Application of Trace-Based Subjective Logic to User Preferences Modeling

Hoang Nam Ho, Mourad Rabah, Samuel Nowakowski, Pascal Estraillier

Application of Trace-Based Subjective Logic to User Preferences Modeling

Hoang Nam Ho[1], Mourad Rabah[1], Samuel Nowakowski[2] and Pascal Estraillier[1]
[1] L3i Laboratory, University of La Rochelle, La Rochelle, France.
[2] University of Lorraine, LORIA, UMR 7503, Nancy, France.
{hoang_nam.ho, mourad.rabah, pascal.estraillier}@univ-lr.fr,
samuel.nowakowski@loria.fr

Abstract

A good way to help users make decisions in an interactive application consists in suggesting choices in accordance with their preferences. This decision problem faces challenging tasks, mainly in choosing a good solution that satisfies users and reaches the defined goal. Classical decision methods take into account the goal, but not all the obtained decisions can satisfy users' preferences. The originality of our explorative research is to associate Subjective Logic (SL) to system's traces (historical information) in order to model the user preferences that improve the decision process. Following JØsang, SL provides a suitable framework for modeling and formally describing users' preferences. We propose to connect data collected in past executions, called traces, to the user intuition in order to support subjective reasoning. Based on this result, we can choose a reasonable decision according to users' preferences. A Tamagotchi system will be presented to validate our result.

Keywords: Subjective Logic, opinion, traces, user preferences.

1 Introduction

Our work considers the adaptation in Interactive Adaptive System (IAS). We confine the interactions using contextualized blocs called *situations* [13]. A *situation* in an interactive application is a component where actors interact using resources in a specific context to achieve the defined objectives. The application execution consists in choosing, related to one given situation, the most appropriate following one. After finishing one situation, users have to select another one among a set of available situations to carry on the application execution. In our illustrative case study, a Tamagotchi game[*], the system is contextualized by a set of situations: *feeding*, *playing*, *sleeping*, *healing*, *socializing*, *cleaning* and *educating*. Assuming that the user has completed the situation *feeding*, he has to choose an appropriate situation among all the seven situations above to keep up the system execution. This choice is addressed to the decision support system [10] that will solve the choice problem by synthesizing system data in real time to prioritize all the available situations. We then obtain a suggestion list of ordered situations among which users can choose the next situation to execute. Although the obtained suggestion by the decision method is the result of an optimized computing, users may not be satisfied with the proposed results because they do not take into account users' way of thinking. Indeed, while making a decision, users might want give their opinion for each

[*] The game consists in bringing up a virtual pet (https://en.wikipedia.org/wiki/Tamagotchi).

situation, e.g. “We believe that the situation *playing* will be good for the next execution step comparing to others”. Here above is an example of a users’ *idea* for the situation *playing*. Users may give successively their ideas for all the situations. The synthesis of these ideas is considered as *users preferences*. However, the fact is that the users are not able to express precisely their *preferences* on these situations or at best they have just a slight idea about their preferences if they rely only on the decision algorithms in [10], [4]. This is why we propose to include the *users’ preferences* in the decision making in order to achieve both the intelligent computing and the users’ satisfaction. Furthermore, if we intend to use the users’ preferences, we will face up to a challenge on *users’ preferences representation*. Indeed, users preferences are given in natural language; the computer cannot compute this type of preferences to conclude a priority list of situations. Hence, we must analyze or model users’ preferences. Basically, in our situation-based context, a user preference model can be used to determine how interesting a *situation* is to that user. The preference model of a user can thus be used to select and prioritize situations that may be interesting for that user.

In this paper, we will tackle these problems by answering these two questions:

- How to model users’ preferences in situation-based application executions?
- How to choose a *situation* from the obtained situations priority list to execute in the next step of the application execution?

Besides, if users provide only partial information about their preferences, the system will not have enough data to make the pertinent decisions. In fact, users are fuzzy and imprecise; that leads to the uncertainty concerning the expression and the reasoning about their preferences. The uncertainty is another challenge in decision making. To settle the questions above, we present a method to formally express users’ preferences and simultaneously improve the uncertainty problem by applying two complementary aspects:

- Model users’ preferences with Subjective Logic [8];
- Make use of system traces (users and system past behavior in previous executions) [11].

The remainder of the paper is organized as follows: Section 2 discusses some related work to users’ preferences modeling. Section 3 deals with the basics of the Subjective Logic. The Section 4 presents our Trace-Based System. A framework of Trace-Based Subjective Logic (TBSL) is presented in Section 5. Section 6 presents our experiment design and we conclude the paper with future work in the Section 7.

2 Related work

In this section, we give a brief overview of the traditional methods used to solve users’ preferences modeling. In general, there are three types of implicit methods: collaborative-based, content-based and hybrid approaches. The content-based approach models user’s preferences based on the features presented in rated items obtained by machine learning and data mining techniques [15]. The objective of this approach is to understand the preferences that led the user to judge as relevant or not a given item. The collaborative approach aims to build the preferences model of a user based on the items that were previously used and rated by that user [1]. The advantage of this approach is that it can overcome the problem of unavailability of the items’ content. In fact, these items are not always available or it is difficult to characterize their contents. The hybrid approach combines principles from the two methods above [1]. However, a serious disadvantage of these methods is that they ignore the uncertainty problem about users’ preferences. Usually, users are not certain or precise when they set preferences. To improve this inconvenient, in [15] authors propose a framework that applies fuzzy set theories. But this approach is not suitable for our situation-based context because it requires additional information that a situation bloc does not provide. In our case, users know only the situation name and what it is supposed to do. With only this information, they have to choose the best one for the next

execution step. Thus, we propose a new approach that can model users' preferences both in certain and uncertain cases. The Subjective Logic (SL) emerges as an interesting method to explore. SL [8] is a type of probabilistic logic that explicitly takes into account uncertainty and belief ownership, which are suitable for modeling and analyzing problems involving incomplete knowledge.

The application of the SL for user modeling in decision support and recommendation context has been previously reported. In [14], authors proposed a hypothetical hybrid recommender system which uses *trust* to exploit the latent relationships between users. They use the SL modeling approach to build *trust* from existing evidence. In [7], authors proposed a method to represent reputation scores and recommendation values within the framework of SL. Both of the two works above have illustrated their performances on their particular context. Nevertheless, these two approaches focus on the users immediate interactions and do not consider overall application's execution through its scenario unfolding. Furthermore, due to the context-dependent nature of these approaches, we cannot apply them with our situation-based structuration.

In fact, by applying SL, users can directly provide their preferences. Based on SL effectiveness, these preferences are formally modeled. However, if we allow users to express their preferences according only to their intuitions, it will satisfy the advanced users but it will be an obstacle for beginners. To overcome this drawback, we re-examine the collaborative principle above. We analyze past users preferences to find preferences relevant to the current user. Therefore, we propose to connect SL to a Trace-Based System that allows us to record all events occurring during all the previous executions of different users.

3 Subjective Logic Fundamentals

Subjective Logic (SL) is a type of probabilistic logic that takes into account the uncertainty and the confidence. It provides a standard set of logical operators for using in domains containing uncertainty, and more specifically in domains in which opinions regarding the truth or falsehood of a domain element differ [12]. In general, SL is well suited for modeling and analyzing in studies involving uncertainty and incomplete knowledge. In order to understand how the subjective logic works, we will start by defining some related concepts.

3.1 Definitions

SL operates on a frame of discernment, denoted by Δ, that contains the set of possible states of a given system. The states in the frame of discernment will be called atomic states because they do not contain any substate. The power set of Δ, denoted by 2^{Δ} contains the atomics states and all the possible unions of the atomic states. The one who believes that some states in the power set of Δ might be true can assign *belief mass* to these states. The belief mass of an atomic state $x \in 2^{\Delta}$ is considered as the belief that the state is true. Besides, the belief mass of a non-atomic state is considered as the belief that one of the atomic states it contains is true, but there is uncertainty about which of them is true [6].

Definition 1 (Belief mass assignment)
Given a frame of discernment Δ, we can associate an assignment to each substate $x \in 2^{\Delta}$ a number $m_{\Delta}(x)$ such that:

$$m_{\Delta}(x) \geq 0$$
$$\mathrm{m}_{\Delta}(\emptyset) = 0$$
$$\sum_{\mathrm{x} \in 2^{\Delta}} \mathrm{m}_{\Delta}(\mathrm{x}) = 1$$

m_Δ is the belief mass assignment on Δ and $m_\Delta(x)$ is called the belief mass of x.

Definition 2 (Belief, Disbelief and Uncertainty)
The belief function corresponding to m_Δ is the function $b: 2^\Delta \mapsto [0,1]$ defined by:

$$b(x) = \sum_{y \subseteq x} m_\Delta(y), \qquad x, y \in 2^\Delta$$

The disbelief function corresponding to m_Δ is the function $d: 2^\Delta \mapsto [0,1]$ defined by:

$$d(x) = \sum_{y \cap x = \emptyset} m_\Delta(y), \qquad x, y \in 2^\Delta$$

The uncertainty function corresponding to m_Δ is the function $u: 2^\Delta \mapsto [0,1]$ defined by:

$$u(x) = \sum_{\substack{y \cap x = \emptyset \\ y \notin x}} m_\Delta(y), \qquad x, y \in 2^\Delta$$

There is a theorem that links the three values of belief, disbelief and uncertainty:

$$b(x) + d(x) + u(x) = 1$$

Depending on the three values above, we distinguish several cases:

- If $b = 1$: is equivalent to the TRUE binary logic
- If $d = 1$: is equivalent to the FALSE binary logic
- If $b + d = 1$: is equivalent to the traditional probability
- If $b + d = 0$: is equivalent to the total uncertainty case
- If $b + d < 1$: expresses the uncertainty case that we deal with in this paper.

Definition 3 (Relative Atomicity)
The relative atomicity of x to y is the function $a: 2^\Delta \mapsto [0,1]$ defined by:

$$a(x/y) = \frac{|x \cap y|}{|y|} \qquad x, y \in 2^\Delta$$

The relative atomicity of the state x to Δ is denoted by $a(x/\Delta)$ or $a(x)$.

Definition 4 (Probability Expectation)
The probability expectation function corresponding to m_Δ is the function $E: 2^\Delta \mapsto [0,1]$ defined by:

$$E(x) = \sum_{y} m_\Delta(y) \times a(x/y) \qquad x, y \in 2^\Delta$$

Definition 5 (Subjective Opinion)
Given a frame of discernment Δ that contains a state x and its complement $\bar{x}$, and assuming a belief mass assignment m_Δ with belief, disbelief, uncertainty and relative atomicity functions on x of $b(x)$, $d(x)$, $u(x)$ et $a(x)$, we write ω_x the opinion over x by:

$$\omega_x = \langle b(x), d(x), u(x), a(x) \rangle$$

The opinion's probability expectation is computed by:

$$E_x = b(x) + a(x) \times u(x)$$

Definition 6 (Opinions classification)
Opinions can be ordered according to probability expectation value. We will use the following rules to determine opinions ordering [8]: let ω_x and ω_y be two opinions. They can be ordered according to the following priority rules:

a. The opinion with the greatest probability expectation is the greatest opinion.
b. The opinion with the least uncertainty is the greatest opinion.

c. The opinion with the least relative atomicity is the greatest opinion.

3.2 How to determine a subjective opinion?

The major difficulty with applying Subjective Logic is to find a way to determine and formally model opinions to be used in opinions classification in the list of priorities. There are two methods available: using statistical evidence and using guidelines [12].

Using statistical evidence

In [5], the author describes how opinions can be determined and modeled based on statistical evidence For instance, assuming that a process produces positive and negative outcomes, if the process has produced *r* positive and *s* negative outcomes, the value of *r* and *s* represents the statistical evidence supporting the opinion. An opinion of the positive outcomes can be expressed as $\omega = \langle b, d, u, a \rangle$ where:

$$b = \frac{r}{r+s+2};\ d = \frac{s}{r+s+2};\ u = \frac{2}{r+s+2}$$

$$a = relative\ atomicity$$

Using guidelines

This approach describes a questionnaire for guiding users to express their beliefs as opinions. Users must follow respectively the order of the following questions.

1. Is the idea clearly expressed?
    a. Yes: go to 2.
    b. No: do it and repeat 1.
2. Do you have an intuitive feeling in favor of or against the idea?
    a. Yes: go to 3.
    b. No: You are totally uncertain *b = 0, d = 0, u = 1*: go to 7.
3. How certain is this idea? Give a value *0 <= x <= 1*: go to 4.
4. How is the intuitive feeling against the idea? Give a value *0 <= y <= 1*: go to 5.
5. How is the intuitive feeling in favor of the idea? Give a value *0 <= z <= 1*: go to 6.
6. Normalization of results

    $b = \frac{z}{z+y+(1-x)}$; $d = \frac{y}{z+y+(1-x)}$; $u = \frac{1-x}{z+y+(1-x)}$: go to 7.

7. What is the total number of states in the event space? *n* = total number of states: go to 8.
8. How many states does the idea cover? *m* = number of states covered by the user's idea: go to 9.
9. Compute the relative atomicity of the idea

    $a = \frac{m}{n}$: go to 10.

10. $\omega = \langle b, d, u, a \rangle$

The statistical evidence approach is based on the information in the previous executions to model an opinion from users' ideas. We appreciate this approach in term of statistics, but it contains a drawback about the users' perception. Indeed, the main goal is to model the users' opinions from the users' ideas but the statistical evidence approach does not consider users importance awarding. While the second approach fully addresses the users' ideas, it does not consider the models or external data that allows users to estimate and improve a part of their ideas. From these two approaches, we

conceived a new framework that reuses the advantage of each approach to overcome their disadvantages by combining users survey and previous executions' information, called traces. The following section describes the Trace-Based System that we propose to use.

4 Trace-Based System

A Trace-Based System (TBS) [9], [11] is a system that allows to collect and to analyze traces. We start this section by introducing some TBS related concepts.

- Observer: an *observer* is associated to each relevant event and describes what happens in the application. We must define an observer for every event that occurs during the application execution.
- Trace: a *trace* is a sequence of information generated by any action regarding an object or an event [2].
- Trace model: each *trace* can be associated to a model, called *trace model* that formally represents the corresponding traces. It contains the properties and attributes concerned by the traces such as: time, date, user id, action performed, etc.
- Modeled-trace: called *m-trace* a trace associated it its model. A system that manages m-traces, is called m-Trace-Based System (m-TBS).

Figure 1 gives an example of models (middle part) and the recorded traces (upper part). The lower part shows the executed *situations* chaining.

Despite the diversity of existing TBSs, they still, at least, include the following phases: i) traces collection, ii) traces transformation and iii) traces analysis.

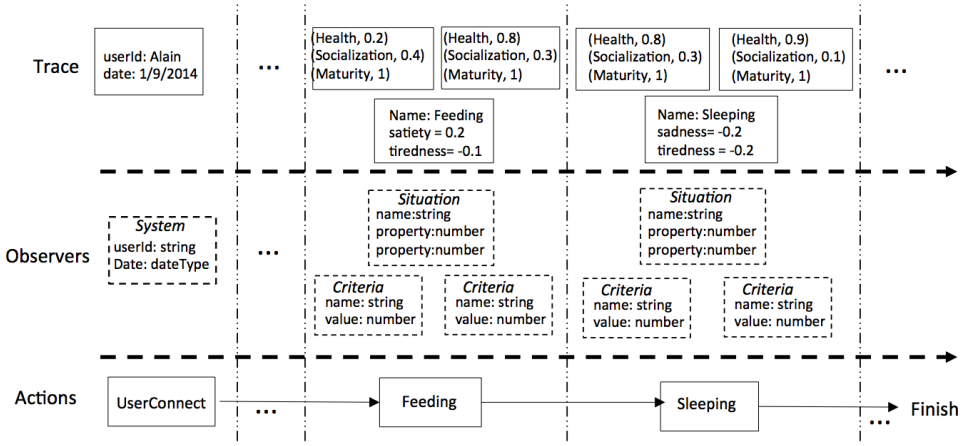

Figure 1: Example of a m-trace in Tamagotchi system

The collection phase identifies the information to collect. During the application execution, users have to choose one situation once the previous one has been completed. In our context, we define three observers associated to three trace models: *system observer* (retrieving system's and users' information), *situations observer* (collecting situations' related information as situations transitions) and *criteria observer* (the logs of criteria fulfillment that is computed by a particular function predefined by the application's designer). Regarding these three trace models, the collected data represent the *primary trace* denoted by $P = (U, S, E, C)$, where

U = *(user_id, date,…)* is user related information, $S = \{s_i\}$ is a set of executed situations, $E \subseteq S \times S$ is a set of situation transitions and C is the set of criteria defined by *(criterion_name, criterion_value)*.

The transformation subsystem transforms primary traces into a new format according to our requirements. In this paper, we want a transformed trace, denoted by $T = (S_{prev}, C_{prev}, S_{next}, C_{next})$, that describes the previous situation with the overall criteria accomplishment and the executed situation with its criteria fulfillment. A transformed trace in the Tamagotchi case study may be: T_i = *<feeding, {(Health, 0.8); (Socialisation, 0.3); (Maturity, 1)}, sleeping, {(Health, 0.9); (Socialization, 0.1); (Maturity, 1)}>*. It indicates that the previous situation was *feeding* and it ended with criteria values (0.8 for Health, 0.3 for Socialization and 1 for Maturity) and it has been followed by the situation *sleeping* that ended with new criteria values (Health increases to 0.9, Socialization decreases to 0.1 and Maturity do not changes).

The analysis step depends on our goal. There is various ways to perform the traces analysis. We want to use SL on transformed traces to improve the decision process

5 Trace-Based Subjective Logic Framework

This section presents the proposed framework that uses the transformed traces to model the users' ideas as subjective opinions. Preferences in our context indicate the suitability of one situation over another one. They are the combination of several subjective opinions obtained from users' ideas. During the application execution, users have to choose a situation among all available ones by giving their preferences on these situations. Users' preferences can fall into the following cases: preferred, not preferred, indifferent or unknown. The variety of users' preferences is not easy to model by a formal object. A framework for preference modeling must contain answers to the following questions:

a. How can users express their preferences?
b. How can we extract past users preferences from an available traces base?
c. How can we represent users preferences and prioritize them for the best choice?

For instance, if a user gives his/her idea about one situation that he/she wants to execute as: "*The situation* playing *is the most appropriate to be executed in the next execution*". From this idea, we suppose that the obtained opinion could be expressed by *ω(0.5, 0.3, 0.2, 0.2)* where the user gives a belief value of *0.5* to this opinion, a disbelief value of *0.3*, an uncertainty value of *0.2* and a relative atomicity value of *0.2*. The user must continue to give his/her idea on different situations and each idea will be modeled by a subjective opinion. In order to determine opinions:

- We can use the guidelines approach to get ideas from users respecting their satisfaction. This is the answer to question (a).
- Concerning question (b), we use data from the TBS built above. From the transformed traces base, we extract the previous users' preferences regarding all criteria values. In fact, not all of these preferences can effectively help the current user because there also exists inappropriate or bad preferences from other users. Thus, we use the criteria values to compute *positive* and *negative outcomes* in order to obtain an evidence base for SL.
- Question (c) is solved by computing the opinion's probability expectation $E(\omega_x) = b + a * u$. Based on this value, we can compare two opinions ω_x and ω_y by respecting the priority rules presented in Definition 6.

If we combine these aspects, we get a Trace-Based Subjective Logic framework for users' preferences modeling. The following procedure describes in detail how to apply TBSL:

1. Do you have an intuitive feeling in favor of or against the opinion?
    a. Yes → 2.
    b. No: you are totally uncertain b = 0, d = 0, u = 1 → 7.
2. What is the previous situation?

Supposing the answer is *S*.
Extract from the transformed traces base all the traces where $S_{prev} = S$.
We obtain a new traces base called extracted traces base → 3.

3. How conclusive is this evidence based on the extracted traces?
   → We define *r* and *s* that are respectively the positive and negative outcomes of the previous execution. For each record in the base (suppose that we have *k* criteria), if $\sum_{i=1}^{k}\left(C_{next} - C_{prev}\right)_i > 0$ then *r = r + 1* otherwise *s = s + 1*. We note *x*, the evidence relevancy that is computed by $x = r/(r+s)$ → 4.
4. How strong is the intuitive feeling against the opinion? Give $y \in [0, 1]$ → 5.
5. How strong is the intuitive feeling in favor of the opinion? Give $z \in [0, 1]$ → 6.
6. Normalization of the results: $\begin{cases} \mathrm{b} = (\mathrm{z}/\mathrm{z} + \mathrm{y} + (1 - \mathrm{x})) \\ \mathrm{d} = (\mathrm{y}/\mathrm{z} + \mathrm{y} + (1 - \mathrm{x})) \\ \mathrm{u} = 1 - \mathrm{x}/(\mathrm{z} + \mathrm{y} + (1 - \mathrm{x})) \end{cases}$ → 7.
7. Compute *a*
   The relative atomicity of the idea to all the available situations:
   a = 1/total number of situations → 8.
8. $\omega = (\mathrm{b}, \mathrm{d}, \mathrm{u}, \mathrm{a})$ is the subjective opinion based on the trace-based evidence and the users' intuition.

The above procedure is our TBSL framework to determine users opinion. After finishing one situation, the user must give his/her idea concerning the overall set of available situations. Each possible situation corresponds to one opinion; for example "*The situation* playing *is the most appropriate one to be executed in the next execution step*". We use the above framework to formally describe this opinion. We proceed consecutively with all the situations and we get a set of subjective opinions. We compute theirs expectation probabilities and then apply priority rules of Definition 6 to order all the possible situations in a priority list.

This is the result of our framework; the users' preferences have been modeled. The chosen situation is the first one in the priority list; it is the suggested result to the user for the next execution step.

6 Illustrative Example and Discussion

We present an example from the Tamagotchi case study [3]. The application consists in looking after and keeping alive a virtual pet called Tamagotchi. The user performs actions to manage this pet whose life is structured by different situations: *feeding (F)*, *cleaning (C)*, *playing (P)*, *healing (H)*, *sleeping (Sl)*, *socializing (So)* and *educating (E)*. We do not describe in detail these situations; their names are explicit enough. Throughout the execution, users will choose successive situations to try to reach an objective combining three criteria: Health, Socialization and Maturity. Each of which is based on a subset of system's properties involved in the overall system's state. After the execution of a given situation, the values of at least one of the three criteria will change. We have developed a prototype depicted in the Figure 2. The Tamagotchi stays in the center of the application and there are seven situations around it. The user has to choose one situation among the seven available ones at each execution step. The user can perform the choice in different ways: he/she can select intuitively one situation to execute, or he/she can consider the result of applying the automated method presented in Section 5 that we implemented in this prototype.

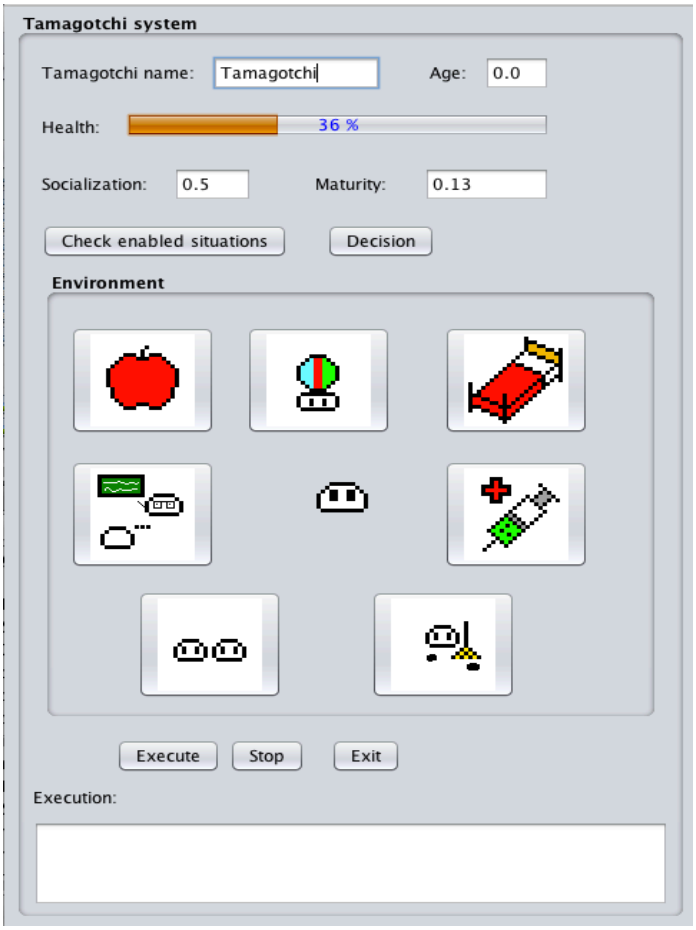

Figure 2: Tamagotchi application

Besides, we include a TBS that is compliant with the specifications of the Section 4. We have conserved all the traces of all the users that have used the Tamagotchi application and we have obtained a base of 1010 traces as the Figure 3.

Relation: Tamagotchi

No.	pre_sit Nominal	pre_health Numeric	pre_socialization Numeric	pre_maturity Numeric	next_sit Nominal	next_health Numeric	next_socialization Numeric	**next_maturity** Numeric
1	feeding	1.4	2.0	0.79	playing	1.6	2.04	1.55
2	playing	1.6	2.04	1.8	sociali...	0.58	1.18	2.84
3	socializing	0.58	1.18	3.19	cleaning	0.54	2.15	5.34
4	cleaning	0.54	2.15	6.05	playing	0.54	2.2	8.23
5	playing	0.54	2.2	8.96	sleeping	0.41	1.98	11.04
6	sleeping	0.41	1.98	11.74	educat...	0.44	1.98	13.75
7	educating	0.44	1.98	14.42	playing	0.37	2.08	16.72
8	playing	1.4	2.0	0.79	sociali...	1.35	1.92	1.57
9	socializing	1.35	1.92	1.83	sleeping	1.18	3.45	3.46
10	feeding	1.4	2.0	0.79	playing	1.64	2.04	1.56
11	playing	1.64	2.04	1.82	playing	1.41	1.82	2.6
12	playing	1.41	1.82	2.86	sociali...	1.18	1.59	3.7

Figure 3: Traces Base of the Tamagotchi application

These traces were used in our TBSL framework to compare the performances of our approach with two others approaches that are collaborative approach [1] (CA) mentioned in the section 2 and the SL framework without traces integration [8] (SL). We have observed the chosen situation for some time units (one time unit corresponds to the execution of the suggested situation to the user). After the execution of the chosen situation, the user evaluates his/her satisfaction about the result: he/she observes the obtained values for the three criteria and marks ✓ if he/she is satisfied or nothing if not.

Time unit	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
TBSL	F	C	F	So	Sl	F	C	P	So	Sl
Satisfaction	✓	✓	✓		✓	✓	✓	✓	✓	✓
SL	F	Sl	P	P	F	So	Sl	F	C	F
Satisfaction	✓	✓	✓				✓			✓
CA	F	C	H	Sl	F	F	P	E	F	F
Satisfaction	✓	✓		✓		✓			✓	✓

Table 1: Comparison of user satisfaction between the 3 approaches: TBSL, SL and CA

Table 1 shows that TBSL gets more user satisfaction than the SL and CA in a general case. In addition, if we consider only the result of TBSL and SL, we notice that the user satisfaction is higher than in the SL without trace analysis. This sample is the one for one new user. We have performed the same test with 10 new users; the obtained result is closely similar to the Table 1. However, the more the number of new users increases, the more the effectiveness of our TBSL is. We can conclude that the traces integration in SL framework can help users to choose efficiently one situation to execute in regards to his/her satisfaction.

	6 beginners			8 experimented users		
	TBSL	SL	CA	TBSL	SL	CA
Satisfaction number	54	46	51	65	77	62

Table 2: Result of user satisfaction between beginners and experimented users

We have also run the test with 14 users having different users profiles (8 experimented users and 6 beginners with the Tamagotchi system) and we also have compared the 10 first time units. The Table 2 summarizes the overall satisfaction. We observe that users satisfaction for the beginners is better with TBSL than with CA and SL. For the experienced users, the satisfaction in TBSL is better than in CA but worse than with SL. The reason is that experienced users are better trained in the application context. Therefore, TBSL is a best-suited approach in determining beginner users preferences to decision-making, especially for non-experienced users first plays.

7 Conclusion

We have explored the association of the Subjective Logic with the Trace-Based System to analyze users' preferences during situation-based application executions, called Trace-Based Subjective Logic. Our TBSL approach can improve users' intuitions by using traces of past executions especially for non-experienced users. We have tested our method and we have obtained promising results.

This work is still in progress. Our current research focus on more advanced tests and on the integration of our approach in an E-learning system. We choose to apply this framework on our Online Distance Learning platform under development by our team. It integrates many mediation functions for the e-Learning courses cycle. The course in our context is defined by several

pedagogical situations as, for instance, *Presentation*, *Go to the board*, *Individual Work*, etc. We will use our TBSL to help the teachers in managing their class.

References

[1] Adomavicius, G. and Tuzhilin, A. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*. 17, 6 (2005), 734–749.

[2] Cordier, A., Lefevre, M., Champin, P.-A., Georgeon, O.L. and Mille, A. 2013. Trace-Based Reasoning - Modeling Interaction Traces for Reasoning on Experiences. *FLAIRS Conference* (Florida, USA, 2013), 363–368.

[3] Ho, H.N., Rabah, M., Nowakowski, S. and Estraillier, P. 2014. Trace-Based Decision Making in Interactive Application: Case of Tamagotchi systems. *IEEE International Conference on Control, Decision and Information Technologies* (Metz, France, 2014), 123–127.

[4] Ho, H.N., Rabah, M., Nowakowski, S. and Estraillier, P. 2014. Trace-Based Weighting Approach for Multiple Criteria Decision Making. *Journal of Software*. 9, 8 (Aug. 2014), 2180–2187.

[5] Jøsang, A. 2001. A Logic for Uncertain Probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 9, 3 (2001), 279–311.

[6] Jøsang, A. and Bondi, V. 2000. Legal reasoning with subjective logic. *Artificial Intelligence and Law*. 8, 4 (2000), 289–315.

[7] Jøsang, A., Guo, G., Pini, M., Santini, F. and Xu, Y. 2013. Combining Recommender and Reputation Systems to Produce Better Online Advice. *Modeling Decisions for Artificial Intelligence SE - 12*. V. Torra, Y. Narukawa, G. Navarro-Arribas, and D. Megías, eds. Springer Berlin Heidelberg. 126–138.

[8] Jøsang, A., Hayward, R. and Pope, S. 2006. Trust Network Analysis with Subjective Logic. *Proceedings of the 29th Australasian Computer Science Conference - Volume 48* (Darlinghurst, Australia, Australia, 2006), 85–94.

[9] Karray, M.-H., Chebel-Morello, B. and Zerhouni, N. 2013. A trace based system for decision activities in CBM process. *2013 IEEE Conference on Prognostics and Health Management* (2013), 1–6.

[10] Köksalan, M., Wallenius, J. and Zionts, S. 2011. *Multiple Criteria Decision Making: From Early History to the 21st Century*. World Scientific.

[11] Laflaquière, J., Settouti, L.S., Prié, Y. and Mille, A. 2006. Trace-Based Framework for Experience Management and Engineering. *10th International Conference, KES 2006* (Bournemouth, UK, 2006), 1171–1178.

[12] Oren, N., Norman, T.J. and Preece, A. 2007. Subjective logic and arguing with evidence. *Artificial Intelligence*. 171, 10–15 (2007), 838–854.

[13] Pham, P.T., Rabah, M. and Estraillier, P. 2015. A Situation-Based Multi-Agent Architecture for Handling Misunderstandings in Interaction. *International Journal of Applied Mathematics and Computer Science*. 25, 3 (2015), 439–454.

[14] Pitsilis, G. and Knapskog, S.J. 2012. Social Trust as a solution to address sparsity-inherent problems of Recommender systems. *CoRR*. abs/1208.1, (2012).

[15] Zenebe, A., Zhou, L. and Norcio, A.F. 2010. User preferences discovery using fuzzy models. *Fuzzy Sets and Systems*. 161, 23 (2010), 3044–3063.