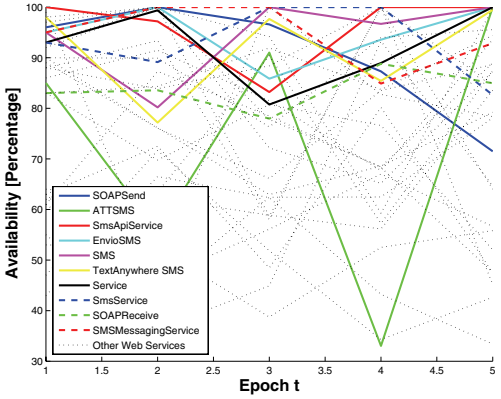

Fig. 3. Sos measurements of the availability of the web services obtained with the collector component

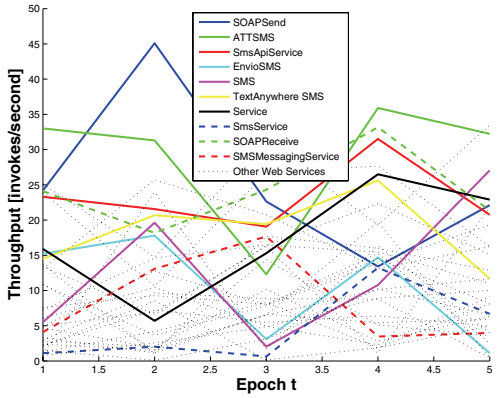

Fig. 4. Sos measurements of the throughput of the web services obtained with the collector component

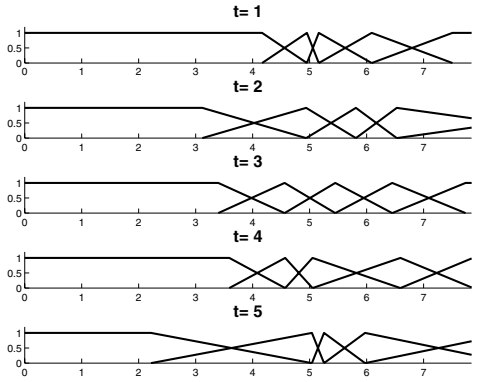

Fig. 5. ED of the logarithm of the *Response Time* for the five periods of time

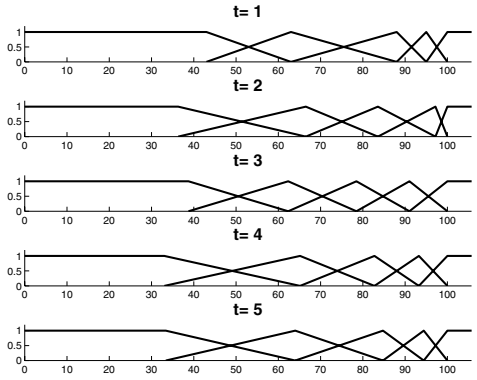

Fig. 6. ED of the *Availability* for the five periods of time

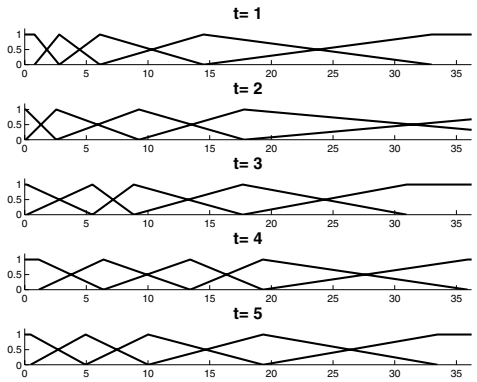

Fig. 7. ED of the *Throughput* for the five periods of time

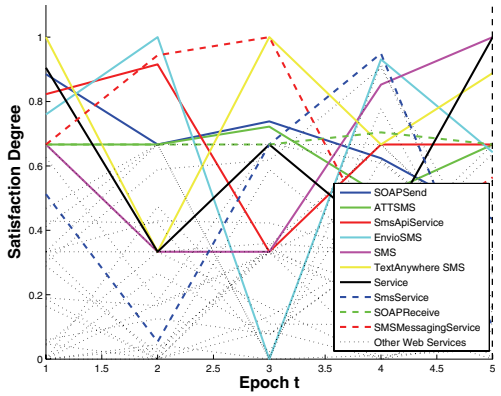

Fig. 8. The $MADM_{design\_time}$ computes the score with the ED obtained at design time

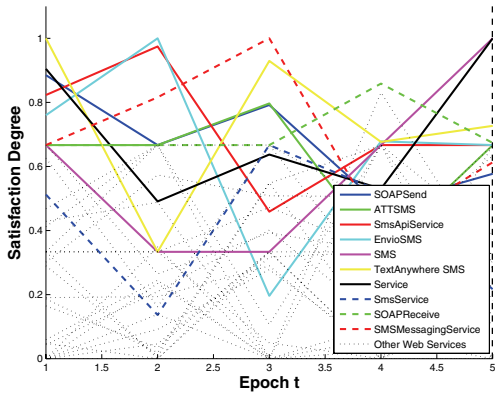

Fig. 9. The $MADM_{current\_time}$ computes the score with the ED obtained at current time

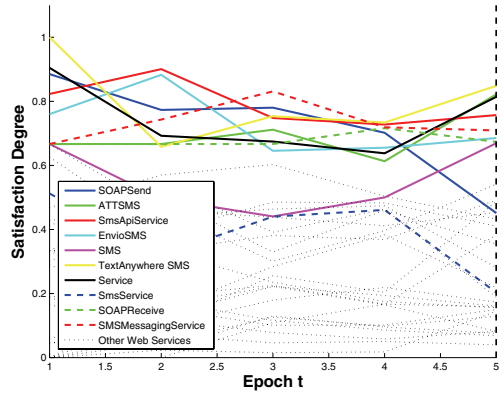

Fig. 10. The $MP-MADM$ correspond to our proposal and it computes the score with the historical ED stored since design time

Figures 8, 4 and 4 show the satisfaction degree for the $MADM_{design\_time}$, $MADM_{current\_time}$ and $MP-MADM$ models, respectively. From Figure 8 to 4, we have arbitrarily highlighted in colors some services so that we can better appreciate and track the dynamic behavior of the aggregated score. Both the $MADM_{design\_time}$ and $MADM_{current\_time}$ exhibit a highly variable behavior. Meanwhile, the $MP-MADM$ is more stable and, therefore, the decision making process becomes more robust. From the ED and the current QoS measurements, the CWW engine computes the satisfaction degree as the score given in equation (5). On the one hand, we have the $MADM_{design\_time}$ where the ED may become obsolete. On the other hand, the $MADM_{current\_time}$ is prone to increase the reconfiguration decision because it is more susceptible to the variability of the service market. It is desired that a reconfiguration decision should be made only when there is enough evidence that the current architectural configuration is violating its requirements.

To analyze the stability, we compute the $R_{violations}$ index as follows. At design time, we select all those services that expose a satisfaction degree above the threshold ρ; that is, we consider only the services that are likely to be selected as part of the architectural configuration. For each of the following periods, we compute the number of times that some of the selected services drop their satisfaction degree below a threshold ρ. Afterwards, we compute the proportion of requirements' violations; that is, the proportion of services that drop their satisfaction degree below a threshold ρ in certain time interval.

$$R_{violations} = \frac{\#selected\_services_{satisfaction<\rho}}{\#selected\_services \cdot \#periods} \quad (6)$$

We executed the experiment 100 times. Table 1 shows the mean and standard deviation of the $R_{violations}$ index obtained for the three different approaches and evaluated at different level of thresholds ranging from 0.50 to 0.95. The numerical results shows that the *MP-MADM* approach provides a better stability than the other two approaches because it obtained a lower $R_{violations}$ index.

Table 1. Comparative table that shows the average and standard deviation of the $R_{violations}$ index. The experiment was executed 100 times.

ρ	$MADM_{design\_time}$	$MADM_{current\_time}$	MP-MADM
0.50	0.1712 ± 0.0262	0.1806 ± 0.0249	0.1061 ± 0.0240
0.55	0.1748 ± 0.0265	0.1872 ± 0.0247	0.1184 ± 0.0284
0.60	0.1809 ± 0.0255	0.1996 ± 0.0239	0.1428 ± 0.0284
0.65	0.1831 ± 0.0269	0.2053 ± 0.0246	0.1600 ± 0.0281
0.70	0.2038 ± 0.0312	0.2099 ± 0.0326	0.1584 ± 0.0361
0.75	0.2045 ± 0.0325	0.2127 ± 0.0320	0.1849 ± 0.0333
0.80	0.2004 ± 0.0338	0.2075 ± 0.0326	0.1895 ± 0.0403
0.85	0.1916 ± 0.0317	0.1989 ± 0.0365	0.1907 ± 0.0421
0.90	0.1825 ± 0.0363	0.1861 ± 0.0381	0.1833 ± 0.0437
0.95	0.1729 ± 0.0437	0.1780 ± 0.0402	0.1775 ± 0.0469

5. Conclusions

In this work, we have proposed a dynamic CWW architecture to support SBSs' owners in their reconfiguration decisions during runtime under changing environments. As in our previous work, we have tackled the obsolescence of the models during runtime by proposing the reification of the requirements into LDM models. We have shown how the inadequacy of the current models to represent non-functional requirements (or in general constraints with qualitative nature) has been addressed. The obsolescence of the design-time models used during runtime have been mitigated transparently and they are naturally underpinned by the LDM models that we provided. Specifically, the CWW engine provided in this paper assesses the satisfaction of the configurations to models and it uses the MP-MDAM data aggregation algorithm to address both the obsolescence of the models and the risk of premature reconfigurations. In a nutshell, the main contribution of this paper is a better management of the trade-off between both the obsolescence of models and the risk of making premature decisions under dynamic environments.

Acknowledgments

We are very grateful to the anonymous referees for their constructive suggestions to improve this paper. The work of R. Torres was partially supported by UNAB Grant DI-1303-16/RG. The work of R. Salas was supported by the grant FONDEF IDeA ID16I10322. H.Astudillo was partially supported by FONDECYT Grant 1140408

References

1. M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.
2. C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Boston, MA, USA, 2nd edition, 2002.
3. D. Bianculli, M. Jazayeri, and M. Pezzè. *Matinée with Carlo Ghezzi: From Programming Languages to Software Engineering*. CreateSpace, 2012.
4. L. Baresi, E. Di Nitto, and C. Ghezzi. Toward open-world software: Issue and challenges. *Computer*, 39(10):36–43, 2006.
5. R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Commun. ACM*, 55(9):69–77, 2012.
6. A. Filieri, C. Ghezzi, and G. Tamburrelli. A formal approach to adaptive software: continuous assurance of non-functional requirements. *Formal Aspects of Computing*, 24:163–186, 2012.
7. G. Blair, N. Bencomo, and R. France. Models@run.time. *Computer*, 42:22–27, 2009.
8. S. Fickas and M.S. Feather. Requirements monitoring in dynamic environments. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, pages 140 – 147, mar 1995.
9. L. Baresi and C. Ghezzi. 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.
10. I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. 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.
11. V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, Francesco Lo Presti, and Raffaela Mirandola. MOSES: A framework for QoS driven runtime adaptation of service-oriented systems. *IEEE Transaction Software Engineering*, 38(5):1138–1159, 2012.
12. R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic QoS management and optimization in service-based systems. *IEEE Transactions Software Engineering*, 37(3):387–409, 2011.
13. R. Torres, N. Bencomo, and H. Astudillo. Mitigating

the obsolescence of quality specifications models in service-based systems. In *Model-driven Requirements Engineering Workshop*, Chicago, USA, 2012.
14. R. Torres. Mitigating the obsolescence of specification models of service-based systems. In *35th International Conference on in Software Engineering (ICSE) 2013*, 2013.
15. L. A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103–111, 1996.
16. F.. Herrera, S. Alonso, F. Chiclana, and E. Herrera-Viedma. Computing with words in decision making: foundations, trends and prospects. *Fuzzy Optimization and Decision Making*, 8(4):337–364, 2009.
17. F. Herrera and E. Herrera-Viedma. Linguistic decision analysis: steps for solving decision problems under linguistic information. *Fuzzy Sets and Systems*, 115(1):67–82, 2000.
18. P. Bonissone. 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.
19. R. Yager. An approach to ordinal decision making. *International Journal of Approximate Reasoning*, 12(3–4):237 – 261, 1995.
20. J. Mendel. The perceptual computer: An architecture for computing with words. In IEEE International Fuzzy Systems Conference, pages 35–38. IEEE, 2001.
21. C. Ghezzi and G. Tamburrelli. Predicting performance properties for open systems with kami. In Raffaela Mirandola, Ian Gorton, and Christine Hofmeister, editors, *Architectures for Adaptive Software Systems*, volume 5581 of *Lecture Notes in Computer Science*, pages 70–85. Springer Berlin Heidelberg, 2009.
22. R. Torres and H. Astudillo. Managing requirements@run.time with a LDM approach. In *Anais do WER14 - Workshop em Engenharia de Requisitos*, Pucón, Chile, 2014.
23. U. Bodenhofer. Ordering of fuzzy sets based on fuzzy orderings. Part I: The basic approach. *Mathware & Soft Computing*, 15(2):201–218, 2008.
24. R. Torres, H. Astudillo, and R. Salas. Self-adaptive fuzzy QoS-driven web service discovery. In *Proceedings of the IEEE International Conference on Services Computing*, SCC '11, pages 64–71. IEEE Computer Society, 2011.
25. R. Yager. Fusion of ordinal information using weighted median aggregation. *Int. J. Approx. Reasoning*, 18(1-2):35–52, 1998.
26. R.E. Bellman and L.A. Zadeh. Decision-Making in a Fuzzy Environment. *Management Science*, 17, 1970.
27. C. Kahraman. Multi-criteria decision making methods and fuzzy sets. In Cengiz Kahraman, editor, *Fuzzy Multi-Criteria Decision Making*, volume 16 of *Springer Optimization and Its Applications*, pages 1–18. Springer US, 2008.
28. M. Delgado, J. Verdegay, and M. Vila. Linguistic decision-making models. *International Journal of Intelligent Systems*, 7:479 –492, 1992.
29. C. Zopounidis and M. Doumpos. *Intelligent Decision Aiding Systems Based on Multiple Criteria for Financial Engineering*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
30. L. Martinez, D. Ruan, and F. Herrera. Computing with words in decision support systems: An overview on models and applications. *International Journal of Computational Intelligence Systems*, 3(4):382–395, 2010.
31. F. Herrera, E. Herrera-Viedma, and J.L. Verdegay. A model of consensus in group decision making under linguistic assessments. *Fuzzy Sets and Systems*, 78(1):73 – 87, 1996.
32. H. Lee. Group decision making using fuzzy sets theory for evaluating the rate of aggregative risk in software development. *Fuzzy Sets and Systems*, 80(3):261 – 271, 1996.
33. R. Yager, L. Goldstein, and E. Mendels. Fuzmar: An approach to aggregating market research data based on fuzzy reasoning. *Fuzzy Sets and Systems*, 68(1):1 – 11, 1994.
34. Ch. Shyi-Ming. A new method for tool steel materials selection under fuzzy environment. *Fuzzy Sets and Systems*, 92(3):265 – 274, 1997.
35. Z. Xu. On multi-period multi-attribute decision making. *Knowledge-Based Systems*, 21(2):164 – 171, 2008.