Anggraini, R. N. E., & Martin, T. P. (2017). Fuzzy Representation for Flexible Requirement Satisfaction. In F. Chao , S. Schockaert , & Q. Zhang (Eds.), *Advances in Computational Intelligence Systems - : Contributions Presented at the 17th UK Workshop on Computational Intelligence* (pp. 28-36). (Advances in Intelligent Systems and Computing; Vol. 650). Springer. https://doi.org/10.1007/978-3-319-66939-7_3

Fuzzy Representation for Flexible Requirement Satisfaction

Ratih N.E. Anggraini [1,2] and T. P. Martin[1]

[1] Intelligent Systems Lab, University of Bristol, BS8 1UB, United Kingdom
[2] Informatics Department, Institut Teknologi Sepuluh Nopember, Indonesia
ra16032@bristol.ac.uk

Abstract. The need for adaptive systems is growing with the increasing number of autonomous entities such as software systems and robots. A key characteristic of adaptive systems is that their environment changes, possibly in ways that were not envisaged at design-time. These changes in requirements, model and context mean the functional behaviour of a system cannot be fully defined in many cases, and consequently formal verification of the system is not possible. In this research, we propose a fuzzy representation to describe the result of requirement verification. We use an adaptive assisted living system as the case study. The RELAX language is used to create a flexible system specification. We model and simulate the system using UPPAAL 4 and use a fuzzy approach to translate the simulation result into fuzzy requirement satisfaction. The result shows the benefit of a more flexible representation by describing the degree of requirement satisfaction rather than a strict yes/no Boolean judgment.

Keywords: fuzzy, requirement satisfaction, adaptive system.

1 Introduction

The trend towards intelligent and autonomous systems (such as assistive robots, autonomous vehicles, network controllers and other intelligent software agents) has led to a need to create software that is increasingly adaptive. An adaptive system is defined as one that can alter its behaviour to suit changes in its environment [1] such as sensor failures, human factors, or network condition. The adaptation subsequently modifies requirements, models, and context making it harder to verify, particularly when changes may have not have been fully anticipated by designers.

RELAX is a requirements engineering language for adaptive systems, able to capture the uncertainty in adaptive system requirements [2] so that the verification process can become easier. RELAX was combined with SysML/KAOS which is a goal oriented requirement engineering to model self-adaptive system in UML modelling language [3]. Unfortunately, RELAX is only a requirement language to describe system specification and is unable to assess requirement satisfaction.

Verification is a way to prove or to disprove requirement satisfaction in a system [4]. The classic way of describing requirement satisfaction is using Boolean values, i.e. a yes or no answer. Using this approach, we only know whether a requirement is

satisfied or not - we are unable to know that a requirement was almost always satisfied, or that it was almost satisfied in all cases.

Thus, in this research we utilize a fuzzy approach to describe the degree to which a requirement is satisfied, when the requirement is specified in RELAX. We used UPPAAL [5] to model and simulate the system. The simulation result is translated into a fuzzy representation showing the degree to which the requirement is satisfied. The application is adapted from the RELAX description [2] and is intended to demonstrate the principles of our approach rather than to be a fully realistic system.

2 Background

2.1 Fuzzy Sets

Classical set theory uses Boolean truth values (0 for false and 1 for true) to represent set membership. On the other hand, as noted by Zadeh [6], many concepts used in natural language are loosely defined and admit elements according to a scale of membership rather than according to an absolute yes/no test. Since requirements are initially expressed in natural language, and concepts can be regarded as labels for sets of entities, it makes sense to use a fuzzy representation for flexible requirements. The key idea of a fuzzy set is that its elements are members of a set to some degree, and that a specific element can belong to a greater (or lesser) degree than another element. This is obvious when the membership is based on a numerical measurement (e.g. *tall* people or *expensive* restaurants) and is equally valid in more complex concepts such as *socially-responsible company,* or *reliable software.* Fuzzy set theory, in its simplest form, expresses set membership as a real number between 0 and 1. By using a fuzzy approach, we can represent requirements in a more flexible way [6].

For example, consider a set of coffee drinks. Instead of assessing the sweetness in a crisp format such as sweet or bitter, a fuzzy approach describes it in a more flexible way such as sweet, rather sweet, rather bitter (or slightly sweet) and bitter (not sweet). This flexible representation is more commonly used in human language.

A fuzzy set can be represented by a membership function which maps elements in the universe U to a membership value between 0 and 1 as shown in equation 1.

$$f:U \rightarrow [0,1] \tag{1}$$

For the coffee drinks example, we may measure sweetness based on how many spoons of sugar are added. Let us say that 3 spoons are sweet and no sugar added is bitter. Then we can represent coffee sweetness by a fuzzy membership as shown in **Table 1**.

Table 1. Representation of coffee sweetness membership function f(x)

Spoon of sugar x	Fuzzy value $f(x)$
3	1
2	0.67
1	0.33
0	0

$U: \{0, 1, 2, 3\}$
$f: U \rightarrow [0,1]$
$f(x)$

The X-μ representation of fuzzy sets [7] focuses on the degree of membership and views a fuzzy set as a collection of objects with a loosely defined boundary. At different membership grades, we have different sets. Thus, if asked to define *sweet coffee*, we would accept coffee containing 3 spoons of sugar as a definite member. By relaxing the definition slightly (i.e. lowering the membership threshold) we would accept coffee with 2 or 3 spoons; by relaxing it even more, we might accept 1, 2, or 3 spoons of sugar. The strength of X-μ lies in its focus on crisp sets, which can be processed by standard methods; these sets vary according to the membership threshold.

2.2 RELAX Requirement Language

A dynamically adaptive system (DAS) can have large uncertainty due to continuous change of the system by the modification of its environment. Subsequently, satisfying the system requirement becomes more challenging and hence, a tolerance of requirement satisfaction is necessary. To facilitate this toleration, the requirement language called RELAX was proposed [2].

RELAX allows requirement specifications to be written in a structured natural language. Requirements are written using the SHALL operator. For non-invariant requirements, the SHALL operator will be followed by a temporal operator to handle its flexibility. The RELAX grammar is shown below (see formulae (2)).

$$\begin{aligned}&\phi ::= true \mid false \mid P \mid SHALL\ \phi \mid \text{MAY } \phi_1 \text{ OR MAY } \phi_2\\&\mid EVENTUALLY\ \phi \mid \phi_1 \text{UNTIL } \phi_2 \mid BEFORE\ e\ \phi \mid\\&\text{AFTER e } \phi \mid \text{IN t } \phi \mid \text{AS CLOSE AS POSSIBLE f } \phi \mid\\&\text{AS } \{\text{EARLY, LATE, MANY, FEW}\} \text{ AS POSSIBLE } \phi \qquad (2)\end{aligned}$$

RELAX semantics are expressed using fuzzy branching temporal logic (FBTL)[8]. Moreover, FBTL describes requirement satisfaction using a standard fuzzy membership scale of real numbers between [0, 1] instead of saying the requirements are satisfied or not. The semantics of RELAX are shown in **Fig. 1**.

3 Fuzzy for Requirement Satisfaction Representation

To show how we can represent fuzzy requirement satisfaction, we use the example given in [2], of an adaptive assisted living system, focusing on R1.1 and R1.2. The RELAX requirements we use are shown in **Table 2**. UPPAAL 4 was used to model and simulate the system [9].

Table 2. AAL requirements used as example

R1.1	The fridge SHALL detect and communicate information with AS MANY food packages AS POSSIBLE
R1.2	The fridge SHALL suggest a dietplan with total calories AS CLOSE AS POSSIBLE TO the daily ideal calories

RELAX expression	Informal	FBTL formalization
SHALL ϕ	ϕ is true in any state	$\mathbf{AG}\phi$
MAY ϕ_1 *OR MAY* ϕ_2	in any state, either ϕ_1 or ϕ_2 is true	$\mathbf{AG}(\phi_1 \vee \phi_2)$
EVENTUALLY ϕ	ϕ will be true in some future state	$\mathbf{A}\ \mathbf{F}\phi$
$\phi_1\ \mathcal{U}\ \phi_2$	ϕ_1 will be true until ϕ_2 becomes true	$\mathbf{A}(\phi_1\ \mathcal{U}\ \phi_2)$
BEFORE e ϕ	ϕ is true in any state occurring prior to event e	$\mathbf{A}\mathcal{X}_{<_{e_d}}\phi$ where e_d is the duration up until the next occurrence of e
AFTER e ϕ	ϕ is true in any state occurring after event e	$\mathbf{A}\mathcal{X}_{>_{e_d}}\phi$
IN t ϕ	ϕ is true in any state in the time interval t	(*AFTER* t_{start} $\phi \wedge$ *BEFORE* t_{end} ϕ) where t_{start}, t_{end} are events denoting the start and end of interval t, respectively
AS EARLY AS POSSIBLE ϕ	ϕ becomes true in some state as close to the current time as possible	$\mathbf{A}\mathcal{X}_{\geq_d}\phi$ where d is a fuzzy duration defined such that its membership function has its maximum at 0 (i.e., $m(0) = 1$) and decreases continuously for values >0
AS LATE AS POSSIBLE ϕ	ϕ becomes true in some state as close to time $t = \infty$ as possible	$\mathbf{A}\mathcal{X}_{\geq_d}\phi$ where d is a fuzzy duration defined such that its membership function has its minimum value at 0 (i.e., $m(0) = 0$) and increases continuously for values >0
AS CLOSE AS POSSIBLE TO f ϕ	ϕ is true at periodic intervals where the period is as close to f as possible	$\mathbf{A}(\mathcal{X}_{=d}\phi \wedge \mathcal{X}_{=2d}\phi \wedge \mathcal{X}_{=3d}\phi \wedge \ldots)$ where d is a fuzzy duration defined such that its membership function has its maximum value at the period defined by f (i.e., $m(d) = m(2d) = \ldots = 1$) and decreases continuously for values less than and greater than d (and $2d$, ...)
AS CLOSE AS POSSIBLE TO q ϕ	There is some function Δ such that $\Delta(\phi)$ is quantifiable and $\Delta(\phi)$ is as close to 0 as possible	$\mathbf{AF}((\Delta(\phi) - q) \in S)$ where S is a fuzzy set whose membership function has value 1 at zero ($m(0) = 1$) and decreases continuously around zero. $\Delta(\phi)$ "counts" the quantifiable that will be compared to q.
AS MANY AS POSSIBLE ϕ	There is some function Δ such that $\Delta(\phi)$ is as close to ∞ as possible	$\mathbf{AF}(\Delta(\phi) \in S)$ where S is a fuzzy set whose membership function has value 0 at zero ($m(0) = 0$) and increases continuously around zero
AS FEW AS POSSIBLE ϕ	There is some function Δ such that $\Delta(\phi)$ is quantifiable and is as close as possible to 0	$\mathbf{AF}(\Delta(\phi) \in S)$ where S is a fuzzy set whose membership function has value 1 at zero ($m(0) = 1$) and decreases continuously around zero

Fig. 1. Semantics of RELAX expressions[2]

The model of the food information detection sub system is shown in **Fig. 1**. A sensor in the fridge will detect food information once a day. We used probability 1:9 to model unforeseen situations that make the food information sensor unable to gather the information on all packages. This is represented in the subsystem diagram by the dotted lines labelled 9 and 1. The success of the system depends on how many food packages are not detected. We use a variable *r* in the model to represent the number of undetected food packages, to be relaxed in accordance with the requirement "as many as possible". The system does not meet the requirement if it fails to detect food packages *r* or more times, otherwise it meets this requirement. RELAX-ing R1.1 allows us to incorporate flexibility, which means that this system requirement is considered satisfied to a degree even though a certain number of food packages are not detected. The fuzzy membership function *f(x)* is used to represent the fuzzy requirement (see Equation (3)), where *r* is the (relaxed) threshold value and Δx is the number of undetected food packages. The X-μ interpretation is that at membership 1, the system must detect all 10 food packages to satisfy "as many as possible", at membership 0.75, either 9 or 10 packages must be detected, etc.

$$f(\Delta x) = \begin{cases} if\ \Delta x \leq r, & \frac{(r+1)-\Delta x}{r+1} \\ else, & 0 \end{cases} \qquad (3)$$

In this simple case, the process can also be treated analytically. **Fig. 3** shows the relationship between the degree of relaxation and the probability that the system detects *all* food packages for a range of detection probabilities. For the fully relaxed defini-

tion of *as many as possible* (i.e. at least 7 out of 10 packages), a sensor success rate of 0.9 or 0.98 will almost always detect *"all"* food packages.

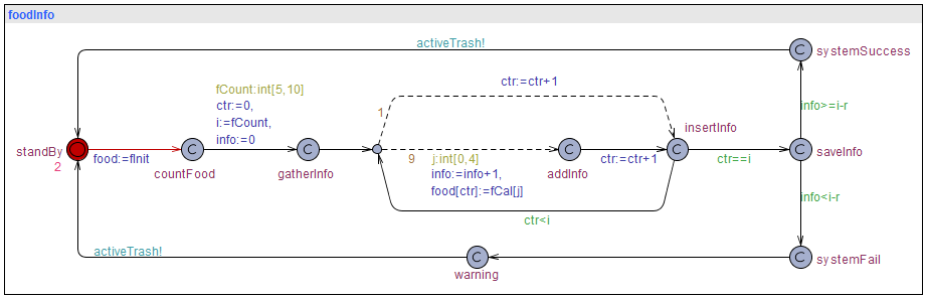

Fig. 2. UPPAAL model to detect food information

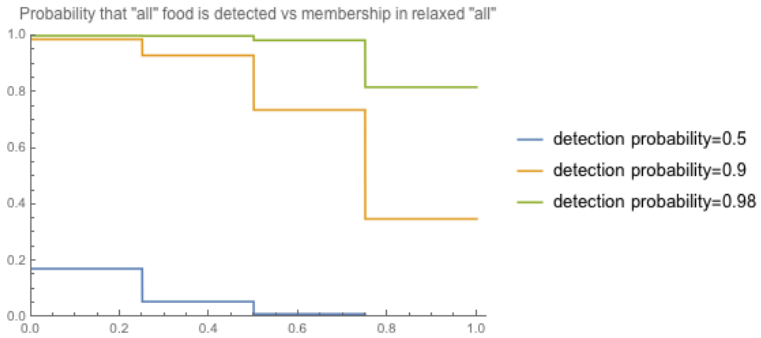

Fig. 3. Probability of detecting *all* food packages vs relaxation of "*all*" definition. The *y*-axis shows the probability, the *x*-axis shows the degree of relaxation in the definition of *all*

In general, we use extended simulation of the system to determine the degree to which the system satisfies the relaxed requirements. **Fig. 4** shows the simulation result of the system model to gather food information. The system performs this task daily. The blue line represents the real food in the fridge and the red line shows the food info that was successfully detected. Moreover, the green one is the absolute minimum food information that should be detected by the system in order for the system to be considered as meeting requirements. In this case, the number of undetected packages is RELAX-ed by 3, so it is considered as a success (to some degree) if the number of undetected food packages is 3 or less. From the chart, we can see that the simulated system works well except at day 32. It means that out of 40 days, the system fails only once in gathering minimum food information. **Fig. 5** shows the (fuzzy) degree to which the requirement is met during 40 days' simulation.

The model of daily calorie intake in **Fig. 6** assumes that Mary[1] has three meals and two snacks time a day where each meal is suggested to be 400 calories and snack 200 calories. The next meal calorie suggestion will be calculated based on how many

[1] Mary is the subject of the assisted living system described in the RELAX paper

calories have been taken up to the current meal. If the system detects that Mary has not taken her meal, it will activate the alarm (3 times). At the end of the day, the system computes total calorie intake and calorie deviation to 1600 calories. The value of 1600 calories is based on calorie calculator [2] which is suggested the total calorie for 65-year-old female, overweight and sedentary activity. The system will be relaxed by $\pm r$ calories and will send a warning to the care-giver or other parties if the diet fails.

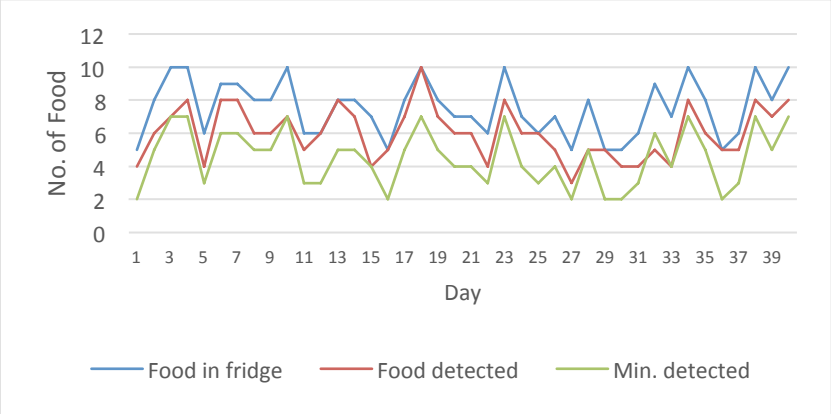

Fig. 4. Simulation result on food information detection

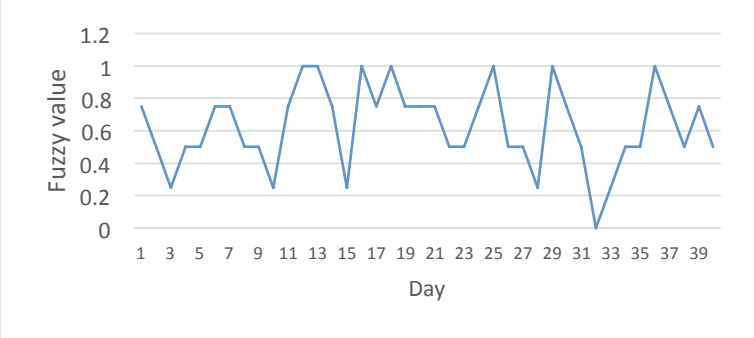

Fig. 5. Fuzzy requirement satisfaction of food detection for 40 days' simulation

The simulation result of calorie intake is shown in **Fig. 7**. The blue line is the daily calorie intake, whilst the green and red lines are the maximum and minimum relaxed calorie intake, respectively. The graph shows rough rises and falls because the decision on taking meals depends on Mary herself. The system can only remind her (activate alarm) and gives warnings if Mary fails to follow the ideal daily intake as shown in the model (see **Fig. 6**).

[2] http://www.healthycalculators.com/calories-intake-requirement.php

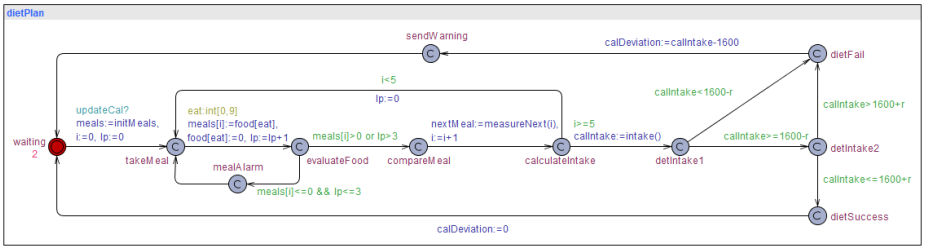

Fig. 6. UPPAAL model to monitor calorie intake

Equation (4) is the membership function *f(x)* used to convert the value of daily calorie intake into fuzzy, where r is relaxing value of calorie deviation and Δx is the actual calorie deviation indicating the difference of ideal daily calorie intake to real consumption. As for the fuzzy requirement satisfaction of calorie intake is described in **Fig. 8**.

$$f(\Delta x) = \begin{cases} -r \le \Delta x \le r, & \frac{(r+1)-|\Delta x|}{(r+1)} \\ else, & 0 \end{cases} \quad (4)$$

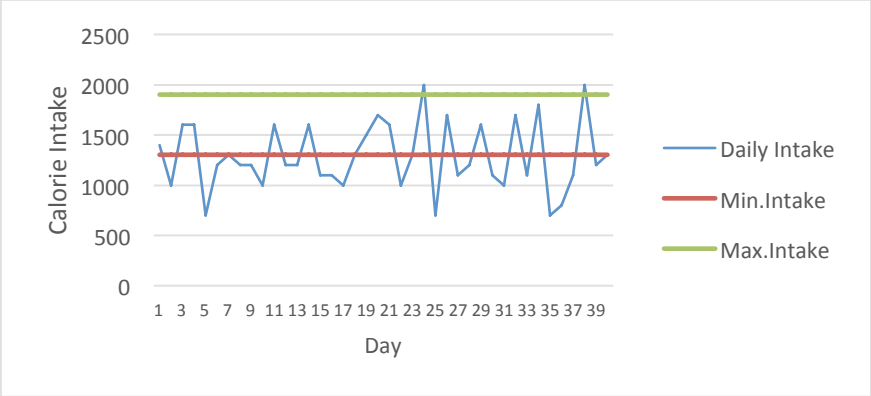

Fig. 7. The simulation result on daily calorie intake

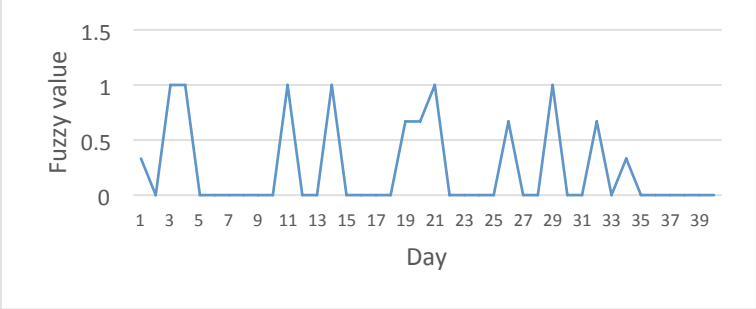

Fig. 8. Fuzzy satisfaction of calorie intake.

Fig. 5 and **Fig. 8** describe the satisfaction of RELAX requirement. Fuzzy value 1 indicates that the requirement is fully satisfied and 0 means it is unsatisfied, and values in between indicate that the requirement is satisfied to some degree.

4 Conclusion

This paper introduces a new way of representing requirement satisfaction using a fuzzy model of flexible requirements. We have modelled a simple system to illustrate the underlying ideas, with the requirement specification written in the RELAX language and UPPAAL 4 used to model and simulate the system. The fuzzy approach has more flexibility than the classic crisp representation so we can describe the degree to which the requirement is satisfied. Simulations are included to illustrate the principles of the approach, and additional analysis will be undertaken to investigate the sensitivity and requirements for reliable simulation.

Future work will examine the theoretical aspects of this approach in greater detail, and develop a fully integrated approach to modelling and refining flexible requirements so that we can verify that a system satisfies the requirements to some degree.

References

1. Tamura, G., Villegas, N., et al., *Towards Practical Runtime Verification and Validation of Self-Adaptive Software Systems*, in *Software Engineering for Self-Adaptive Systems II: Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, R. de Lemos, et al., Editors. 2013, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 108-132.
2. Whittle, J., Sawyer, P., et al., *RELAX: a language to address uncertainty in self-adaptive systems requirement.* Requirements Engineering, 2010. **15**: p. 177-196 (ISSN 1432-010X).
3. Ahmad, M., N. Belloir, and J.-M. Bruel, *Modeling and verification of Functional and Non-Functional Requirements of ambient Self-Adaptive Systems.* Journal of Systems and Software, 2015. **107**: p. 50-70.
4. *Systems and software engineering -- Vocabulary.* ISO/IEC/IEEE 24765:2010(E), 2010: p. 1-418.
5. *UPPAAL website*. [cited 2017; Available from: http://www.uppaal.org/.]
6. Zadeh, L.A., *Fuzzy sets.* Information and Control, 1965. **8**(3): p. 338-353.
7. Martin, T.P, *The x-mu representation of fuzzy sets.* Soft Computing, 2015. **19**(6): p. 1497-1509.
8. Moon, S.-i., K.H. Lee, and D. Lee, *Fuzzy branching temporal logic.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004. **34**(2): p. 1045-1055.
9. David, A., et al., *Uppaal SMC tutorial.* International Journal on Software Tools for Technology Transfer, 2015. **17**(4): p. 397-415.