International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 5, October 2019, pp. 4336∼4343
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i5.pp4336-4343

Optimizing requirement analysis by the use of meta-heuristic in Search Based Software Engineering

Rajesh Kumar[1], Rakesh Kumar[2]
[1]Department of Computer Science & Applications, CRM Jat College, Hisar, Haryana, India
[2]Department of Computer Science & Applications, Kurukshetra, University, Kurukshetra, Haryana, India

Article Info

Article history:

Received Oct 16, 2018
Revised Apr 17, 2019
Accepted Apr 22, 2019

Keywords:

Genetic algorithm
Human based computation
Optimization
Requirement selection
Search Base Software Engineering

ABSTRACT

Requirements analysis is the first phase of software development process and it is one of the main concerns of software engineers. The selection of requirements is a complex problem caused by the heterogeneity of the users and their varied interests and demands. In this paper, it is justified that their is a strong need of optimization in requirement analysis. The paper argues that requirement selection can be viewed as an application area of Search-Based Software Engineering(SBSE). The aim is to justify the claim that requirement engineering can be re-formulated as search problem to which meta-heuristic technique can be applied.

Corresponding Author:

Rajesh Kumar,
CRM Jat College, Hisar, Haryana, India.
Tel: +919896402345
Email: lsntl@ccu.edu.tw

1. INTRODUCTION

Requirement analysis is an important part of the software engineering process. The process of requirement engineering is described as a series of activities including elicitation, modeling & analysis, specification and validation & verification.

In a problem, sometimes the size of requirements is huge and it is very difficult to implement all the requirements. It is also observed that the requirements gathered from heterogeneous users are of varying size. A key aspect in any successful project is to determine an appropriate set of requirements for implementation. The requirements selection needs to be optimized to limit the infinite flow of demands posed by the users. Requirements selection is an engineering process to select an optimal set of requirements out of all the requirements proposed by the customers. The objective of requirement analysis is to address the question: "What do you want to achieve in a product or service?", which however is a subtle task in the praxis.

Hence, a formalized representation technique is required to optimize the requirement selection process. Thus, it is a big challenge to represent these requirements in a formalized way. This justify that Requirements selection can be viewed as an application area of Search-Based Software Engineering (SBSE). Search based Software Engineering (SBSE) seeks to reformulate Software Engineering problems as search problems.There is a need of optimization to find out the perfect requirement set using some metaheuristics. Harman[1, 2] coined a term Search Based Software Engineering (SBSE) in which Search Based Optimisation is applied to Software Engineering. The first objective of this paper is to reformulate the requirement selection as a search based problem.

2. REQUIREMENT SELECTION: A SEARCH PROBLEM

It is assumed that an initial set of requirements has been collected and the requirements have been identified using a requirements elicitation process. The task of optimizing requirements is usually regarded to be a challenging and hard problem in itself[3, 4, 5] Nevertheless, there are still several feasible approaches in previous works that address the solution of this problem[6, 7, 8] but none is absolute. So, in software engineering, determining the set of requirements is a critical foundation for the success of a project. Including or excluding requirements inappropriately may result in the emergence of products that fail to satisfy stakeholders' needs and might cause a huge loss of revenue. However, uncertainty is inevitable in the early phase of requirements engineering and could lead to unsound requirement decisions. So, requirement engineering needs some intelligent techniques to overcome these type of problems. So, first objective is to represent the problem in the form of search space. Then selection will be performed on using some fitness function followed by encoding scheme. Then crossover, mutation and replacement will be take place and repeat these steps until the terminating criteria reached.

The framework proposed in the paper is working in the context of requirement representation in the form of state space. The emphasis of the framework is to provide a systematic method which uses the search-based optimization approach to solve the computational and cognitive complexity of the requirement selection.

3. SEARCH BASED SOFTWARE ENGINEERING

Search Based Software Engineering is the name given by Harman[1] to a oeuvre in which Search Based Optimization is applied to Software Engineering. Optimization approaches have been applied to solve software engineering problems since the 1970s decade[9]. There are different solutions to optimization problem and one solution is searching. According to Harman[10] there are three aspects for formulating software engineering problem as search based Optimization problem.

(a) Problem Representation Technique
(b) Selection of Fitness Function
(c) Choice of Search-Based Technique

3.1. Problem representation

Problem representation should be in the form of state space. Choosing a proper problem representation technique is essential for reconstructing a problem into a search-based optimization problem. To represent the problem in the form of state space firstly state space need to be define.

3.1.1. State space representation

Search-based requirements selection and optimization lies within the general SBSE framework[11]. SBSE is concerned with the application of search-based algorithms to software engineering topics such as Requirement analysis. The purpose of all search algorithms is to explore the optimal, near-optimal or "good enough" solutions among a number of possible solutions in a search space. Search/state space search is formally describes a search problem. A search problem consists of the following:

(a) S: the non empty set of states.
(b) S_0: the initial state where $s_0 \subseteq$ S.
(c) S⇒S is a set of operator.
(d) G is the set of final state. Note that G $\subseteq$ S.

3.2. Use of meta-heuristic

Different meta-heuristic technique are available such as Genetic Algorithms (GA) [12], Simulated Annealing (SA)[13] and Hill Climbing (HC)[14]. In this paper Genetic Algorithm has been proposed to search the state space.

3.2.1. Genetic algorithms

GA is a generalized search and optimization technique. It works with populations (chromosomes) of individuals, each representing a possible solution to a given problem. Each individual is evaluated to give some measure of its fitness to the problem from the objective functions. Algorithm applies the principle of survival of fittest to find better approximations. At each generation, a new set of approximation is created by the process of selecting individual potential solutions according to their level of fitness in the problem domain and breeding

them together using operators borrowed from natural genetics. A simple GA that yields good results in many practical problems is composed of following operators[15]:

(a) Encoding

(b) Selection

(c) Crossover

(d) Reproduction

3.2.2. Encoding schemes-categorization

Depending on the structure of encoding, it can be classified into two groups, namely, one dimensional and two-dimensional. Binary, Value and Permutation encoding is one dimensional and Tree encoding is two dimensional encoding schemes[6]. Studying these encoding schemes, one can infer that characters represented by permutation encoding are position dependent. In Binary encoding, real value encoding, the characters are value oriented. The two factors identified by studying different encoding schemes are locus and value of character in the chromosome. So, factors like locus and value should be kept in mind while encoding a solution for a particular problem.

3.3. Goldberg's classification of encoding techniques

Goldberg has declared in his work that fitness function for a specific encoding scheme is dependent on two factors- value and order[8]. Three different categories of encoding can be grouped depending on fitness evaluation factors such as:

(a) Encoding schemes where fitness depends on value only: f(v). Eg: Value encoding

(b) Encoding schemes where fitness depends on value and order: f(v,o). Eg: Binary Encoding

(c) Encoding schemes where fitness depends on order only: f(o). Eg: Permutation encoding.

4. NEED OF NOVEL ENCODING

It can be stated that the existing encoding schemes fall under the three categories mentioned above and they are dependent on value or order or both factors for assessment of fitness function. These existing encoding schemes are not suitable for the representation of the requirements because there is no mathematical way to represent all the requirements in the form of state space. Consequently, there arises a need to unearth a new encoding scheme that is independent of these two factors. It is a big challenge for the researchers to represent the requirements, gathered from the different users, in the form of a chromosome. The size of the requirements is not fixed. In this purposed encoding scheme, requirements are stored in the form of a SET.

4.1. Prerequisites for purposed encoding scheme

It is assumed that there are N users and M possible system requirements are gathered from users, out of these M requirements some are common and some are altogether different. Every requirement is assign a unique number on the basic of their uniqueness. It is assumed that there is a **SET** of users, N=$\{N_1, N_2, ..., N_n\}$ and a **SET** of possible system requirements, R = $\{R_1, R_2, ..., R_M\}$.

In this new Encoding scheme all the requirements gathered from a user are represented as a chromosomes in raw form. In figure 1, chromosome 1 represent the requirements gathered from user 1 and so on.

Chromosome 1: $\boxed{R_1, R_3, R_6, R_8, R_9, R_{12}, R_{15}}$

Chromosome 2: $\boxed{R_1, R_3, R_4, R_6, R_7, R_{12}}$

Chromosome 3: $\boxed{R_1, R_2, R_5, R_8, R_{10}, R_{16}, R_{20}, R_{23}}$

Figure 1. Representation of the requirements as chromosome

5. FITNESS FUNCTION

It can not be denied that there is uncertainty, inconsistency and ambiguity in the requirement analysis phase. Requirement selection is a complex task as it is difficult to select the fitness function for requirements. No mathematical function is applicable to measure the fitness value of a particular requirement [16]. Thus, in this section, the author tries to justify that Human Based Computation can be used as a fitness function for the genetic algorithm. Human-based computation is a technique in which human computational power is utilized to solve the problems that computers cannot yet solve[17]. Alex Kosorukoff[18], who coined the term, designed a genetic algorithm that allows humans to suggest solutions that might improve evolutionary processes. Since the evolution of Artificial Intelligence(AI) research in the 1950s, computer scientists have been trying to imitate human-like capabilities, such as visual processing, language and reasoning. Alan Turing wrote in 1949:The idea behind computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human mind using mathematical model. [19] The idea of using human effort to perform tasks that computers cannot yet perform[20] is called Human-based computation. It is a technique that makes use of human abilities for computation to solve complex problems[21]. The thesis *"Human Computation"* [17] defines the term as: A paradigm for utilizing human processing power to solve problems that computers cannot yet solve.

Both CAPTCHA (which stands for Completely Automated Public Turing Test to Tell Computers and Humans Apart) and reCAPTCHA are the invention of Luis von Ahn, a Carnegie Mellon computer scientist and MacArthur "genius grant" recipient. "A couple hundred million CAPTCHAs are typed daily around the world," von Ahn tells NEWSWEEK. "The first time I did the calculations, I felt quite proud. And then I felt bad because people really find these annoying." They're also wasteful. It takes about 8 to 10 seconds to type a CAPTCHA more, obviously, if you err and have to start over meaning a total of some 400,000 human hours per day are spent typing them in. As a point of comparison, according to von Ahn, the Empire State Building took 7.5 million human hours to build. "Life is only like 750,000 hours," The Author says. "It's almost the equivalent of a life. One thought, is there any way oneself can use this human effort in a way that's good for humanity?"

Recognizing distorted words is one of the things that the human mind can still do better than computers. In order to make old newspaper, books and other texts searchable, pages are scanned and fed into optical character recognition software. Because ink and paper degrade over time, some words remain incomprehensible. The reCAPTCHA system presents Web users with two words: one word that computers can not read, and one that they can. So long as you type the known word in correctly, and a few other people agree with you on the unknown word, you have helped digitize an archival page. And, Von Ahn says, typing in two words instead of one does not cost you a significant amount of extra time.

It has been said that by oneself should not ask what is next in terms of what the technology and Internet will be able to do but instead try to understand what society already got and figure out how to put it to good use. Von Ahn's efforts surely prove that point. They also show that in some ways, peoples can help computers as much as they help society.

Another example of human based computation is Duolingo. Duolingo is a free-ware language learning platform. It includes a language learning website and application, as well as a digital language competence assessment exam. As of Aug., 2018, the language learning website and application offer 81 different language courses across 36 languages. The application has about 350 million registered users across the world. Duolingo wins Apple's iPhone App of the Year award in 2013.

On the basis of above instances, it is concluded that human based computation is a compelling fitness function in requirement selection. With the ever-increasing number of Internet users, the relevance of man based computation becomes all the more prominent. It is a potential medium to procure the fitness value of requirements.

6. SELECTION

Selection is the first genetic operation in the reproductive phase of genetic algorithm. The objective of selection is to choose the fitter individuals in the population that will create offsprings for the next generation, commonly known as mating pool. The mating pool thus selected takes part in further genetic operations, advancing the population to the next generation and hopefully close to the optimal solution. In other words, Selection is the process of choosing breeding stock or parents from a population. As the generations pass, the

members of population should get fitter and fitter. Individuals from the mating pool are used to generate new offsprings, with the resulting offspring forming the basis of next generation. So it is desirable that the mating pool should have good individuals. Selection operator works at the level of chromosomes. The key idea of selection operator is to give preference to better individuals by allowing them to pass on their genes to the next generation and prohibit the entrance of worst fit individuals into the next generation. The goodness of each individual depends on its fitness. Fitness value is determined by an objective function [21]. Selection of individuals in the population is fitness dependent and is done using different algorithms[8]. Some are roulette wheel selection, rank selection, tournament, steady state selection and many more. Selection acts as active force in a genetic algorithm by regulate the genetic search towards favorable domain in the search space. Selection operator emulate phenomena and processes in nature. Selection chooses more fit individuals in analogy to Darwin's theory of evolution – survival of fittest[22]. All the individuals have a chance of being selected into the mating pool, but there are chances that an individual in the population can be selected more than once depending upon its fitness. Selection schemes are characterised by selection pressure, selection variance and loss of diversity. They primarily determine the convergence characteristics of genetic algorithms. Selection has to be balanced. Too strong selection means suboptimal highly fit individuals will take over the population reducing the diversity and too weak selection will result in too slow evolution[23]. Goldberg and Deb grouped selection methods in to four categories: Proportionate, Ranking, Tournament and Steady state selection.

(a) Roulette Wheel Selection

(b) Rank Selection

(c) Tournament selection

(d) Steady State Selection

6.0.1. Rank selection

Rank selection is used in this paper for selection. Rank Selection sorts the population first according to fitness value and ranks them. Rank N is assigned to the best individual and rank 1 to the worst individual. Then every chromosome is allocated selection probability with respect to its rank[24]. Individuals are selected as per their selection probability. Rank selection is an explorative technique of selection. Rank selection prevents too quick convergence and differs from roulette wheel selection in terms of selection pressure. Rank selection overcomes the scaling problems like stagnation or premature convergence. Ranking controls selective pressure by uniform method of scaling across the population. Rank selection behaves in a more robust manner than other methods[25, 26].

7. CROSSOVER

Genetic algorithms are optimization algorithms and mimic the natural process of evolution. Important operators used in genetic algorithms are selection, crossover and mutation. In the previous chapters, different forms of selections have been discussed. Crossover and mutation operators are used to introduce diversity in the population. Type of crossover and mutation operator used for a problem depends on the type of encoding used. Different crossover operators are described in this section. In natural systems, crossing-over is a complex process that occurs between pairs of chromosomes. Two chromosomes are physically aligned, breakage occurs at one or more corresponding locations on each chromosome, and homologous chromosome fragments are exchanged before the breaks are repaired. This results in a recombination of genetic material that contributes to variability in the population. In genetic algorithms, crossover operator exchanges substrings between chromosomes represented as linear strings of symbols. The basic crossover operation is a three-step procedure[27]. First, two individuals are chosen at random from the population of 'parent' strings generated by the selection operator. Second, one or more string locations are chosen as breakpoints or crossover points delineating the string segments to exchange. Finally, parent string segments are exchanged and then combined to produce two resultant offspring individuals. Crossover operates on selected genes from parent chromosomes and creates new offspring. Simplest crossover may be exchanging genetic material of two strings with respect to single crossover point. Crossover can be quite complicated and depends mainly on the encoding of chromosomes. Specific crossover made for a specific problem can improve performance of the genetic algorithm. Crossover combines parental solutions to form offspring with a hope to produce better solutions. Crossover operators are critical in ensuring good mixing of building blocks[28].

7.1. Uniform crossover

Uniform crossover operator does not divide the parent chromosome into segments for recombination. Rather, it treats each gene of the chromosome independently to choose for the offspring. In Uniform crossover, number of crossover points is not fixed initially. It recombines genes of parent chromosomes on the basis of crossover mask. It selects x number of crossover points in the chromosome where the value of x is a random value less than the length of the chromosome. Crossover mask is generated according to this random value. In this crossover, each gene in the offspring is created by copying the corresponding gene from one of the parents. The selection of the corresponding parent is undertaken via a randomly generated crossover mask[29, 30, 31] . At each index, the offspring gene is taken from the first parent if there is a 1 in the mask at this index, and if there is a 0 in the mask at this index, the gene is taken from the second parent. Due to this construction principle uniform crossover does not support the evolvement of higher order building blocks. Uniform crossover does not exhibit positional bias but do exhibit distributional bias due to which uniform crossover has a strong tendency towards transmitting 50% of the genes from each parent and against transmitting an offspring a large number of co-adapted genes from one parent.

Table 1. Uniform Crossover

Parent 1	$R_1,$	$R_2,$	$R_8,$	$R_4,$	$R_9,$	$R_6,$	R_7	
Parent 2	$R_4,$	$R_2,$	$R_6,$	$R_3,$	$R_5,$	$R_7,$	$R_1,$	R_8
Mask	1	1	0	1	0	1	1	0
Child 1	$R_1,$	$R_2,$	$R_6,$	$R_4,$	$R_5,$	$R_6,$	$R_7,$	R_8
Child 2	$R_4,$	$R_2,$	$R_8,$	$R_3,$	$R_9,$	$R_7,$	R_1	

8. CONCLUSION AND FUTURE WORK

The application of metaheuristics to resolve Software Engineering problems is part of a nearly new field called Search Based Software Engineering (SBSE). In this paper, the author justified the claim that requirement engineering can be re-formulated as a search problem. This paper has presented the novel encoding technique for requirement optimization. The author used the idea of harnessing human computation power in order to solve a problem that computers cannot yet solve. People engage in computation not because they want to do a good deed but because they enjoy it. The future of SBSE is a luminous one. There are many areas to which the techniques associated with SBSE surely apply, but have yet to be fully considered. In existing fields of application, the results are already very encouraging.

REFERENCES

[1] Mark Harman and Bryan F. Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833–839, December 2001.

[2] Sarro, Federica and Kessentini, Marouane and Deb, Kalayanmoy Guest Editorial Special Issue on Search-Based Software Engineering. *IEEE Transactions on Evolutionary Computation*, 3(22):333–333, 2018.

[3] Tom Tourwé, Wim Codenie, Nick Boucart, and Vladimir Blagojević. Demystifying release definition: from requirements prioritization to collaborative value quantification. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 37–44. Springer, 2009.

[4] Thakurta, Rahul Understanding requirement prioritization artifacts: a systematic mapping study. *Requirements engineering*, 4(22):491–495, 2017.

[5] Joachim Karlsson and Kevin Ryan. A cost-value approach for prioritizing requirements. *IEEE Softw.*, 14(5):67–74, September 1997.

[6] Franz Rothlauf and David E. Goldberg. *Representations for Genetic and Evolutionary Algorithms*. Physica-Verlag, 2002.

[7] Pal, Sankar K and Wang, Paul P Genetic algorithms for pattern recognition. *CRC press*, 4(20):499–510, 2017.

[8] T. Gilb. *Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*. Elsevier Science, 2005.

[9] W. Miller and D. L. Spooner. Automatic generation of floating-point test data. *IEEE Trans. Softw. Eng.*, 2(3):223–226, May 1976.

[10] Mark Harman, S. Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.*, 45(1):11:1–11:61, December 2012.
[11] Mark Harman.umbarkar2015crossover The current state and future of search based software engineering. In *2007 Future of Software Engineering*, FOSE '07, pages 342–357, Washington, DC, USA, 2007. IEEE Computer Society.
[12] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.
[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.
[14] Sheldon H. Jacobson and Enver Yücesan. Analyzing the performance of generalized hill climbing algorithms. *Journal of Heuristics*, 10(4):387–405, July 2004.
[15] K.A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.
[16] John J. Grefenstette. Genetic algorithms and machine learning. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, pages 3–4, New York, NY, USA, 1993. ACM.
[17] Luis Von Ahn. *Human Computation*. PhD thesis, Pittsburgh, PA, USA, 2005. AAI3205378.
[18] A. Kosorukoff. Human based genetic algorithm. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, volume 5, pages 3464–3469 vol.5, Oct 2001.
[19] A. M. Turing. Computers & thought. chapter Computing Machinery and Intelligence, pages 11–35. MIT Press, Cambridge, MA, USA, 1995.
[20] Edith Law and Luis von Ahn. Input-agreement: A new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1197–1206, New York, NY, USA, 2009. ACM.
[21] Man-Ching Yuen, Ling-Jyh Chen, and Irwin King. A survey of human computation systems. *2009 International Conference on Computational Science and Engineering*, 4:723–728, 2009.
[22] David B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.
[23] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.
[24] James E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.
[25] Darrell Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
[26] Thomas Bäck. Evolutionary algorithms. *SIGBIO Newsl.*, 12(2):26–31, June 1992.
[27] Conor Ryan, Arthur H.M. van Roermund, and Christina Johanna Maria Verhoeven. *Automatic Reengineering of Software Using Genetic Programming*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.
[28] T. Baeck, D.B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Basic algorithms and operators. Taylor & Francis, 2000.
[29] D.H Ackley. *A Connectionist Machine for Genetic Hillclimbing*, volume SECS28 of *The Kluwer International Series in Engineering and Computer Science.* Kluwer Academic Publishers, Boston, 1987.
[30] Umbarkar, AJ and Sheth, PD CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW.. *ICTACT journal on soft computing*, 6(1):299–301, 2015.
[31] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

BIOGRAPHIES OF AUTHORS

Rajesh Kumar obtained his B.Sc. Degree, Master's Degree (Master of Computer Applications) from Kurukshetra University, Kurukshetra. Currently, He is an Assistant Professor in the Department of Computer Science and Application's, Chhaju Ram Memorial Jat College, Hisar, Haryana, INDIA. His research interests are in Genetic Algorithm, Theory of Automata, Software Engineering, Artificial Intelligence, Design and Analysis of Algorithm and Linux.

Prof. Rakesh Kumar obtained his B.Sc. Degree, Master's Degree – Gold Medalist (Master of Computer Applications) and PhD (Computer Science & Applications) from Kurukshetra University, Kurukshetra. Currently, He is Professor & Chairperson in the Department of Computer Science and Applications, Kurukshetra, University, Kurukshetra, Haryana, INDIA. His research interests are in Genetic Algorithm, Software Testing, Artificial Intelligence, and Networking. He is a senior member of International Association of Computer Science and Information Technology (IACSIT).