






# Integrating Fuzzy Logic Technique in Case-Based Reasoning for Improving the Inspection Quality of Software Requirements Specifications

Salama A. Mostafa<sup>1</sup> , Saraswathy Shamini Gunasekaran<sup>2</sup> ,  
and Shihab Hamad Khaleefah<sup>3</sup> 

<sup>1</sup> Faculty of Computer Science and Information Technology,  
Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Johor, Malaysia  
salama@uthm.edu.my

<sup>2</sup> College of Computing and Informatics, Universiti Tenaga Nasional,  
43000 Kajang, Selangor, Malaysia  
sshamini@uniten.edu.my

<sup>3</sup> Faculty of Computer Science, Al Maarif University College,  
31001 Anbar, Iraq  
shi90hab@gmail.com

**Abstract.** The development success of software is essentially based on the quality of its Software Requirements Specifications (SRS). A requirement represents the main objective that needs to be accomplished, while a specification is a full description of this objective. The inspection of the Software Requirements Specification (iSRS) system is developed to ensure that the SRSs are of high quality. This paper presents the contribution of integrating a fuzzy logic technique in the Case Base Reasoning (CBR) as a reasoning framework in the iSRS system. The fuzzy logic technique provides a disambiguation mechanism within the Retrieve, Reuse, Revise, and Retain steps of the CBR cycle. Specifically, it is used as a similarity measurement technique in the matching process between the inspected SRS cases and the existing SRS cases in the CBR case base. It then classifies and labels the cases in the case base to no-match, partial-match and, complete-match cases. This classification improves the overall reasoning and inspection of the SRS quality by comparing the inspected case with the most similar cases of the case base.

**Keywords:** Software Requirements Specifications · Quality inspection · Similarity measurement technique · Case-based reasoning · Fuzzy logic

## 1 Introduction

Software is a complex configuration surrounded by many parameters that must be taken into consideration by the developers' team. These parameters include functional and non-functional requirements such as customer requirements, hardware requirements, quality requirements, market requirements, technology requirements feasibility

requirements, and others [1]. The Software Requirements Specifications (SRS) is an itinerary of documenting a collection of specified, standardized, and organized information involving in a software project development processes it is meant to demonstrate future software complete view [2].

The Inspection of Software Requirements Specification (iSRS) system is based on the SRSQAS system that is proposed by [3]. 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 Quality Inspection Metrics (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 into 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].

Additionally, the iSRS applies a Quality Inspection Checklist (QIC) that consists of ten categories of 50 checklist questions. The QIC is collected from different sources including the work of [6–8]. 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.

Subsequently, Case-Based Reasoning (CBR) is used in the implementation of the iSRS to handle the SRS inspection process through the QIM and QIC measurements. The inspection entails referring to previously-stored SRS inspection cases (i.e. past experiences) that have been successfully completed [3, 9]. However, there are problems of complexity and uncertainty in the case base content that affects the overall CBR inspection performance, especially, the retrieving process. The complexity results from the difficulty of the SRS inspection and the variation between the SRS cases. The SRSs are not following a standard template or format and they belong to a wide range of applications. These issues affect the matching and reasoning processes of the CBR and the final outcomes. On the other hand, the uncertainty is presented partially in the indexing and retrieving, sparse coverage of the problem space by the existing cases, and in the description of the inspection. Moreover, it is presented in the semantics of abstract features (i.e., QIC and QII) used to index the cases, the evaluation of the similarity measures computed across these features, the determination of relevancy of the similar cases, and the modification of the rules used in the case adaptation phase.

The integration of fuzzy logic with CBR is found useful in memory organization, selection or retrieval, matching, similarity measures, adaptation, evaluation and forgetting [10]. The fuzzy logic has the capability of easily integrating with other

techniques. 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 of the iSRS system to improve the inspection accuracy result of SRSs.

## 2 Related Work

The related work covers software requirements inspection, CBR and fuzzy logic. Apparently, the combination of fuzzy logic and CBR is yet to be used in SRS inspection. The CBR alone has been used in several SRS inspection systems as in [3, 9].

John Yen and Tiao [11] propose a House of Quality (HOQ) framework in the requirements inspection. The main idea behind the HOQ framework is that software products should be designed to reflect clients' desires. Subsequently, this framework provides a platform in which all participants can communicate their thoughts about a product to identify the requirements and their interrelationships. In the HOQ framework, fuzzy logic is used to capture participants' imprecise requirements and revise them in such a way that reduces the ambiguous phrases and facilitates the interactions between the participants.

The work that is made by Karsak [12] propose a Quality Function Deployment (QFD) framework to improve software product. The QFD applies several client satisfaction parameters. It also considered additional development limitations parameters that are related to the software product cost, budget and technical difficulties. It considers several functions of an organization such as design engineering, marketing and manufacturing. The QFD utilizes fuzzy logic with multiple objective approaches in the inspection process. The fuzzy logic determines the fulfilment of the requirements during the design phase by using sets of linguistic variables. These variables represent and evaluate the inspection parameters.

Sen and Baracl [13] propose fuzzy Quality Function Deployment methodology named (QFD) in handling the requirements analysis. QFD is used to overcome problems that occur in the requirements analysis process by obtaining and translating the requirements into a set of detailed design specifications. It focuses mainly on handling the specifications of the non-functional requirements. Fuzzy QFD approach is deployed to determine which of the non-functional requirements reported by earlier studies are important to a company's software.

The work of Dhote [14] presents three categorize of requirements namely business requirements, user requirements and functional requirements that need to be evaluated during software project development phases. This work proposes a framework that utilizes fuzzy logic technique to handle the uncertainty or the ambiguity in the specifications of these requirements. The framework includes linguistic analysis and checklist approaches to determine the ambiguity level of the requirements. Finally, a traceability table is generated to specify the ambiguous requirements and the necessary revision.

Mat Jani and Mostafa [3] propose an SRS quality assurance and audit framework to determine whether the required standards and procedures of the SRS document are being carefully achieved. The framework adopts Wilson et al. [2] SRS evaluation criteria and utilizes a CBR reasoning technique to analyse the quality of the SRSs. The framework is implemented to SRS Quality Analysis system (SRSQAS) as an executable prototype. The users of the SRSQAS are project architects and SRS reviewers that online interaction with the system through a series of Q&S sessions and checklists to check quality and characteristics of the SRSs. The CBR reasoning is performed by referring to stored similar cases in the case base as a past experience.

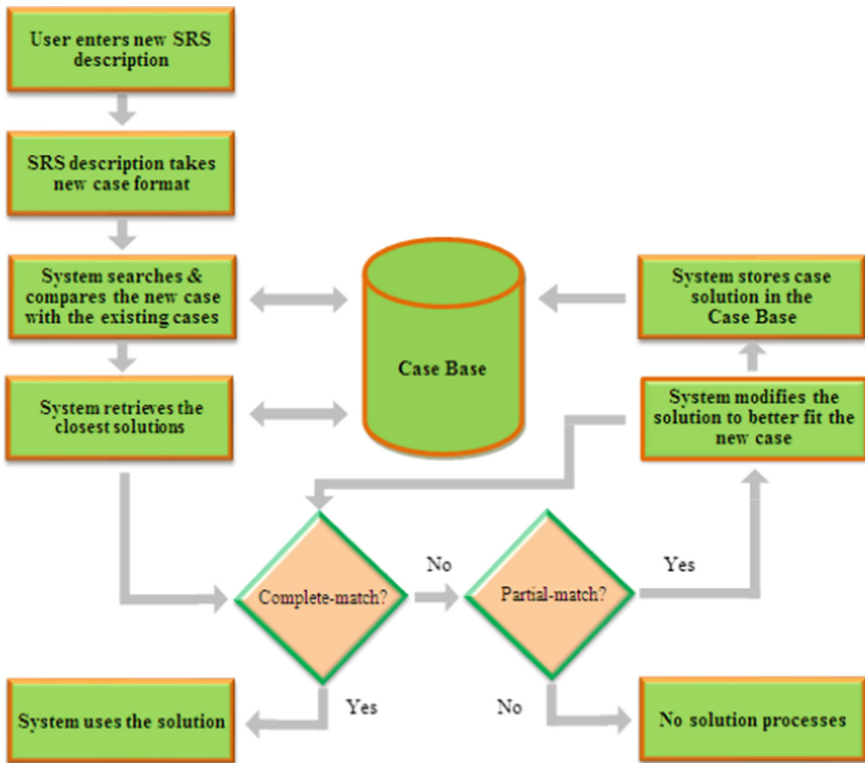
Daramola et al. [9] propose a framework for managing implicit requirements (ImRs) based on semantic reasoning and by using CBR technique. The framework attempts to analyze and discover the ImRs in the SRSs. The ImRs are basic requirements within an organizational context that usually not explicitly expressed in the SRS and neglected by the software development team. They claim that the insufficiently addressed ImRs is one of the causes of project failure. The framework provides automated means for analyzing and discovering the ImRs. The CBR applies analogies to analyze the ImRs then reuse the stored ImRs in the case base as solutions to modify the SRSs.

### 3 The Proposed Framework

The SRS reflects stakeholders' needs in developing an application with a clear description of the provided services. Generating the requirements is based on the users' actual work in the application domain [11]. Requirements Engineering is the process of extracting, gathering, analysing, specifying, validating and maintaining requirements [15].

This paper presents the iSRS system reasoning framework that employs a fuzzy logic technique within the CBR cycle with the objectives of improving its reasoning and decision-making process of the iSRS system [17–21]. The main aim of the iSRS system is to ensure that the SRS meets the satisfaction level of the stakeholders and users and eases the developers work by inspecting the SRS quality of software projects [16].

The iSRS system provides an interface for users to enter the new SRS descriptions. The system operates a parser to generate features of the SRS description and built a new case. Then the CBR cycle of the iSRS system begins with matching the new SRS inspection case with the cases that are stored in the case base (i.e., past cases). Figure 1 exemplify the CBR cycle of the iSRS system.



**Fig. 1.** The architecture of the CBR system.

Heuristic search is performed to find the relevant SRS cases to the new case. Based on Fig. 1, the search algorithm, the system classifies searched cases into three types: (i) complete-match in which the new case has a complete match with one case in the case base, (ii) partial-match in which there are some cases that partially matched with the new case and (iii) no-match in which there is no case matched with the new case [3, 18]. In the first type, if there is a complete-match, then the solution that the retrieved case has is directly used for the new case and the reasoning cycle has a successful retrieve operational state. More so, in the second type, if there is a partial-match then the cases that partially match the new case are revised to fit the new case. Here, the system performs two levels of matching: cases matching level and features matching level. These two matching levels are performed by the fuzzy logic in order to further explore the data of the cases and to find the patterns that most fit the new case. The outcome of the revision process is considered as a solution and the reasoning cycle of the CBR has a successful revise operational state. Furthermore, the new solution is retained in the case base as a possible solution or part of the solution(s) to future cases.

After the revision, the retaining step ensures that the system makes the matching process between the solution case and the other cases in the case base to prevent case overlap and redundancy. The retain or retaining may contain unlearning operation applied to the case base in order to eliminate less needed cases and to prevent overflow status in the case base. Finally, If there is a no-match case then the CBR reasoning cycle ends up unsuccessful and with no solution. Subsequently, the system notifies the user that the new case does not match any case that the system has. It ignores the new case information and implicitly generates a standard solution as a proposed solution. The no-match solution is an example that helps the user on how to write an SRS in high quality and highlights the most common problems. This example helps the user to manually revise the SRS and resubmit it to the system for inspection.

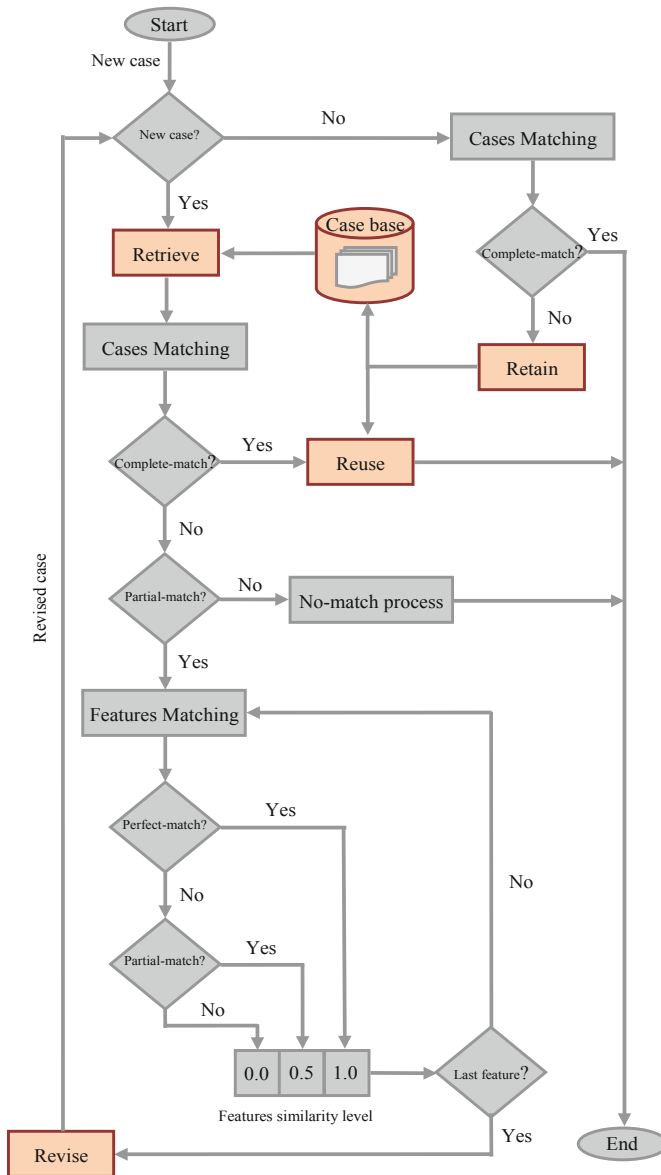
## 4 The Modeling of Fuzzy Logic

The theory of fuzzy logic is applied to a particular problem to remove the fuzziness that the problem might have [19, 20]. 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, data classifiers, and etc. According to Rich, Knight, and Nair [12], “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.” The CBR technique has all the potentials and features of a complete intelligent system [6, 21].

In the iSRS system, fuzzy logic resides in the CBR cycle. As a result, the environment that the fuzzy logic is situated in has an effective impact on its performance level. Moreover, by applying fuzzy set logic to the ideal CBR type, system cases can be more understandable and they become as configurations of attributes that appear in a different extent. Thus, the differences in cases kind and degree can also be easily studied and understood [13]. The following sub-sections illustrate how fuzzy logic is implemented in the CBR cycle.

### 4.1 The Fuzzy Logic in the CBR

In the Retrieve step, the system starts its cycle by checking the features that each case has to measure the similarity level between the existing cases and the problem case. In the proposed framework, fuzzy logic starts matching these cases using the obtained features 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 as shown in Fig. 2. Adjusting the fuzziness of the matching features depends on some relations 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 is also required as illustrated in Fig. 2.



**Fig. 2.** The fuzzy logic implementation chart diagram.

In short, fuzzy logic in the CBR works on the following:

- 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 (no-match, partial-match and perfect-match).

- Enhancing the input parameters of the Revise step by finding features similarity levels.
- In Retain step, matching case base cases again (if necessary) to assist the unlearning process by deciding which case needs to be eliminated.

## 4.2 Fuzzy Cases Matching

The iSRS starts the reasoning cycle when a user enters the required information to inspect the quality of the SRS which is represented by both QIM and QIC sessions. The SRS quality description is extracted by inspecting the eleven QIM and the ten QIC items. The QIM is represented by 61 QII values (each of the QII has a range of values from 1 to 5, where 5 is the highest value). The QIC is represented by 50 questions (each of the questions has a range of values from 1 to 5, where 5 is the highest value).

Ultimately, each case in the case base is represented by 21 features (i.e. 11 items of the QIM and 10 items of the QIC), and each feature takes several elements and the overall elements in the case are 111 elements (i.e. 61 elements of the QIM and 50 elements of the QIC). In the SRS quality measurements, the elements of each case are weighted with a variable  $w$ . The fuzzy logic approach determines cases classifications according to dynamic  $w$  (i.e.,  $w$  values are changing according to exploring processes that are held in retrieving the relevant cases). The  $w$  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.

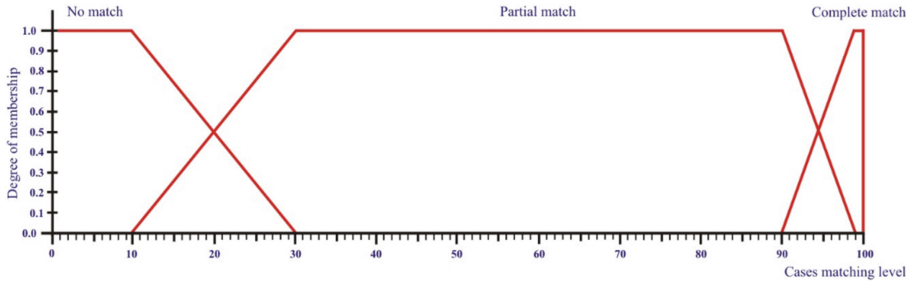
After retrieving the relevant cases, the fuzzy logic starts matching process of the cases in the case base. However, within the CBR cycle, matching processes are required in the Retrieve and Retain steps only (as shown in Fig. 2) in order to ensure efficient retrieval and storing of cases. During the Retrieve step, the cases are classified into three groups; no-match, partial-match and complete-match cases. Table 1 shows a sample of fuzzy sets of the CBR cases matching results. The approach classifies all the relevant cases in the case base to the three matching categories or classes.

**Table 1.** The cases classification of the fuzzy logic.

Fuzzy sets of CBR cases matching result		
<i>no-match</i>	<i>partial-match</i>	<i>complete-match</i>
1. $CN_1$	4. $CP_1$	7. $CC_1$
2. $CN_2$	5. $CP_2$	8. $CC_2$
3. $CN_i$	6. $CP_j$	9. $CC_k$

In Table 1, CN represents no-match cases; i represent no-match cases number; CP represents partial-match cases; j represents partial-match cases number; CC represents complete-match cases, and k represents complete-match cases number. The sum of i, j, and k are equal to the total number of relevant cases. The actual range of the no-match state starts from 0% until 10% matching level as shown in Fig. 3.



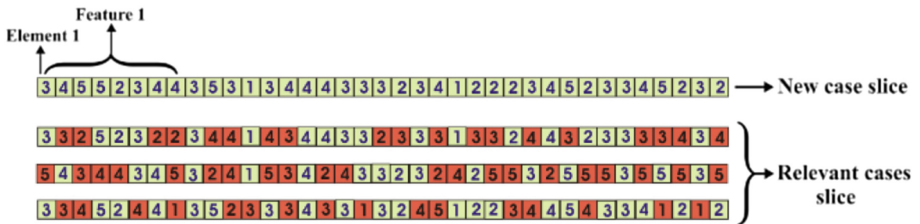


**Fig. 3.** The fuzzy cases matching.

From Fig. 3, cases matching 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%. The complete-match actual value is 100% case matching and it shares another 9.9% fuzzy area with partial-match case state.

The cases matching classifier is used to find the matching level of each case in order to find the solution case(s). If there is a complete-match state, the solution is directly used by the system and if the system does not find a complete-match case then, the highest-level matching case(s) are to be used to find some solutions.

Figure 4 is an example of how the system performs the similarity measurement to determine the relevant cases where each green scale area in the figure shows the matched elements in the relevant cases as compared with the new case.



**Fig. 4.** An example of the fuzzy cases matching results.

Then, in the features matching process, each partial-match relevant 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. 2.

The evaluation to the performance of the cases selection, retrieval and similarity measurement handled by the illustrated fuzzy logic approach, the iSRS system runs over 64 cases with the possibility of showing the three situations (no-match, partial-match and complete-match). The approach is able to filter out 40% of the irrelevant cases as the cases similarity level remains within the no-match range. The framework is

able to reduce similar cases to approximately 5%. As a result, the inspection quality of the iSRS system is clearly improved. The inspection accuracy is increased by 11.6% on average in the performed 10 tests comparing with the original system.

## 5 Conclusion and Future Work

This paper has contributed to finding and implementing a new framework that is used in the Inspection of the Software Requirements Specification (iSRS) System. In this framework, both Case-Based Reasoning (CBR) and fuzzy logic are used in the inspection process. The CBR is used in reasoning the SRS quality by benefiting from experiences that are stored in its case base. Moreover, fuzzy logic makes the CBR and the overall system more effective by assisting in the classifications of the cases in the case base in finding the most similar cases that match the new given problem case. Hence, the accuracy of the Retrieve, Reuse, Revise and Retain steps within the CBR is increased. As part of future prospects, experiments with other techniques in refining iSRS performance are to be studied.

**Acknowledgements.** This project is partially supported by University Tenaga Nasional (UNITEN) under the UNIIG Grant Scheme No. J510050772.

## References

1. Galin, D.: Software Quality Assurance: From Theory to Implementation. Pearson Education Limited, London (2004)
2. Wilson W.M., Rosenberg, L.H., Hyatt, L.E.: Automated analysis of requirement specifications. In: Proceedings of the 19th International Conference on Software Engineering, pp. 161–171. ACM (1997)
3. Mat Jani, H., Mostafa, S.A.: Implementing case-based reasoning technique to software requirements specifications quality analysis. *IJACT: Int. J. Advance. Comput. Technol.* **3**(1), 23–31 (2011)
4. Stephen, E., Mit, E.: Framework for measuring the quality of software specification. *J. Telecommun. Electron. Comput. Eng. (JTEC)* **9**(2–10), 79–84 (2017)
5. Nordin, A., Zaidi, N.H.A., Mazlan, N.A.: Measuring software requirements specification quality. *J. Telecommun. Electron. Comput. Eng. (JTEC)* **9**(3–5), 123–128 (2017)
6. Firesmith, D.G.: Specifying good requirements. *J. Object Technol.* **2**(4), 77–87. [http://www.jot.fm/issues/issue\\_2003\\_07/column7/](http://www.jot.fm/issues/issue_2003_07/column7/)
7. Fitzgerald, B., Stol, K.J.: Continuous software engineering: a roadmap and agenda. *J. Syst. Softw.* **123**, 176–189 (2017)
8. Boegh, J.: A new standard for quality requirements. *IEEE Softw.* **25**(2), 57–63 (2008)
9. Daramola, O., Moser, T., Sindre, G., Biffl, S.: Managing implicit requirements using semantic case-based reasoning research preview. In: Regnell, B., Damian, D. (eds.) *REFSQ 2012*. LNCS, vol. 7195, pp. 172–178. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-28714-5\\_15](https://doi.org/10.1007/978-3-642-28714-5_15)
10. Bonissone P.P., Ramon L.M.: F4.3 Fuzzy Case-Based Reasoning Systems. Citeseer (2008). <http://www.mendeley.com/research/f4-3-fuzzy-casebased-reasoning-systems/>. Accessed 9 Aug 2011

11. John Yen, C.T., Tiao, W.A.: House of quality: a fuzzy logic-based requirements analysis. Elsevier, *Eur. J. Oper. Res.* **117**, 340–354 (1999)
12. Karsak, E.E.: Fuzzy multiple objective programming framework to prioritize design requirements in quality function deployment. Elsevier, *Comput. Ind. Eng.* **47**, 149–163 (2004)
13. Sen, C., Baracl, H.: Fuzzy quality function deployment based methodology for acquiring enterprise software selection requirements. Elsevier, *Exp. Syst. Appl.* **37**, 3415–3426 (2010)
14. Dhote, P.C.: Handling ambiguous data during requirements verification using fuzzy logic. *Int. J. Comput. Sci. Commun.* **2**(1), 105–107 (2011)
15. Asghar, S., Umar, M.: Requirement engineering challenges in development of software applications and selection of customer-off-the-shelf (COTS) components. *Int. J. Softw. Eng. (IJSE)* **1**(2) (2010)
16. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. *Comput. Hum. Behav.* **51**, 915–929 (2015)
17. Mostafa, S.A., Mustapha, A., Mohammed, M.A., Ahmad, M.S., Mahmoud, M.A.: A fuzzy logic control in adjustable autonomy of a multi-agent system for an automated elderly movement monitoring application. *Int. J. Med. Inform.* **112**, 173–184 (2018)
18. Mostafa, S.A., Ahmad, M.S., Firdaus, M.: A soft computing modeling to case-based reasoning implementation. *Int. J. Comput. Appl.* **47**(7), 14–21 (2012)
19. Ghani, M.K.A., Mohammed, M.A., Ibrahim, M.S., Mostafa, S.A., Ibrahim, D.A.: Implementing an efficient expert system for services center management by fuzzy logic controller. *J. Theor. Appl. Inf. Technol.* **95**(13) (2017)
20. Mostafa, S.A., Darman, R., Khaleefah, S.H., Mustapha, A., Abdullah, N., Hafit, H.: A general framework for formulating adjustable autonomy of multi-agent systems by fuzzy logic. In: Jezic, G., Chen-Burger, Y.-H.J., Howlett, R.J., Jain, L.C., Vlacic, L., Šperka, R. (eds.) *KES-AMSTA-18 2018. SIST*, vol. 96, pp. 23–33. Springer, Cham (2019). [https://doi.org/10.1007/978-3-319-92031-3\\_3](https://doi.org/10.1007/978-3-319-92031-3_3)
21. Mohammed, M.A., et al.: Genetic case-based reasoning for improved mobile phone faults diagnosis. *Comput. Electr. Eng.* **71**, 212–222 (2018)