International Journal on Advanced Science Engineering Information Technology

Vol.9 (2019) No. 6
ISSN: 2088-5334

A Fuzzy Case-Based Reasoning Model for Software Requirements Specifications Quality Assessment

Salama A. Mostafa[#], Saraswathy Shamini Gunasekaran[*], Shihab Hamad Khaleefah[+], Aida Mustapha[#], Mohammed Ahmed Jubair[#], Mustafa Hamid Hassan[#]

[#]*Faculty of Computer Science and Information Technology,Universiti Tun Hussein Onn Malaysia, Johor, 86400, Malaysia*
E-mail: salama@uthm.edu.my, aidam@uthm.edu.my, mohamed.a.jubair@gmail.com, mustafa.hamid.alani@gmail.com

[*]*College of Computing and Informatics, Universiti Tenaga Nasional, Selangor, 43000, Malaysia*
E-mail: sshamini@uniten.edu.my

[+]*Faculty of Computer Science, Al Maarif University College, 31001, Anbar, Iraq*
E-mail: shi90hab@gmail.com

***Abstract*— Different software Quality Assurance (SQA) audit techniques are applied in the literature to determine whether the required standards and procedures within the Software Requirements Specification (SRS) phase are adhered to. The inspection of the Software Requirements Specification (iSRS) system is an analytical assurance tool which is proposed to strengthen the ability to scrutinize how to optimally create high-quality SRSs. The iSRS utilizes a Case-Based Reasoning (CBR) model in carrying out the SRS quality analysis based on the experience of the previously analyzed cases. This paper presents the contribution of integrating fuzzy Logic technique in the CBR steps to form a Fuzzy Case-Based Reasoning (FCBR) model for improving the reasoning and accuracy of the iSRS system. Additionally, for efficient cases retrieval in the CBR, relevant cases selection and nearest cases selection heuristic search algorithms are used in the system. Basically, the input to the relevant cases algorithm is the available cases in the system case base and the output is the relevant cases. The input to the nearest cases algorithm is the relevant cases and the output is the nearest cases. The fuzzy Logic technique works on the selected nearest cases and it utilizes similarity measurement methods to classify the cases into no-match, partial-match and complete-match cases. The features matching results assist the revised step of the CBR to generate a new solution. The implementation of the new FCBR model shows that converting numerical representation to qualitative terms simplifies the matching process and improves the decision-making of the system.**

***Keywords*—software requirements specifications; heuristic search; fuzzy logic; case-based reasoning; classification; similarity measurement.**

I. INTRODUCTION

Software Requirements Specifications (SRS) is an understanding of a customer's system requirements and dependencies in an organized way at a given point in time. [1], [2]. The SRS is an itinerary in software development processes. It is a collection of specified, standardized, and organized requirements (e.g. functional and non-functional requirements) surrounding a software development project and demonstrating future system complete behaviour [3], [4].

The inspection of the Software Requirements Specification (iSRS) system uses a Case-Based Reasoning (CBR) model in carrying out the SRS quality analysis based on the experience of the previous cases. The main reasoning steps of any CBR systems are; *retrieve*, *reuse*, *revise* and *retain* [3], [5]. However, the CBR is exposed to uncertainty when there are limited resources in the case base or due to the continues run. This uncertainty affects both the indexing and retrieving operations and spares the coverage of the problem space by the existing cases [6], [7]. It further affects the determination of solution cases, the modification of the rules used in the case adaptation phase and the revision of the cases to produce new cases [8], [9]. All these issues reduce the performance accuracy of the CBR systems.

The fuzzy logic is a logical disambiguation method that is used to deal with approximate reasoning with the capability of providing correct decisions in uncertain circumstances [10], [11]. It is designed to operate with fuzzy linguistic expressions [12]. The theory of a fuzzy logic-based system could remain fuzzy until it is applied to a particular problem

to remove the fuzziness that the problem might have [13]. Fuzzy logic has been used in a broad spectrum of applications ranging from domestic appliances like washing machines and cameras to more sophisticated ones that include turbine control, tracking and data classifiers [12], [14]. Similarly, the CBR is fundamentally an analogical reasoning technique that can operate with linguistic expression [9], [10]. Additionally, both CBR and fuzzy logic have the capability of easily integrating with other techniques [6], [14], [15]. According to Elaine et al. [16], "fuzzy logic by itself does not exhibit intelligence. Invariably, systems that use fuzzy logic are augmented with techniques that facilitate learning and adaptation to the environment in question". As a result, the environment that the fuzzy logic is situated in has an effective impact on its performance level. Consequently, the combination of CBR and fuzzy logic is capable of providing methods for applying real-world data and at the same time, helpful for acquiring new knowledge [7], [8], [14]. Fuzzy case identification and retrieval mechanism can provide dynamic solutions and improve knowledge creation. Some advantages of applying fuzzy logic in the CBR systems are as follows [9]:

- Allow numerical features to be converted into fuzzy terms,
- Allow multiple indexing of a case on a single feature, which makes it easier to transfer knowledge across domains,
- Allow term modifiers to be used to increase the flexibility in case retrieval.

In summary, the previous studies found that the integration of fuzzy logic with CBR is useful in memory organization, selection or retrieval, matching, similarity measures, adaptation, evaluation and forgetting [7], [8], [10]. It is proven to provide solutions to some of the addressed problems in this work.

Subsequently, this paper proposes the integrating of fuzzy logic technique in the CBR to form a Fuzzy Case-Based Reasoning (FCBR) model. In the iSRS, CBR technique is used to inspect the SRS quality. The main steps of the CBR are *retrieve*, *reuse*, *revise* and *retain* reasoning steps. The fuzzy logic technique is used in this work to improve the performance of the CBR and the inspection accuracy of the iSRS system. Within the FCBR implementation in the iSRS system, some key facts are considered such as case features identification, cases similarity measurement, cases retrieval, and solution adaptation from the most similar cases.

II. Material and Methods

The Inspection of Software Requirements Specification (iSRS) system is based on the SRSQAS system that is proposed by [3]. It is an analytical assurance technique which strengthens the ability to scrutinize the SRS to create a high-quality SRS. Its cycle starts when a user enters the information required to analyse the quality of the SRS which is represented by both SRS Quality Inspection Metrics (QIM) and Quality Inspection Checklist (QIC). It aims to enhance the effectiveness of the SRS success on software development and implementation by inspecting the SRS and evaluating its quality.

The SRS inspection process entails measurements of eleven SRS QIM including complete, consistent, correct and unambiguous. Each of the QIM is defined by a subset of nine interrelated Quality Inspection Indicators (QII) including continuances, directives, imperatives, and options. The QII closely looks and takes in to account structure, syntax and semantics of the SRS. This QIM is originally proposed by Wilson et al. [2] and it is widely adopted by many researchers such as [3], [4] and [5]. Moreover, the iSRS applies an SRS QIC that consists of ten categories of 50 checklist questions. The QIC is collected from different sources including the work of [1], [17], and [18]. Within the QIC, the questions are separated into several main categories that represent different aspects and perspectives of the SRS inspection process. Examples of these categories include "the alignment level to the business objectives", "the compliance level to the standards", "the coverage level to the needs of the stakeholders" and "the depth level to the details of the specification". Different types of techniques like the defect-based and scenario-based are applied in constructing the questions of the QIC.

The iSRS use a CBR model to manage and operates the SRS inspection process. It is because the CBR literature highlights that the best use of CBR is as advisory and consultancy system that rely on users' inputs in its performance and solutions estimation and adaptation. In order to learn the adaptation knowledge, there are various methods such as other domain knowledge, expert user, and the case base under consideration.

The CBR cycle is started with the matching of the new case with the cases that are stored in the case base. Heuristic search is performed to find the *relevant cases*. Here, the system classifies search result cases into three situations: (1) the new case has a complete match in the case base, (2) there are some cases that partially match the new case and (3) there is no case match with the new case as shown in Fig. 1.

Diagnosing the available cases to find a complete or a part of the desired solution in the case base requires a searching process. In the first situation, if there is a complete-match, the solution that the retrieved case has is to be used by the system. This solution leads to confirm the satisfaction of the new case elements and highlights the parts that need improvements. All of this is in the building of an arbitrator SRS quality analysis report.

In the second situation, if there is a partial-match then the cases that partially match the new case will be revised to fit the new case situation. Here, the system implements two levels of matching; cases matching level and features matching level. The matching process is done by fuzzy logic in order to further explore the data of the cases and to find the pattern that they have. The outcome of the revision process is considered as a solution and it is used as a minimum successful solution. Furthermore, in the system learning the part, the new solution is retained in the case base as a possible solution or part of the solution(s) to the new coming problems. In the *retain* step, the system makes the matching process between the solution case and the other cases in the case base to prevent case overlap. In addition, the *retain* step contains unlearning operation applied to system cases to eliminate less needed cases and to prevent overflow status in the case base.

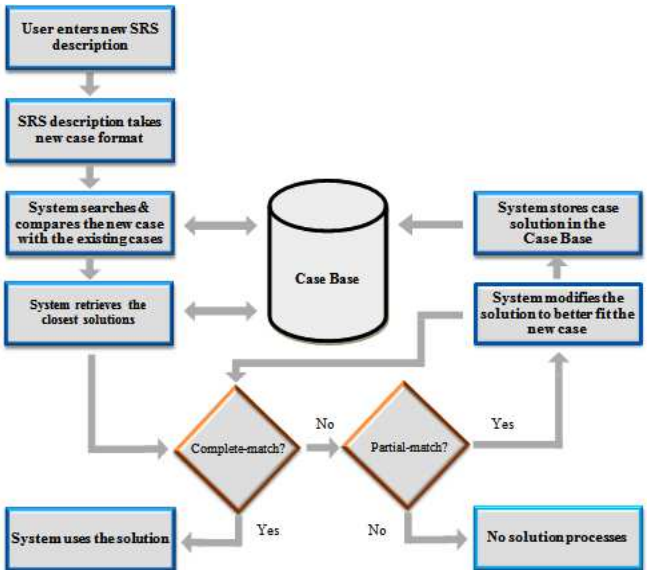

Fig. 1 The proposed iSRS model

Lastly, in the third situation, if there is no-match, the system notifies the user that the new case does not match any case that the system has. From the new case information and the cases in the case base, the system implicitly finds a solution. It gives the generated solution as a proposed solution (in report form), which can help the user on what the problems are with the SRS and what areas in the SRS that need to be improved.

A. *Cases Representation*

In the iSRS, the case base is divided into indexed partitions to allocate each case location. After the user has gone through both QIM and QIC analytics, the obtained data is represented in a case form to be manipulated by the CBR. The QIM is represented by 11 features of criteria and QIC is represented by 10 features of checklist categories that describe the quality of a SRS. The QIM 11 features are represented by 61 elements of the QII and the QIC 10 categories are represented by 50 elements of questions. Each element has a value that is ranging from 1 to 5, where 5 is the highest value. As a result, each case in the case base is represented by 21 features and each feature takes several elements and the overall elements in the case are 111 elements. The description of the case used in the system is shown in Fig. 2.

Case 1									
E1,1	E1,2	E1,3	E1,4	E1,5	E1,6	E1,7	E1,8	E2,1	E2,2
E3,1	E3,2	E4,1	E4,2	E4,3	E4,4	E4,5	E5,1	E5,2	E6,1
E6,2	E6,3	E6,4	E6,5	E6,6	E7,1	E7,2	E7,3	E7,4	E7,5
E8,1	E8,2	E8,3	E8,4	E8,5	E8,6	E8,7	E9,1	E9,2	E9,3
E9,4	E9,5	E9,6	E9,7	E9,8	E9,9	E10,1	E10,2	E10,3	E10,4
E10,5	E10,6	E10,7	E11,1	E11,2	E11,3	E11,4	E11,5	E11,6	E11,7
E11,8	E12,1	E12,2	E12,3	E12,4	E13,1	E13,2	E13,3	E13,4	E14,1
E14,2	E14,3	E14,4	E14,5	E14,6	E15,1	E15,2	E15,3	E15,4	E15,5
E15,6	E16,1	E16,2	E16,3	E17,1	E17,2	E17,3	E17,4	E17,5	E18,1
E18,2	E18,3	E18,4	E18,5	E19,1	E19,2	E19,3	E19,4	E19,5	E19,6
E20,1	E20,2	E20,3	E20,4	E20,5	E21,1	E21,2	E21,3	E21,4	E21,5
E21,6									

Fig. 2 The representation of a case

In the case representation process, the features of the case are indexed by $E_{i,j}$, where; E is the element cell, i refers to the feature number and j is the element number of the feature i. During the quality measurements, the elements of each case are weighted with the variable w. However, w is assigned based on each element importance (w value is; $0 \leq w \leq 1$) and it also depends on the number of the elements that correspond to a particular type of features.

B. *Cases Selection and Retrieval*

As mentioned earlier, the CBR has all the potentials and features of a complete intelligent system by integrating it with other techniques. Most of the AI techniques implementation requires search processes to find a solution or parts of a solution from the search space pool. Cases retrieval is a very important step in the CBR cycle. The system works on finding the most fitting case(s) among a number of cases that exist in the case base. In the complex CBR systems, cases are numerous and each case contents is complex and very detailed. The solution that is generated through CBR reasoning steps is extremely dependent on the quality of the retrieved cases. Hence, cases retrieval efficiency has a huge impact on the other steps and the overall system performance.

1) Relevant Cases Selection Algorithm: In a CBR system, cases selectivity depends on the ability of a selection algorithm to identify attributes that enable the system to measure the similarity level of each case. Only those cases that have the potential to give solutions must be checked and promoted by the algorithm. However, cases design and retrieval mechanisms depend on the application type; therefore, there is no specific way that can fit all CBR systems. The selection algorithms might comprise a heuristic search to reduce or limit the search space of solutions in case bases with complicated application domains. heuristic search is a type of search that uses the rule of thumb strategies for an educated guess. It uses domain knowledge in heuristic rules or procedures that are related to a usually speculative formulation to direct the progress of a selection algorithm as a guide in the investigation on the goal. In the cases retrieval of the iSRS system, the Relevant Cases Selection (RCS) algorithm of Fig. 3 and the Nearest Cases Selection (NCS) algorithm of Fig. 4 are proposed.

The RCS algorithm uses mathematical calculations to find the relevant cases from the cases that are retained in the base of the case. The main parameter for the relevant cases membership determination is the case evaluation result which is stored separately from the other case components. Fig. 3 illustrates the flow chart of the RCS algorithm where Index[R] Relevant Cases is a one-dimensional array that contains the relevant cases, and R is the index that represents relevant cases number.

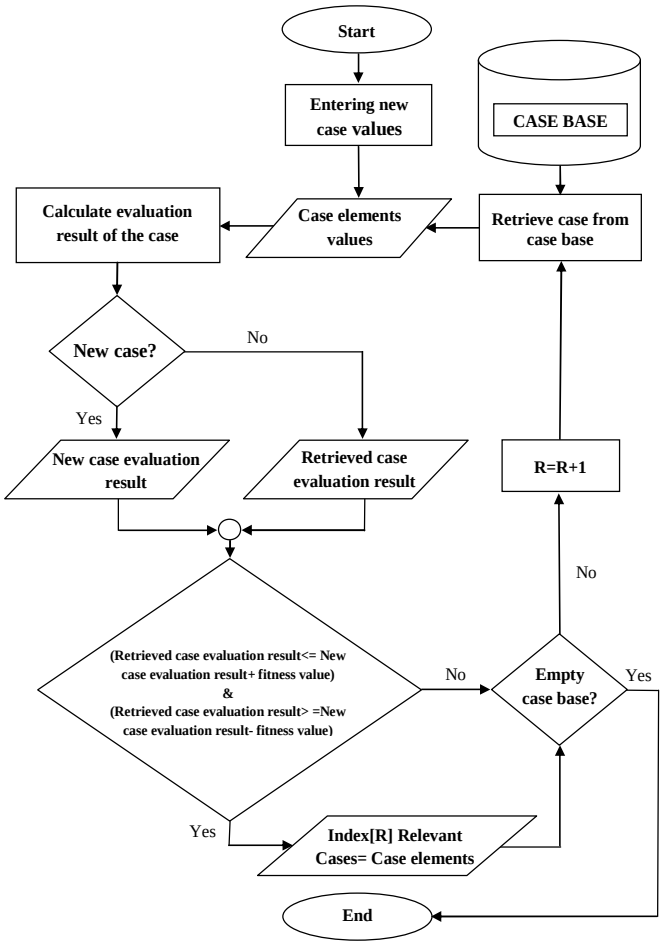

Fig. 3 The RCS algorithm

2) Nearest Cases Selection Algorithm: After all the relevant cases are obtained, the Nearest Cases Selection (NCS) algorithm is used to find the nearest cases to the problem case. Mathematical calculations are also used in this algorithm to find the nearest cases from the gathered relevant cases as shown in Fig. 4. The relevant cases that Index[R] Relevant Cases array contains are being matched with the new case and the matching result is saved in Index[r] Relevant Cases matching result array. After that, the system finds the maximum match (Max) value and the minimum match (Min) value from the Index[r] Relevant Cases matching result array. The mean score of Max and Min is calculated, and then only those cases whose matching result are greater or equal to the mean score, and smaller or equal to the Max are saved in Index [N] Nearest Cases array. These are to be used by the fuzzy logic as nearest cases.

C. Fuzzy Case-based Reasoning

In the FCBR model, fuzzy logic is resident in the CBR cycle (as shown in Fig. 5). Subsequently, by applying fuzzy set logic to the ideal type, system cases can be more understandable and they become as configurations of attributes that appear to a different extent. Thus, the differences in cases kind and degree can also be easily studied and understood.

In the FCBR *retrieve* step, the system starts its cycle by checking the attributes that each case has to measure the similarity level between system cases and the problem case. In the proposed framework, fuzzy logic starts matching the selected *nearest cases* using the obtained attributes in determining how much each case is close to the problem case. The cases are then classified into three groups; no-match, partial-match and complete-match cases.

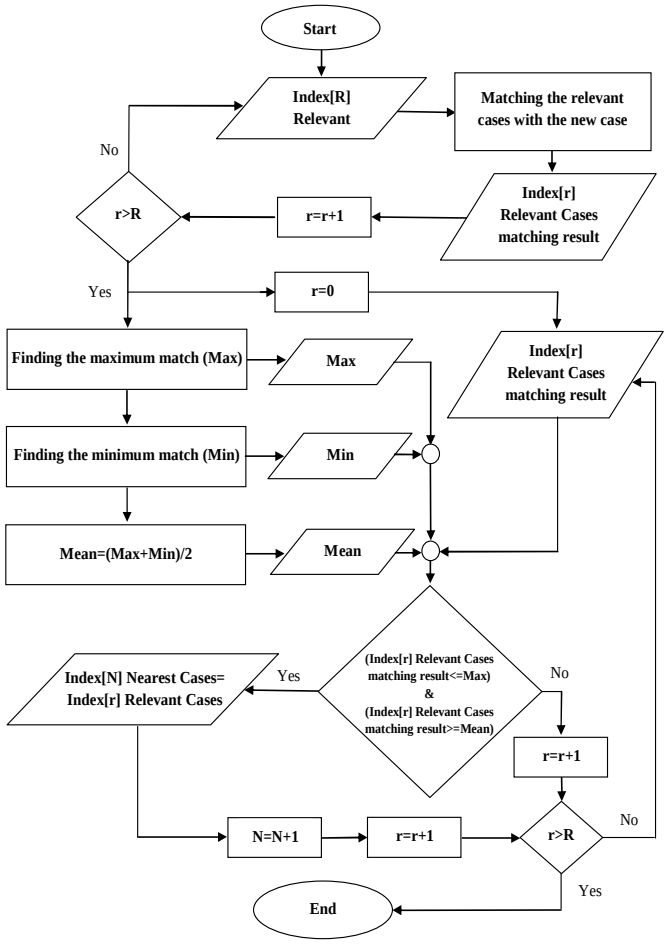

Fig. 4 The NCS algorithm

Adjusting the fuzziness of the matched cases depends on some attributes that are obtained from the CBR case base during heuristic *retrieve* step. However, if there is a partial-match situation with the new case or problem case, the approach does another level of the matching process. Moreover, within the CBR cycle, matching processes to the *retain* step are also required (as shown in Fig. 5). In short, fuzzy logic in iSRS works on the following issues:

- Solving the fuzziness of cases matching in the *retrieve* step (no-match, partial-match and complete-match).
- Solving the fuzziness of case features matching in the *retrieve* step (i.e. no-match, partial-match and perfect-match).
- Enhancing the input parameters of the *revise* step by finding features similarity levels.
- In the *retain* step, matching system cases again (if necessary) to assist the unlearning process by deciding which case needs to be eliminated.

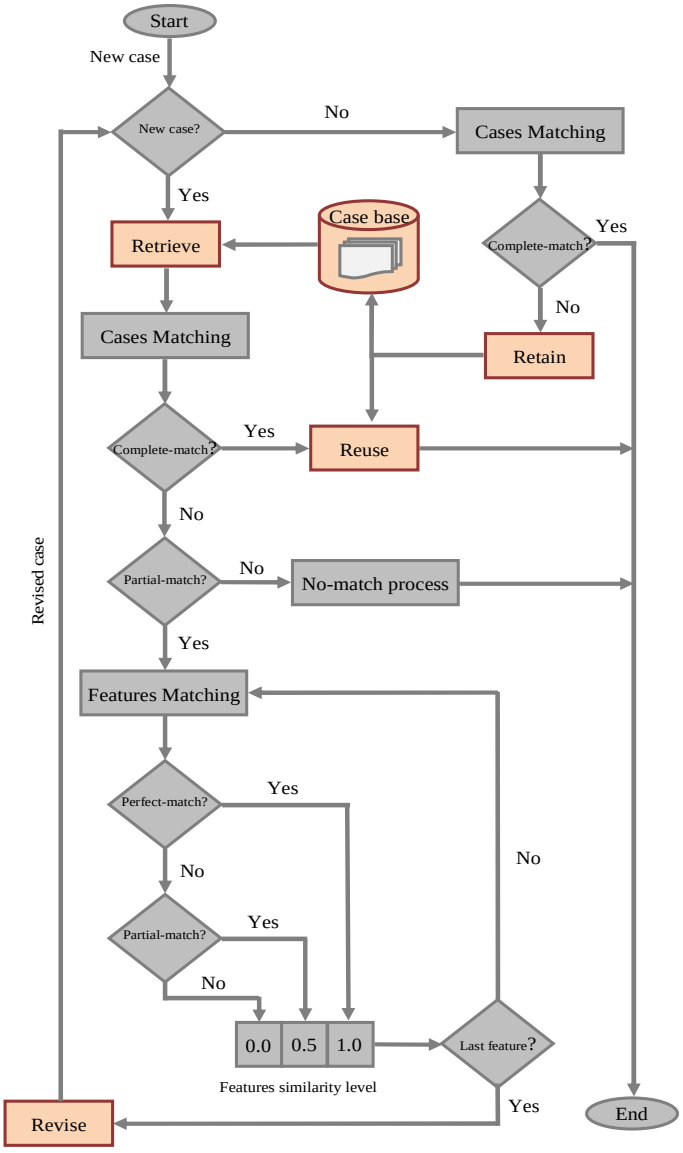

Fig. 5 The FCBR modelling flowchart diagram.

1) *The Fuzzy Cases Matching:* After retrieving the nearest cases, the fuzzy logic approach starts matching the nearest cases. In the *retrieve* step, the cases are classified into three groups; no-match, partial-match and complete-match cases. Table I shows a sample of fuzzy sets of FCBR cases matching result. The approach classifies all the nearest cases into the three matching categories.

TABLE I
FUZZY LOGIC CASES CLASSIFICATION

No-Match	Partial-Match	Perfect-Match
FN_1	FP_1	FC_1
FN_2	FP_2	FC_2
FN_i	FP_j	FC_k

where CN represents no-match cases, $_i$ is no-match cases number, CP represents partial-match cases, $_j$ represents partial-match cases number, CC represents complete-match cases, $_k$ represents complete-match cases number. The addition of $_{i, j, k}$ is equal to the *nearest cases* number.

The fuzzy logic approach determines cases classifications according to dynamic thresholds (i.e., thresholds values are changing according to exploring processes that are held in retrieving the *nearest cases* and mainly depending on their number). The actual range of the no-match state starts from 0% until 10% matching level as shown in Fig. 6. It then falls under a fuzzy area shared with the partial-match state until 30% matching level. The actual partial-match range starts from 30% until 90% and it shares a fuzzy area with the complete-match state that starts from 90% and ends at 99%. Complete-match actual value is 100% case matching and it shares another 9.9% fuzzy area with partial-match case state.

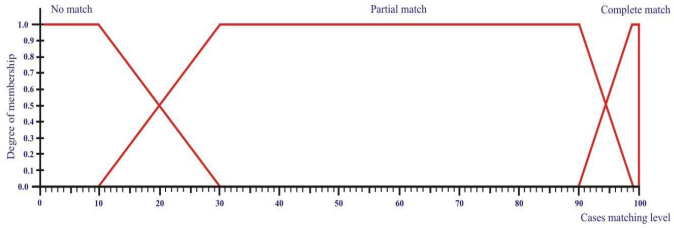

Fig. 6 The fuzzy cases matching

The cases matching classifier is used to find the matching level of each case in order to find the belonging area of each *nearest case*. If there is a complete-match state, the solution is directly used by the system and if the system did not find a complete-match case then, the highest level matching case(s) is to be used to find a solution. Fig. 7 is an example of how the system does the similarity measurement to determine the *nearest cases* matching level, where each green scale areas in the figure show the matched elements in the *nearest cases* as compared with the new case.

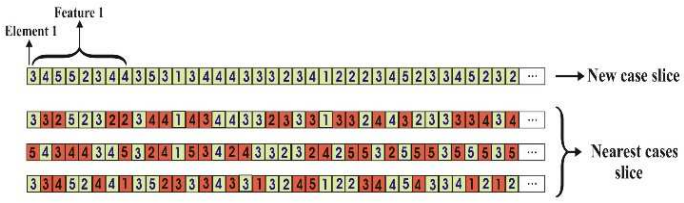

Fig. 7 The fuzzy cases matching result example

Then, in the features matching process, if the system detected partial-match state, each partial-match *nearest case* features are to be matched with the new case features independently to classify the features of each case to; no-match, partial-match and perfect-match features as shown in Fig. 5 and illustrated in IV.B.

2) *Fuzzy Features Matching:* As mentioned earlier, each case contains 21 features and each feature has from 2 to 9 elements (i.e. what the feature represents in the QM and QC methods) which mean each feature elements number depends on the feature itself. fuzzy logic matches cases features by matching the elements of the cases features with problem case features elements. Table II shows a sample of fuzzy sets of CBR case features a matching result.

TABLE II
FUZZY LOGIC CASE FEATURES CLASSIFICATION

No-Match	Partial-Match	Perfect-Match
FN1 FN2 FNi	FP1 FP2 FPj	FC1 FC2 FCk

where FN represents features of the no-match state, $_i$ represents features of no-match number, FP represents features partial-match state, $_j$ represents features partial-match number, FC represents features perfect-match state, $_k$ represents features perfect-match number and the addition of $_{i, j, k}$ is equal 21 which is the case features number.

The formulas that are used by the system in the Fuzzification processes to set cases feature boundaries to find the belonging to each of the no-match, the partial-match and the perfect-match features states are as follows:

Let *f(x)* be the partial-match function. To determine features partial-match state, the following formula is used:

$$f(x) = n * r + x \quad (1)$$

where *n* is the number of elements in the feature (features elements numbers ranged (2-9)), *x* goes from 0 until 1 and *r* is a random number (*r*: 0.299 <= *r* >= 0.499). The *r* values range is selected to fit features matching boundaries.

Let *g(x)* be the no-match function. To determine features no-match state, the following formula is used:

$$g(x) = j + 1 \quad (2)$$

where *j* goes from -1 to [*f(0)* – 1].

Similarly, let *h(x)* be the perfect-match function. To determine features perfect-match state, the following formula is used:

$$h(x) = n \quad (3)$$

where *x* and *n* are the same variables that are used in formulas (1) & (2).

Based on formulas (1), (2) and (3), the possible fuzzy matching boundaries to the case features elements are shown in Fig. 8. Each coloured boundary shows one probability, that one of the case features might fall under. Different features type take a different number of elements. That is why the boundaries of the features matching level need to be shifted according to the feature type or property.

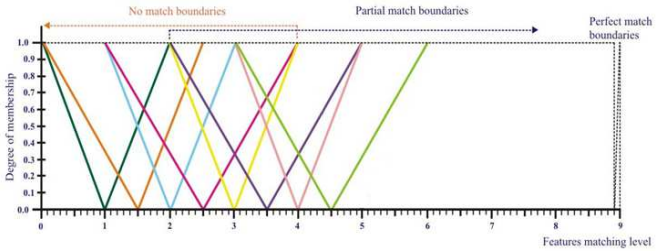

Fig. 8 The eight probabilities of features matching

The main advantage of the feature matching process is to simplify the *revise* step. Fig. 9 shows an example of features matching result where each *nearest case* features are classified into 1.0 which is a perfect-match feature, 0.5 which is a partial-match feature and 0.0 which is a no-match feature.

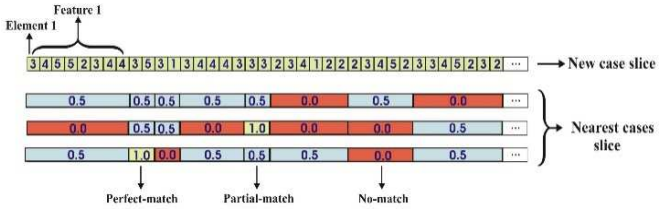

Fig. 9 The feature matching result example

3) Unlearning: The notion of CBR does not only denote a particular reasoning method, regardless of how the cases are acquired, it also denotes a machine learning paradigm that enables sustained learning through case base modification in cases retaining process. In the CBR *retain* step of the iSRS system, the learning process is made by storing the information of the SRS analysis process in the case base after converting it into a case form (as shown in Fig. 2). Using fuzzy logic helps in the unlearning process in the prevention of cases redundancy and avoiding overflow situation to ensure case base efficiency. Similar to the *retrieve* step, fuzzy logic will eventually perform matching at the case level before retaining the new solution in the case base (as explained in II.B).

The unlearning process includes eliminating the cases that are considered not useful by using usability parameters to each case. Usability parameters contain information about each case state (i.e., number of time used, time of usage and similarity level). From these parameters, the system can determine the cases that are less useful and need to be eliminated. As a result, the non-useful case is replaced with the chosen solution (i.e., revised case).

III. RESULTS AND DISCUSSION

A. The RCS and NCS Algorithms

Experimental examples show that the two algorithms select only several cases among 64 cases available in the system case base. The performance analysis of the relevant cases and nearest cases algorithms are shown in Fig. 10. Where the black curve represents the new case indication level, and the coloured curves represent the most similar cases (i.e. the relevant cases and the nearest cases).

Fig. 10 shows how closely the nearest cases are to the new case. This reflexes the importance of applying the RCS and the NCS algorithms to the cases in the *retrieve* step and before the *revise* step is initialized. Both algorithms have succeeded in reducing the cases of the case base to less than 6% with acceptable similarity level. However, fuzzy logic results and the two algorithms' results depend on the new case and case base contents.

The two algorithms are mainly useful in determining the absolute state of the no-match and complete-match cases. They also can suggest the cases that have partial-match (i.e. the *nearest cases*) to the system with the absolute limit. Moreover, to determine the absolute limit of the no-match, partial-match and complete-match to the system cases in which each case has 21 features and 111 elements is a very challenging task. Thus, the system has to adopt other technique like fuzzy logic to handle this issue.

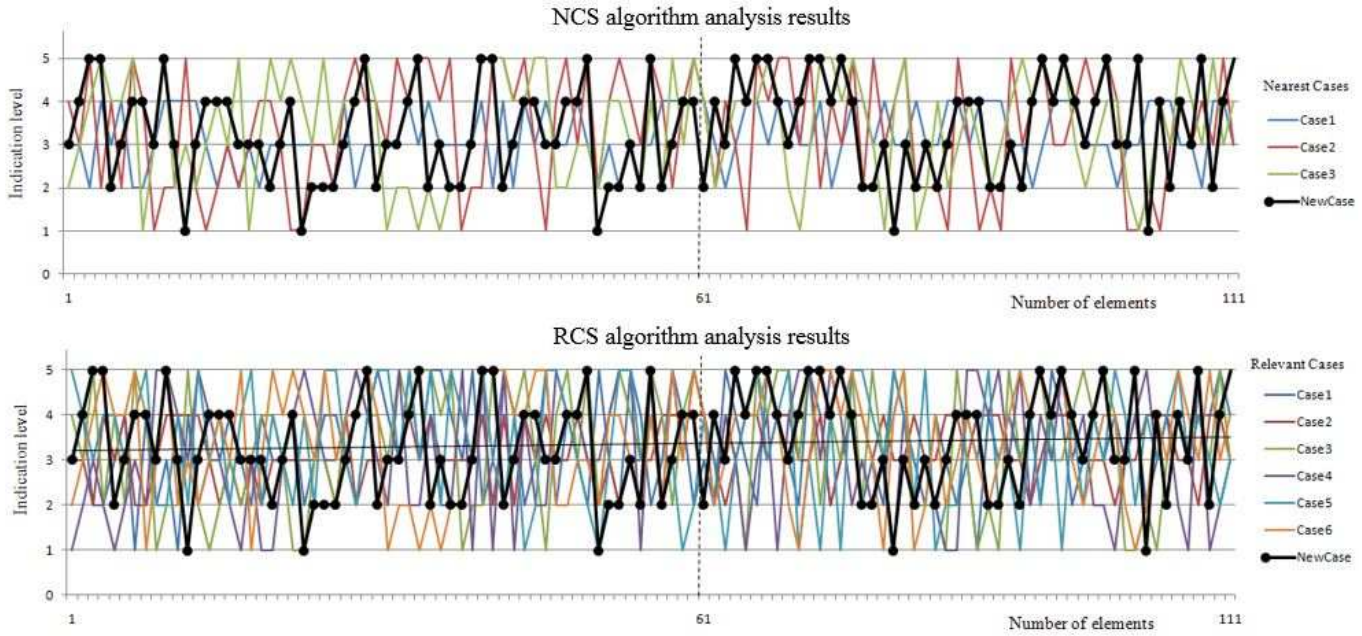

Fig. 10 The RCS algorithm versus NCS algorithms

B. Analysis and Discussion

This is to evaluate the performance of the cases selection, retrieval and similarity measurement handled by the illustrated approaches (the RCS and NCS algorithms and fuzzy logic). The system runs over 64 cases with the possibility of resulting the three matching states (no-match, partial-match and complete-match).

In the first state (blue graph), most of the 64 cases initially do not have a high similarity level with the new case as shown in Fig. 11. Even after the RCS algorithm removal of the irrelevant cases (90.6% of the entire cases), the similarity level did not improve. Subsequently, the NCS algorithm works on the *relevant cases* and is able to eliminate 40% of the *relevant cases* but the cases similarity level remains within the no-match range. As a result, the fuzzy logic decision is no-match state as shown in Fig. 11.

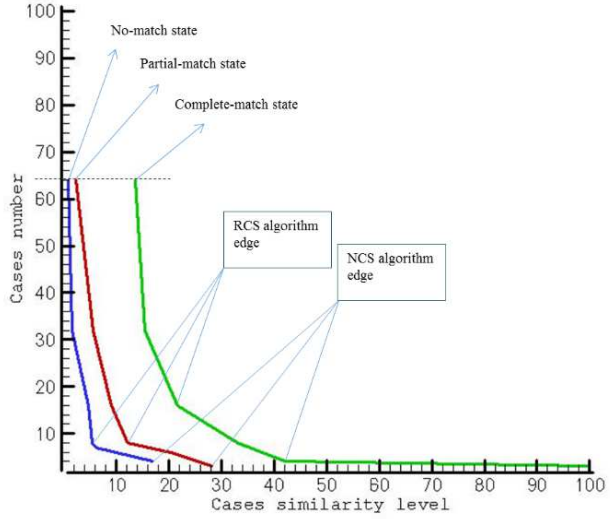

Fig. 11 The similarity measurement of the cases.

Similarly, for the second state (red curve), the two algorithms are able to reduce the 64 cases to 3 *nearest cases* and hence, the fuzzy logic decision resulting in a partial-match scenario (the similarity level is within the partial-match region). Finally, in the third state (green curve), the algorithms perform well by tuning down the 64 cases into 1 case as a result of a high similarity level with the new (problem) case. This similarity level is within the complete-match region as such, the fuzzy logic's decision is a complete-match. These three testing states illustrate the ability of the fuzzy logic to improve the CBR reasoning cycle by providing suitable example cases to be used in generating solutions which leads to improving the overall performance of the system.

IV. Conclusion

Inspection of Software Requirements Specification (iSRS) is a successfully devised analysis tool that can measure the extracted discriminating features of the SRS document and then, guides the user to do the recommended improvements. This system uses Quality Inspection Metrics (QIM) and Quality Inspection Checklist (QIC) as SRS quality measurement methods. In this work, the iSRS is supported by a powerful decision-making mechanism that relies on Fuzzy Case-Based Reasoning (FCBR) model in inspecting the SRSs. The FCBR model has the benefit of using fuzzy logic for cases matching process in case *retrieve* and *retain* steps and it also simplifies the *revise* step by performing cases features matching. With fuzzy logic implementation, the cases that the system has are becoming more usable and useful. Moreover, the model introduces the Relevant Cases Selection (RCS) algorithm and the Nearest Cases Selection (NCS) algorithm. The two algorithms successfully narrow the search space of the cases selectivity options in order to fetch the goal cases by employing certain match criteria. With this in hand, the iSRS analysis results attain higher accuracy. To our knowledge, there exists no methods or standard that can lead us in obtaining QIM and QIC weights,

hence the weights of the FCBR are user-defined. As a part of future prospects, utilizing a genetic algorithm to set up and adjust the QIM and QIC weights according to a database of past experiences is to be proposed.

ACKNOWLEDGMENT

This project is partially sponsored by University Tenaga Nasional (UNITEN) under the UNIIG Grant Scheme No. J510050772. It is also partially supported by the Universiti Tun Hussein Onn Malaysia (UTHM) under RMC Research Fund Vot E15501.

REFERENCES

[1] D. Galin, *Software Quality Assurance: From Theory to Implementation*: Pearson Education Limited, 2004.

[2] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated analysis of requirement specifications," In Proc. of the 19th international conference on Software engineering, ACM, 1997, pp. 161-171.

[3] H. Mat Jani, and S. A. Mostafa, "Implementing Case-Based Reasoning Technique to Software Requirements Specifications Quality Analysis", The International Journal of Advancements in Computing Technology, (IJACT), Vol. 3, No. 1, 2011, pp. 23-31.

[4] A. A. Alshazly, A. M. Elfatatry and M. S. Abougabal, Detecting defects in software requirements specification. *Alexandria Engineering Journal*, *53*(3), 513-527, 2014.

[5] H. Mat Jani, "Applying Case-Based Reasoning to Software Requirements Specifications Quality Analysis System", in The Proceeding of The 2nd International Conference of Software Engineering and Data Mining (SEDM 2010): IEEE/AICIT, Chengdu, China, pp 140-144, 2010.

[6] M. A. Jubair, S. A. Mostafa, A. Mustapha and H. Hafit, "A Survey of Multi-agent Systems and Case-Based Reasoning Integration," In *2018 International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR)*, IEEE, pp. 1-6, Aug., 2018.

[7] S. Nikolaidis, and C. Lazos, Fuzzy Case Identification in Case-Based Reasoning Systems, Computational Intelligence, Volume 15, Number 3, 2000.

[8] P. P. Bonissone, and L. M. Ramon, F4.3 Fuzzy Case-Based Reasoning Systems: Citeseer, 2008 [online]. Available: http://www.mendeley.com/research/f4-3-fuzzy-casebased-reasoning-systems/.

[9] K. P. Sankar, and C. K. Simon, Foundation of Soft Case-Based Reasoning: John Wiley & Sons, Inc., Hoboken. New Jersey, 2004.

[10] S. A. Mostafa, M. S. Ahmad and M. Firdaus, A soft computing modeling to case-based reasoning implementation, *International Journal of Computer Applications*, *47*(7), 14-21, 2012.

[11] S. A. Mostafa, A. Mustapha, M. A. Mohammed, M. S. Ahmad and M. A. Mahmoud, A fuzzy logic control in adjustable autonomy of a multi-agent system for an automated elderly movement monitoring application. International journal of medical informatics, 112, 173-184, 2018.

[12] H. T. Nguyen, C. L. Walker and E. A. Walker, *A first course in fuzzy logic*. CRC Press, 2018.

[13] S. A. Mostafa, R. Darman, S. H. Khaleefah, A. Mustapha, N. Abdullah and H. Hafit, A general framework for formulating adjustable autonomy of multi-agent systems by fuzzy logic. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, Springer, Cham, pp. 23-33, Jun. 2018.

[14] M. K. A. Ghani, M. A. Mohammed, M. S. Ibrahim, S. A. Mostafa and D. A. Ibrahim, Implementing an Efficient Expert System for Services Center Management by Fuzzy Logic Controller. *Journal of Theoretical & Applied Information Technology*, vol 95,13, 2017.

[15] M. A. Mohammed, M. K. A. Ghani, N. A. Arunkumar, O. I. Obaid, S. A. Mostafa, M. M. Jaber and D. A. Ibrahim, Genetic case-based reasoning for improved mobile phone faults diagnosis. Computers & Electrical Engineering, 71, 212-222, 2018.

[16] R. Elaine, K. Kevin, and B. Shivshankar, Artificial Intelligence (3nd Edition): McGraw-Hill Education, India, 2009.

[17] Firesmith, D. (2003). Specifying good requirements. *Journal of Object Technology*, *2*(4), 77-87.

[18] J. D. Blaine and J. Huang, Software quality requirements: how to balance competing priorities. IEEE Software, 25(2):22–24, 2008.