Intelligent Method for Software Requirement Conflicts Identification and Removal: Proposed Framework and Analysis

Maysoon Aldekhail and Djamal Ziani,

King Saud University, Riyadh, Saudi Arabia

Summary
Requirement engineering has recently assumed a significant role in software engineering. In software development, requirements should be correct, complete and consistent. Consistency refers to requirements without any conflicts or contradictions. Requirement consistency is a critical factor in project success as any conflict may waste cost, time and effort. This paper will propose a novel intelligent approach in finding and solving conflicts in functional requirements. The approach works at two levels; a rule-based system to detect the conflicts in functional requirements; and the application of genetic algorithm to resolve conflicts and optimize the set of function requirements to produce minimum conflicts

Key words:
Requirement conflicts, genetic algorithm, rule-based system

1. Introduction

Conflicts among requirements are a serious concern for project success. In requirement engineering, the term conflict involves inference, interdependency, and inconsistency between requirements [20]. In a recent research study [21], a very high number of conflicting requirements was identified among software projects. It was reported to have discovered n2 conflicts in n requirements. Another research study [19] has reported 40% to 60% of requirements in conflict. Previous studies have stated that one of the main reason for high project cost and time is the failure in managing requirement conflicts [6]. To prevent repetition of all the phases, it is important to detect and resolve conflicts in early phases of the project's lifecycle [15]. Many research studies have shown the risks of working with requirements that are in conflicts with other requirements. These risks include overtime or over budget which can lead to project failure. At the very least, it would result in extra effort being expended. The requirement phase is the most critical phase of the software development cycle because the quality of the requirements phase affects the overall quality of the software. Wrong or incomplete requirements may cause incomplete or incorrect project [3].

The literature review [2] demonstrated that most techniques proposed to decrease the risks and detect requirements conflicts are manual. Thus, this takes a lot of time and effort for the software engineering techniques whereas the automated approaches are tools based on human analysis. However, these may incur costs to the project due to human error and wrong decision making. Moreover, most of the proposed approaches have not been evaluated to measure their rate of efficiency. No previous works have used Artificial Intelligence (AI) techniques to find or resolve conflicts. The application of AI techniques in Requirement Engineering (RE) is an emerging area of research that includes the development of ideas across two domains.

By applying an artificial intelligence technique to detect and resolve conflicts in requirements, it would replace human beings and thus, save a lot of time and effort for engineers. Additionally, this increases the quality of analysing the requirements, which in turn provides more accurate results in detecting and resolving conflicts. Moreover, using artificial intelligence technique would lack the human side that uses rational thinking and thus, reduce costs the project would have incurred due to human error and incorrect decisions. Artificial intelligence techniques are self-learning and evolving; these will provide better solutions and make reusing them easier. In addition, it would reduce the cost for hiring experts in requirement management.

The structure of the rest of the paper is as follows: Section 2 provides a description of requirements conflicts and the current research in detecting them while Section 3 presents the current requirements conflict resolution and their critique. Section 4 offers a review on rule-based systems and its application in requirements engineering. An overview of genetic algorithms and its different application in software engineering is provided in Section 5. The following section expands on the new approach and the potential benefits of applying this method. Finally, we conclude with recommendations for the future in Section 7.

2. Requirement Conflict Identification

Successful development of software systems requires complete, consistent and clear-cut requirements. Conflicting requirements is a problem that occurs when a requirement is inconsistent with another requirement [28]. Kim, Park, Sugumaran and Yang provide a useful definition

Manuscript received December 5, 2017
Manuscript revised December 20, 2017

of requirements conflict, “The interactions and dependencies between requirements that can lead to negative or undesired operation of the system.” [18]. Aldekhail, Chikh and Ziani provided general classifications for requirements conflicts based on types of requirements, functional requirements and non-functional requirements [2]. An example of conflicts in non-functional requirements is security (privacy metric) with usability (ease of function learning metric), so there is a compromise. However, the developer must choose an acceptable solution to find the right balance of attributes that work.

Many research studies are trying to find a new method to define and detect conflicts between requirements. In [2], the paper provides an overview on the previous research conducted in this area. It has analysed and classified twenty-two different techniques into different categories. The categorization is based as follows:

- The first classification is based on the method that is used to identify the conflicts, either manually by the requirement engineers or automatically using software tools.
- The second classification is focused on the type of requirements that the technique will be applied to: functional or non-functional requirements.
- The third classification is to determine the scope of the proposed approach to study if it covers the detection problem, to review detection and analysis of the conflicts requirements into different conflict types, and confirm if the proposed approach offers a resolving technique.
- The fourth classification is based on the representation type used for requirements. If the technique uses a particular formalization form, it structures the requirements in a particular model, or it uses an ontology.

The literature review [2] has established that most techniques that are proposed to detect requirements conflicts are manual techniques that take extensive time as well as effort and may cause delays in the project. In addition, these are considered fallible since there is human effort involved. Some conflict techniques have built in some tools trying to automate the detection process. Thus, this would decrease the human effort and time. However, all the automation approaches are still based on human analysis to detect and resolve conflicts.

3. Requirement Conflict Resolution

In order to provide a complete picture about how conflicts are solved practically, different techniques are proposed by experts and software engineers. Described below are some techniques that are used to solve conflicts between requirements.

Sameer Abufardeh from University of Minnesota Crookston offered a few techniques from his experience and from the literature [1]:

- Using a process called rethinking the requirements: By going back to the sources of the conflicting requirements and trying to understand it and thereafter, addressing it differently.
- Getting all the stakeholders in one place and making them discuss and analyze the trade-offs amongst the conflicting requirements, and coming up with prioritization process with regards to the value to the project, cost, time, etc.
- Trying to replace two or more conflicting requirements with a single one that addresses the goals of the conflicting requirements

On the other hand, Samuel Sepúlveda from Universidad de La Frontera had other options as follows [1]:

- Using group-techniques such as focus group, brainstorming, KJ method, workshops, etc.
- Using a win-win model.
- Using GORE and i* diagrams to share goals and objectives with the stakeholders.

David Espina proposed deploying a prioritization method that scored each requirement with regards to the value, cost, and risk for the organization [10].

Jeff Grigg claims that one should prioritize their business goals, and then trace the requirements back to the business goals that they are trying to achieve. The next step would be to assign a higher priority to the requirement that traces back to a higher-priority business goal [12]. When conflicts are detected, negotiation for conflicts resolution can be conducted either by selecting alternatives or re-evaluating priorities.

Papa, Daniels, and Spiker presented some management methods that are used for negation and conflict resolution like theory x, where the managers are responsible for resolving conflicts between employees and decision making about what to follow up with to the management [22].

However, this method is not very popular due to its negative point of view from the employees.

Another theory called 'theory y' was introduced where the whole responsibility for resolving conflicts was given to the employees. Theory z was added as a point to theory y where goals were set for employees before they started working and how they should always work on achieving them. Finally, theory w worked through mutual consideration and as such, it makes everyone a winner due to its nature.

Pair wise comparison method (PCM) was mentioned in [14] to be used in requirements conflict resolution. It used a matrix where the requirements are listed with their priorities for each stakeholder.

Another technique used was the Win-win model, which used 'theory' and was based on the idea that everyone is a winner. It has four steps: identify conflict issues, exploring options in architectural strategies, reaching agreements and eliciting win-win conditions.

Finally, we can see that conflict resolution is based on two main techniques: negotiation between stakeholders, and application of prioritization in requirements based on business goals and objectives. However, there is a lack in resolving techniques; all the existing ones are manual techniques and usually mere guidelines to help software engineers fix problems. No works exist that resolve the conflicts automatically. Also, most of them are proposed techniques that are not evaluated for their efficiency in detecting and resolving conflicts.

By studying the limitations in previous works, this research proposes applying artificial intelligence techniques to fix this gap in the area of requirements conflicts.

4. Rule-Based Systems

The rule-based system is the simplest form of artificial intelligence that uses rules as a way of representing the knowledge that is saved in the knowledge base [13]. A rule-based system depends on the expert system idea that mimics the reasoning behind the human expert's decisions in problem solving and decision making.

The research on applying rule-based systems in requirements engineering have mostly used rule-based systems for verification purposes. Wang, Bai, Cai, and Yan presented a rule-based expert system to help evaluate software quality, and their evaluation results showed an improvement in design efficiency [32]. Chan et al. proposed a new requirement modelling approach called rule-based behaviour engineering to formally model requirements and provided a tool for communication among stakeholders [6]. Dzung and Ohnishi proposed a method for using rule-based system to verify the correctness of requirement ontology [7]. By increasing the size of ontology, it becomes difficult to check the accuracy of information stored in it.

5. Computational Intelligence and Genetic Algorithm

Computational Intelligence (CI) is a sub-branch of artificial intelligence, which is also known as soft computing. It refers to the ability of a computer to learn a specific task from data or experimental observation [23]. Computational intelligent has many paradigms, neural networks, evolutionary algorithms, swarm intelligent, and fuzzy systems [9]. Figure 1 displays the CI paradigms and the evolutionary algorithms (evolution computing) as a subdivision of soft computing:

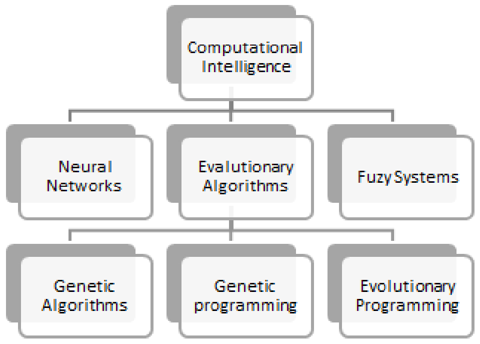

Fig .1 Computational intelligent techniques.

The objective of the evaluation algorithms is to mimic the process from natural evolution, where the main idea is the survival of the fittest and how the weak will eventually die [8]. Evolution is an optimization process where the goal is to improve the ability of a system to survive in a dynamically changing and competitive environment [16].

In the domain of search techniques, evolutionary algorithm is a family of stochastic search techniques that mimic the natural evolution proposed by Charles Darwin in 1858. The following classification (Figure 2) indicates the position of evolutionary algorithms in the area of search techniques:

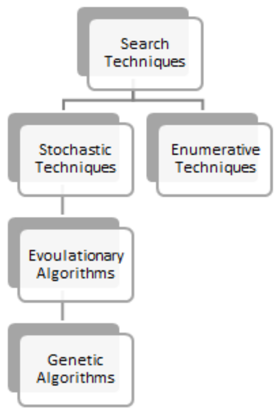

Fig. 2 Search techniques.

Genetic algorithm is a search-based optimization technique based on the principle of genetics and natural selection. It is usually used in optimization problems where there is a need to maximize or minimize a given objective function value under a given set of constraints [17]. Genetic algorithms start with guesses and attempts to improve these guesses by evolution. It is one of the most powerful methods with which high quality solutions are quickly created in response to a problem [27].

There are some basic terminologies that will be used while working with genetic algorithm [5] as follows:

- **Search space** – All possible solutions to the specific problem
- **Population** – It is a subset of all the possible solutions to the given problem.
- **Chromosome** is one such solution to the given problem.
- **Gene** is one element position of a chromosome.
- **Allele** is the value a gene takes for a particular chromosome.

According to Goodman, GA essentially includes the following [11]:

1. Representation of a solution called a chromosome; this should be represented in specific data structure or in binary.
2. An initial set of solutions; an initial population is usually build randomly
3. The fitness function; measures the fitness of any proposed solution to meet the objective.
4. The selection function; selects which chromosome will participate in the next evolution phase.
5. The crossover operator; used in reproduction new chromosome by exchanging genes from two chromosomes.
6. The mutation operation; changes a gene in a chromosome and in turn, creates new chromosome.
7. The termination condition; determines when a genetic algorithm run will stop running.

A pseudo-code for a basic algorithm for a genetic algorithm is as follows [30]:

```
GA()
Initialize population
Find fitness of population
While (termination criteria is reached) do
    Parent selection
    Crossover with probability pc
    Mutation with probability pm
    Decode and fitness calculation
    Survivor selection
    Find best
Return best
```

Figure 3 below presents the flowchart of the basic genetic algorithm.

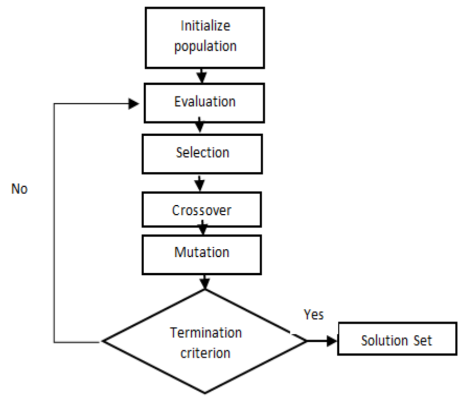

Fig. 3 Flowchart of basic GA.

Genetic algorithm has mostly been applied to the scheduling and optimization of problems and for searching problems like TSP [24]. In [25], genetic algorithm technique has been used for conflict identification and resolution for project activities.

[26] and [27] have both presented a list of some applications of GAs in software engineering and the benefits of applying them. Many research studies have used GA in project effort and time estimation which is one of the most challenging aspects in software development, and the results were very positive. Also, different research studies using GA to help measure the performance of the system by applying GA in software metric in design, coding, quality, reliability and maintenance are presented. GA has had very good results in software testing.

Sharma, Sabharwal and Sibal demonstrated that GA has been used in all types of software tests, functional tests, model-based test case generation, regression testing, object-oriented unit testing as well as in black box testing [29]. Software testing is laborious and time-consuming work; it spends almost 50% of software system development resources [31]. Research has shown how this percentage generally decreases by applying AI techniques and especially when using GA.

6. Proposed Intelligent Conflict Identification and Removal Framework

As sections have shown, there are limitations in the previous works in requirements conflicts identification and resolutions, and that there is a need for applying AI technique in this area and its benefits. Thus, we will work on function requirements. The proposed solution is divided into two parts:

1. *Defining requirements conflicts*: The proposed solution is to build a rule-based system in if-then form based on discussion with experts in requirements engineering regarding the definition of conflicts between two function requirements. A set of rules will be defined, and these rules will determine when requirements are in conflict.
2. *Resolving requirements conflicts*: We are searching for optimum solution via alternative solutions, and what these optimization techniques do. Optimization algorithm searches for an optimal solution by an iterative process. We will use a genetic algorithm to solve the conflicts in requirements intelligently.

Figure 4 below shows the basic structure of the proposed model for detecting and resolving requirements conflict:

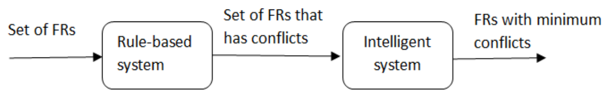

Fig. 4 Basic structure of proposed approach.

6.1 Algorithm for Proposed Approach

The main steps in the proposed approach are as follows:

Part A:

1. Develop the rule-based system (if-then-else) to detect conflicts between function requirements.
2. Read the input (function requirements set) from the excel file
3. Calculate the number of conflicts in the original function requirements set, and list the function requirements that have conflicts and the rule number that detects the conflicts

Part B:

1. Build Initial population randomly by generating attributes within the domain.
2. Find Fitness
3. Apply genetic algorithm until least one conflict solution is found.

The algorithm for proposed model is as follows:

```
Start
Get Input from Excel
Calculate conflict
Initial population is built randomly by generating values for attributes within domain.
Run Fitness (Conflict on each solution)
Select Solutions for GA
Apply Crossover between FRs
Apply Mutation between FRs
Repeat Fitness to Mutation until Stopping Criteria
Stop
```

Figures 5 and 6 below show the flowchart for each part of the proposed model:

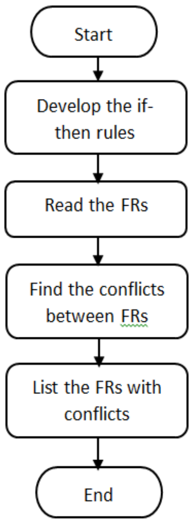
Start
Develop the if-then rules
Read the FRs
Find the conflicts between FRs
List the FRs with conflicts
End

Fig. 5 Flowchart of part (A), Rule-based system.

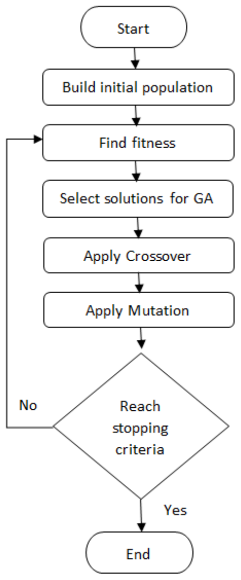

Fig. 6 Flowchart of part (B), Intelligent system.

6.2 Theoretical Analysis of Proposed Approach

By applying the proposed approach of using Rule-based system and genetic algorithm, we expect that we would receive the benefits of an automated process of finding the conflicts between requirements with a simulation to expert work completed intelligently. This will reduce human error, time, and effort of the software engineers.

Also, we expect positive results by applying genetic algorithm in reducing the number of conflicts as much as possible to reach an optimal solution, and to resolve any conflicts. Furthermore, all previous research studies that have applied AI techniques in solving requirement problems in requirement engineering field reported positive results.

Using a genetic algorithm in resolving requirement conflicts will automate the task intelligently and increase the quality of the software development because it will remove human input, which would then provide accurate results in defining and solving conflicts. Also, it will eliminate the emotional side in solving the conflicts between different stakeholders and save costs due to human errors and inefficient decisions. Artificial intelligent techniques are self-learning and improving which allows us to keep reusing them and would thereby, reduce the cost of hiring experts.

7. Conclusion and Future Works

Working with inconsistence requirements will cost the project a lot; from time and effort expended which eventually leads to project failure. This novel approach proposes using an artificial intelligence technique in defining and resolving functional requirements technique. A rule-based system can be used to identify the conflicts and a genetic algorithm can be employed to resolve conflicts and produce a set of functional requirements with a minimum number of conflicts. Applying artificial intelligent technique would increase project efficiency, quality and reduce human effort and errors. In future works, the proposed approach will test and check the results on different sets of functional requirements within different projects.

Acknowledgements

The authors would like to thank the College of Computer and Information Sciences and the Research Center at King Saud University for their sponsorship.

References

[1] S. Abufardeh and S. Sepúlveda. “How to Deal with Stakeholders Conflicts in Requirements Gathering?” https://www.researchgate.net/post/How_to_deal_with_stakeholders_conflicts_in_requirements_gathering2, accessed Sept. 2. 2016
[2] M. Aldekhail, A. Chikh and D. Ziani, D. “Software Requirements Conflict Identification: Review and Recommendations.” IJACSA, 7(10), pp. 336 – 335, 2017.

[3] A.A. Alshazly, A.M. Elfatatry and M.S. Abougabal. "Detecting defects in software requirements specification." AEJ, 53, pp. 513 – 527, 2014.
[4] W. H. Butt, S. Amjad and F. Azam. Requirement conflicts resolution: using requirement filtering and analysis. New York, U.S.A.: Springer. 383 – 397, 2011.
[5] J. Carr, "An introduction to genetic algorithms." Senior Project, pp. 1 – 40. https://www.whitman.edu/Documents/Academics/Mathematics/2014/carrjk.pdf, accessed Aug. 21. 2015
[6] L.W. Chan, R. Hexel and L. Wen. "Rule-based behaviour engineering: Integrated, intuitive formal rule modelling." Proc. 22nd Australian Software Engineering Conf. (ASWEC), Hawthorne, Victoria, Australia, pp. 20 – 29, June 2013.
[7] D.V. Dzung, and A. Ohnishi, "Customizable rule-based verification of requirements ontology." Proc. IEEE 1st Int. Workshop on AIRE. Karlskrona, Sweden. pp. 19 – 26, August 2014.
[8] A.A. El-Sawy, M.A. Hussein, E.S.M. Zaki and A.A. Mousa. "An Introduction to Genetic Algorithms: A Survey A Practical Issues." Int. Journal of SER, 5, pp. 252 – 262, 2014.
[9] A.P. Engelbrecht. Computational intelligence: An Introduction. John Wiley & Sons, 2007.
[10] D. Espina. Scope. "How Do You Manage Conflicting Stakeholder Demands?" http://pm.stackexchange.com/questions/1399/how-do-you-manage-conflicting-stakeholder-demands, accessed Nov. 26. 2016
[11] E.D. Goodman. "Introduction to Genetic Algorithms." Proc. Companion Publication of the 2014 Annu. Conf. on Genetic and Evolutionary Computation. Vancouver, BC, Canada: ACM. pp. 205 – 226, July 2014.
[12] J. Grigg,. "Conflicting Requirements." http://c2.com/cgi/wiki?ConflictingRequirements, accessed Nov. 10. 2016
[13] C. Grosan and A. Abraham. "Rule-Based Expert Systems." Intelligent Systems. Intelligent Systems Reference Library, (17). Springer, Berlin, Heidelberg, pp. 149 – 185, 2011.
[14] F. Hameed and M. Ejaz. "Model for conflict resolution in aspects within Aspect Oriented Requirement engineering" (Master's thesis). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.476.4292&rep=rep1&type=pdf, accessed Aug. 6. 2016
[15] M. Heisel and J.A. Souquières. "Heuristic Algorithm to Detect Feature Interactions in Requirements." In: S. Gilmore and M. Ryan, eds., Language Constructs for Describing Features. Springer, London, pp. 143 – 162, 2001.
[16] M.A. Iqbal, N.K., Khan, M.A. Jaffar, M. Ramazan and A.R. Baig. "Opposition Based Genetic Algorithm with Cauchy Mutation for Function Optimization. Proc. 2010 Int. Conf. on Information Science and Applications. Hotel Rivera, Seoul, South Korea, pp. 1 – 7, April 2001.
[17] H. Jiang. Can the Genetic Algorithm Be a Good Tool for Software Engineering Searching Problems? Proc. 30th Annu. Int. COMPAC. Chicago, USA, 2, pp. 362 – 366, September 2006.
[18] M. Kim, S. Park, V. Sugumaran and H. Yang. "Managing requirements conflicts in software product lines: A goal and scenario based approach." Data & Knowledge Engineering, 61, pp. 417 – 432, 2007.
[19] D. Mairiza and D. Zowghi. "An ontological framework to manage the relative conflicts between security and usability requirements." Proc. 3rd Int. Workshop on MARK, Sydney, Australia, pp. 1 – 6, September 2010.
[20] D. Mairiza, D. Zowghi and N. Nurmuliani. "Managing conflicts among non-functional requirements." In L. Ngyen, D. Randall and D. Zowgi, eds., ARWE 2009 Proc. 12th AWRE, Sydney, Australia: University of Technology, Sydney, October 2009.
[21] T. Moser, D. Winkler, M. Heindl and S. Biffl. "Requirements Management with Semantic Technology: An Empirical Study on Automated Requirements Categorization and Conflict Analysis." Proc. from 23rd Int. Conf. AISE, Berlin, Springer-Verlag, Berlin, Heidelberg, pp. 3 – 17, June 2011.
[22] M.J. Papa, T.D. Daniels and B.K. Spiker. Organizational communication perspectives and trends. SAGE Publications, Inc, 2008.
[23] H.M. Pandey. "Solving lecture time tabling problem using GA." Proc. 6th Int. Conf. – Cloud System and Big Data Engineering, Noida, India, Amity University, Department of Computer Science & Engineering, pp. 45 – 50, January 2016.
[24] R. Raghavjee and N. Pillay. (2008, October). "An Application of Genetic Algorithms to the School Timetabling Problem." Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology. Wilderness, South Africa. New York, U.S.A.: ACM. pp. 193 – 199, October 2008.
[25] M. Ramazan, M.A. Iqbal, M.A. Jaffar, A. Rauf, S. Anwar and A.A. Shahid. "Project Scheduling Conflict Identification and Resolution Using Genetic Algorithms." Proc. Int. Conf. on Information Science and Applications. Hotel Rivera, Seoul, South Korea, pp. 1 – 6, April 2010.
[26] P. Reena, K. Bhatia. "Application of Genetic Algorithm in Software Engineering: A Review." IRJES, 6, pp. 63–69, 2017.
[27] Samriti. "Applications of Genetic Algorithm in Software Engineering, Distributed Computing and Machine Learning." IJCAIT, vol. 9, no. 2, 2016.
[28] B. Schär. "Requirements Engineering Process HERMES 5 and SCRUM." (Master's thesis). University of Applied Sciences and Arts, Northwestern Switzerland, 2015.
[29] C. Sharma, S. Sabharwal and R. Sibal. "Applying genetic algorithm for prioritization of test case scenarios derived from UML diagrams." IJCSI, vol. 8, no. 3, pp. 433 – 444, 2011
[30] S.N. Sivanandam and S.N. Deepa. Introduction to genetic algorithms. Berlin: Springer-Verlag Berlin Heidelberg, 2007.
[31] P.R. Srivastava and T. Kim. "Application of genetic algorithm in software testing." IJSEA, vol. 3, no. 4, pp. 87 – 96, 2009.
[32] X. Wang, Y. Bai, C. Cai, and X. Yan. "A production rule-based knowledge system for software quality evaluation." Proc. 2nd Int. Conf. ICCET, Chengdu, China., 6, pp. 208 – 211, April 2010.

Maysoon Aldekhail has been a PhD candidate at King Saud University in the Computer Sciences and Information Systems College since 2013. She received her Master’s degree in Information Systems from King Saud University, Saudi Arabia in 2009. Her current professional occupation is Lecturer at the Information System Department at the College of Computer and Information Sciences in Al-Imam University in Riyadh, Saudi Arabia. Her research interests include Requirements Engineering and ERP.

Djamal Ziani has been an associate professor at King Saud University in the Computer Sciences and Information Systems College since 2009. He is also a researcher in ERP and in the data management group of CCIS, King Saud University. He received a Master’s degree in Computer Sciences from the University of Valenciennes, France in 1992, and a Ph.D. in Computer Science from the University of Paris Dauphine, France in 1996. He has been a consultant and project manager in various companies in Canada, such as SAP, Bombardier Aerospace, and Montreal Stock Exchange from 1998 to 2009.