Review of Computer Engineering Research
2019 Vol.6, No.1, pp.45-56
ISSN(e): 2410-9142
ISSN(p): 2412-4281
DOI: 10.18488/journal.76.2019.61.45.56

AN EXAMINATION OF THE METHODS OF INCREASING SOFTWARE EFFICIENCY BASED ON SOFT COMPUTING TECHNOLOGY

Shafagat Mahmudova *SE Department, Institute of Information Technology of ANAS, Baku, Azerbaijan.*
Email: shafagat_57@mail.ru

Article History
Received: 12 February 2019
Revised: 15 March 2019
Accepted: 23 April 2019
Published: 3 June 2019

Keywords
Soft computing
Software efficiency
Software development
Programming technology
Soft computing components.

ABSTRACT

This paper examines the methods of increasing software efficiency based on soft computing technology by analyzing the benefits and challenges of the components used in software development. The functions and features of machine learning, comprising neural networks, perceptron, support vector machines, and fuzzy logic are highlighted and explained, along with those of evolutionary computation (e.g., evolutionary and genetic algorithms) and probability. Recommendations are presented in conclusion.

Contribution/Originality: This paper is one of only a few to investigate and analyze the problems of soft computing technologies. By reviewing the utilization of new estimation methodologies, it contributes to the existing literature and comparative analyses.

1. INTRODUCTION

Computers affect all processes in society, including research and economics, generally enabling development in different areas and changing the way people work. Perceptions of new technologies and their application can lead to the development of new systems and software; software is a set of computer programs and related documentation.

Programming technology refers to the tools used in the process from the creative idea to software development, and software development technologies are based on the technical tools and modern automated methods used during their lifetime. Technical tools are also included in the software system, which, together with the hardware system, constitutes the systems environment. However, the high cost of the necessary technologies increases the price of software products.

Different technologies and methods are used when developing high-quality software systems, one of which is soft computing. The concept of soft calculations was introduced by Lotfi Zadeh in 1994, to solve the general class of problems using approximation methods that tolerate partial and imprecise data and are intended for those issues beyond solution, such as biology, medicine, management, and computer science [1]. Figure 1 illustrates the components of soft computing [2, 3]. This paper explores the utilization, benefits, and challenges of soft computing technologies for software development, a brief description of which is provided.

2. ANALYSIS OF AND PROBLEMS WITH SOFT COMPUTING TECHNOLOGIES

2.1. Machine Learning

Machine learning refers to a subset of artificial intelligence methods that are applied for a direct solution of issues based on patterns and inferences, using "training data" to resolve similar problems. The optimization of mathematical statistics, probability theory, graph theory, and other tools are all employed to develop these methods.

The concept of "machine learning" was first used in 1959 by American scholar **Arthur Samuel** (1901–1990), one of the pioneers in the field of artificial intelligence and computer gaming [4, 5].

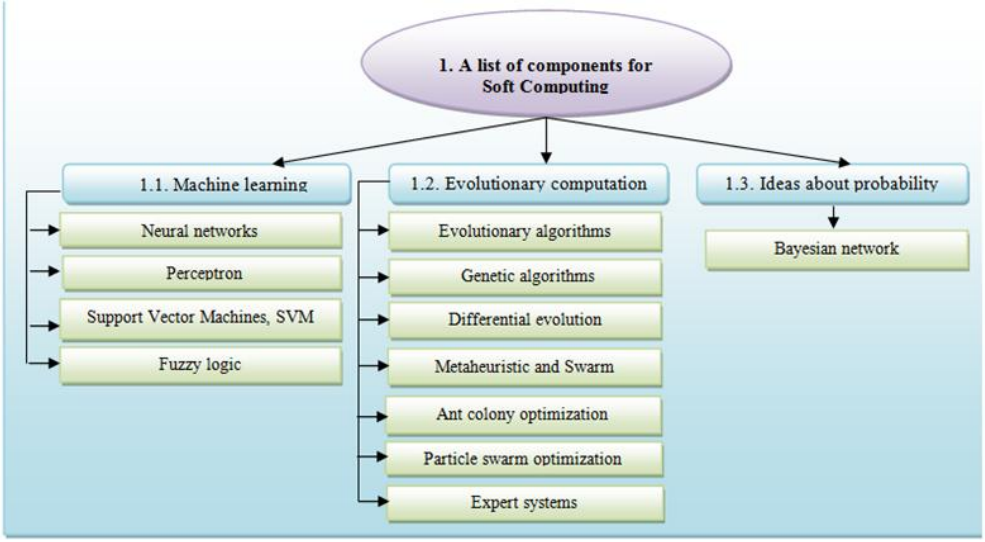

Figure-1. Soft computing components.

Two types of training data are used in this field:

Induction: A method whereby general conclusions are drawn from specific facts or particular coincidences.

Deduction: A term of Latin origin meaning an acquisition, production, or extraction, and one of the key methods for making judgments in research. Unlike induction, this method draws specific logical conclusions from any general idea or proposes a suggestion. The components of machine learning are listed in Figure 1, the first of which is neural networks.

2.1.1. Neural Networks

Artificial neural networks include blended computing systems with biological neural networks comparable to the brain of living beings. These systems "learn" to solve the problem in accordance with the examples provided, but without applying any rules to specific problems. For example, they can learn to recognize a rabbit by analyzing a range of images, without any further detailed information such as the facts that they have fur, tails, whiskers, and a face. Automatically processing and generating the characteristics from the learning material, they manually record it as "rabbit" or "no rabbit." [6, 7].

2.1.2. Perceptron

Perceptron (Latin: *perceptio*: German: *perzeptron*) is a computer model for brain-capture (i.e., a cybernetic model of the human brain) proposed by American psychologist Frank Rosenblatt in 1957. It was first realized as an electronic machine in 1960: the Mark 1 Perceptron was one of the first models of the peripheral neural network and the first neurocomputer [8].

Perceptron consists of three types of elements and transmits the signals coming from the regulators to the associative elements and then to subsequent reactionary elements; thus, perceptron enables the generation of a set of "associations" from the necessary reactions of the output to the input stimulus. In modern terminology, perceptron can be classified as an artificial neural network owing to:

1. the hidden layer;
2. threshold transmission functions;
3. the direct spread of the signal.

2.1.3. Support Vector Machines

Support vector machines are supervised learning models with a set of learning algorithms for classification and regression analysis (a statistical method for studying the effects of one or more independent variables, or regressors, on the dependent variable, or criterion) [9, 10]. Classification refers to linear classifiers, and can also be seen as a special case of Tikhonov regularization. The main feature of the support vector machine method is minimizing the empirical classification error at the same time as maximizing the margin (prediction interval); therefore, it is also known as the maximal margin classifier (interval adjustment method). The equations and formulas are empirical (Greek: *empeiria*): they are based on experience rather than theory.

2.1.4. Fuzzy Logic

Fuzzy Logic is a type of polysemantic logic: the correct value of the variables can be any real value in the range [0,1]. It is partly used to process the concept of reality in which the correct value can be absolutely true and totally incorrect [0,1]. In contrast, the actual value of the variables in Bull logic can be either 0 or 1, being only integers [11-13].

The term "fuzzy logic" was coined by Lotfi Zadeh when he introduced his fuzzy set theory in 1965; however, such logic had been studied since the 1920s, especially by Łukasievic and Tarski. It is applied in a range of areas, from artificial intelligence to management theory.

The concept of fuzzy sets is a generalization of the concept of sets based on elements estimation and in 1965, was proven by Lofti Zadeh to be a natural generalization of fuzzy logic. Hence, an object can either be or not be an element of a specific set or be a partial element of a fuzzy set.

2.2. Evolutionary Computation

In computer science, evolutionary computation refers to algorithms required for global optimization, based on biological evolution, and the artificial intelligence and soft computing that analyze those algorithms. It is a trial-and-error method for solving technical metaheuristic or stochastic (probability) optimization problems [14].

Evolutionary computation generates the first set of possible solutions and then produces iterative updates by the stochastic deletion of less desirable solutions and introduction of random minor changes. In a biological sense, each population of solutions is exposed to either natural or artificial selection and mutation [15].

These methods can be used to produce highly optimized solutions for a broad spectrum of problems, which may comprise numerous options for data structures. Evolutionary computation is sometimes used as an experimental procedure in evolutionary biology as well, to study the common aspects of general evolutionary processes.

2.2.1. Evolutionary Algorithms

Evolutionary algorithms (EA) are a subset of evolutionary computation in the field of artificial intelligence. As biological evolutionary mechanisms, such as reproduction, mutation, recombination, selection, are applied and reapplied, the population of solutions evolve [16].

EA is ideal for approximating the solutions for all types of problems, as no assumptions are made about the constituents. When applied to biological evolution modeling, these algorithms are often restricted to microevolution studies and planning models based on cell processes. In many real-world situations, the complexity of computation, due to assessment by the fitness function, hinders development; however, fitness approximation is one solution to this difficulty. It thus seems that simple EA can resolve complex problems quite quickly [17].

2.2.2. Genetic Algorithms

Genetic algorithms (GA) in the field of computer science and operations research are metaheuristic, arising from natural selection As a class, the largest of the evolutionary algorithms, they are often used to generate high-quality solutions to optimization and search problems [18, 19].

2.2.3. Differential Evolution

Differential evolution (DE) is a method in evolutionary computations that optimizes problems by finding an approximate solution and improving it in accordance with specific quality indicators. Such a method is metaheuristic: no assumptions are made about the issues being optimized and the largest areas of possible solutions are probed; however, there is no guarantee that an optimal solution will ever be found.

DE is used with multidimensional functions, but does not use the gradient of the optimized problem, as in classic optimization methods [20].

2.2.4. Metaheuristic and Swarm Intelligence

Metaheuristic is a procedure to find, generate, or select a heuristic (partial search algorithm) in the field of computer science and mathematical optimization [21, 22]. Metaheuristics select solutions from a set that is too large to be completely sampled. Several solutions may be available for the optimized problem being solved, and so metaheuristics can be used to resolve various issues [23].

However, compared with optimization algorithms and iterative methods, metaheuristics is no guarantee that a global optimal solution can be found for some classes of issues [23, 24].

Swarm intelligence refers to the collective behavior of decentralized, self-organized natural or artificial systems. The term was introduced in 1989 by Gerardo Beni and Jing Wang, in the context of automated cell systems [25].

2.2.5. Ant Colony Optimization

Soft computing technology focuses on resolving management issues, using a set of tools that include fuzzy neural networks, genetic algorithms, and evolution modeling (e.g., immune algorithms and intelligent algorithms based on the behavioral reactions of animal groups, birds, ants, and bees). Different soft computing methods complement each other and are often used in combination [26, 27].

Ant colony optimization (ACO) is an effective polynomial algorithm for finding solutions to both the "traveling salesman problem" (TSP) and problems with analogical path searches in graph theory. The essence of this approach is its analysis and use of an ant behavioral model, which searches for paths between the colony and food source and represents metaheuristic optimization. The first version, aimed at finding the optimal path in a graph, was proposed by Dr. Marko Dorigo in 1992.

2.2.6. Particle Swarm Optimization

Particle Swarm Optimization (PSO) does not require the exact gradient of the optimized problem to be known.

Kennedy, Eberhart, and Shi demonstrated that PSO imitated social behavior, while its later simplification was found to perform optimization [28, 29]. Kennedy [30] described many philosophical aspects of PSO and swarm

intelligence, and the former then continued to be extensively studied by Eberhart, et al. [31]. PSO first optimizes the problem, then takes possible solutions, called "particles," and moves them around the solutions area according to a simple formula.

2.2.7. Expert Systems

An expert system is a computer system that emulates the decision-making of human experts and is designed to resolve complex issues by using a knowledge base and if–then rules [32]. These systems were first developed in the 1970s and became widespread in the 1980s [33]. Although being particularly successful in the field of artificial intelligence, some specialists note that expert systems are not part of true artificial intelligence [33-35].

2.3. Ideas about Probability

To use probability models properly to determine mathematical thinking, they need to be updated. Their components are derived from standard mathematical models and decisions; however, it is necessary to identify in which relationships, sometimes by their context, probability thinking can be distinguished. In some cases, though, the criteria may not be at all probable [36].

2.3.1. Bayesian Network

A Bayesian network—also known as a decision network or Bayesian model—is a probabilistic graphical model in which a graph represents the conditional dependencies between random variables. For example, a Bayesian network may represent the possible relationship between diseases and symptoms, so that given the symptoms, the probability of suffering a range of diseases may be estimated [37].

Using the abovementioned soft computing components, scientists resolve various intellectual issues. Some examples of soft computing software used to find solutions are provided in the next section.

3. MACHINE LEARNING SOFTWARE

Khomh, et al. [38] studied industrial software systems based on machine learning models, reviewing the testing and application of those systems.

Organizations try to secure the expected functionality and quality of such systems within the intended time frame and budget. Despite numerous advanced techniques to evaluate these efforts, performing the evaluations and re-evaluations lead to project deficits and significant losses for the organizations.

Consequently, Rijwani and Jain [39] offered a machine learning-based approach to calculate the optimum effort and level of confidence. For given variables, a genetically trained neural network evaluated the optimum effort required, while the confidence level was determined through fuzzy logic to indicate the point at which the predicted effort would not exceed the limit.

Appelt, et al. [40] focused on the Microsoft Windows firewall as an important safeguard in online software system maintenance. Due to the constant creation of new types of external attack and their increasing complexity, firewalls should be regularly updated and tested to prevent malevolent access. This study reviewed testing against SQL injection attacks and proposed adaptions to the general principles and strategies for other contexts. The result was ML-Driven, based on machine learning and an evolutionary algorithm that first detected those SQL injection attacks bypassing the firewall and then identified and learned their patterns to prevent further access.

The software development life cycle (SDLC) comprises a range of activities that result in a software product: specification, design, implementation, testing, modification, and maintenance. In addition, other supporting activities are employed, such as configuration and change management, quality assurance, project management, and user experience evaluation. The infrastructure that supports all these activities is the software warehouse, which comprises several systems, including management, error tracking, code change, code checking, setup, binary files,

wikis, and forums. Guemes-Pena, et al. [41] provided an overview of research and discussion on intelligent database software over the decade up to 2018 and identified future challenges in upgrading machine learning strategies, existing business topics, and organizational decision-making systems.

Systems analysts often use software fault prediction models in the design stage of SDLC to identify modules vulnerable to failures and predict those that may be fault-prone based on their size. Some researchers [42] have studied the effectiveness of models employing machine learning methods and demonstrated that cost–benefit prediction models perform best at the low (47.28%), median (39.24%), and high (25.72%) thresholds. They also reviewed the nine feature selection methods that proved the best in removing redundant metrics and predicting faults.

Web applications are crucial to the software industry and are constantly evolving to satisfy new criteria and functions. Although quality is assured through testing, defects prevent further development. As defects, which can be caused by a number of factors, are extremely expensive to minimize, it is important to detect any faults at an early stage of development by applying a prediction model to identify potential failures in web applications. Malhotra and Sharma [43] thus compared 14 machine learning methods to reveal the relationship between object-oriented metrics and error predictions within these applications, using different editions of Apache Click and Apache Rave data sets.

Since the late 20th century, considerable research has been undertaken on the application of machine learning algorithms to overcome the deficiencies in traditional and parametric estimation methods, increase the success of software projects, and conform to modern project development and management practices. However, due to inconclusive results and ill-defined models, there has been no implementation. One study [44] thus offered practical, effective approaches to narrow this theory/practice divide through the latest research findings and organizational best practice.

One component of machine learning is neural networks [45] for which Figure 2 [46] shows the best 20 software products, and Figure 3 the ratings of those in most common use in 2018.

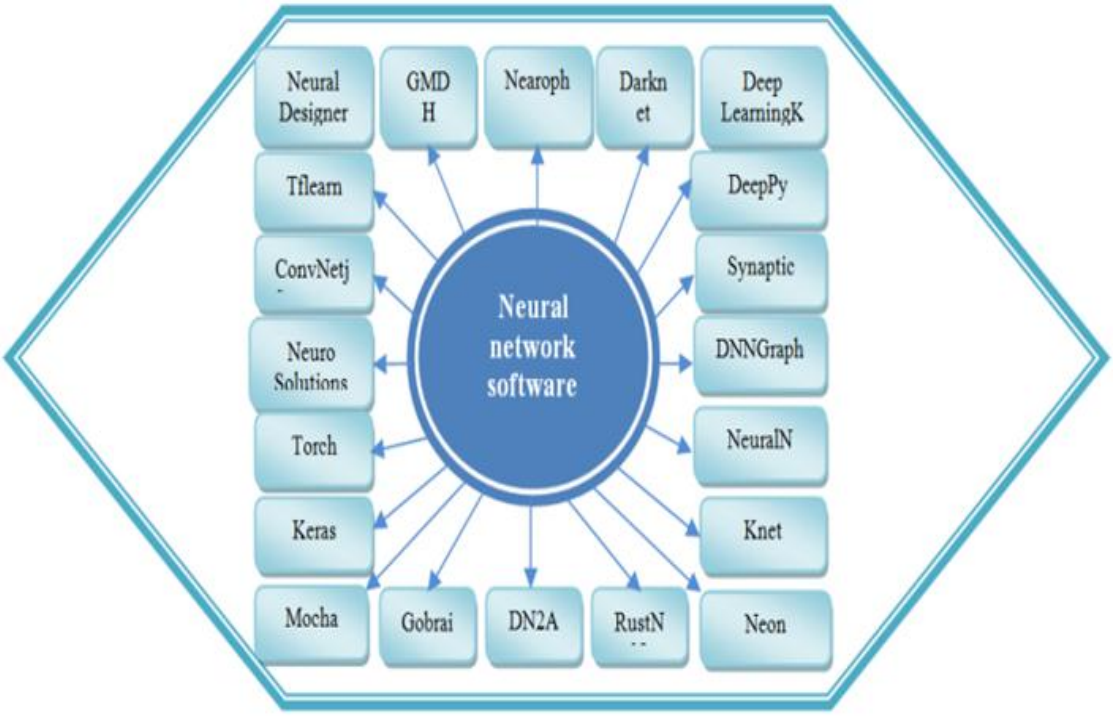

Figure-2. The best 20 neural network software products.

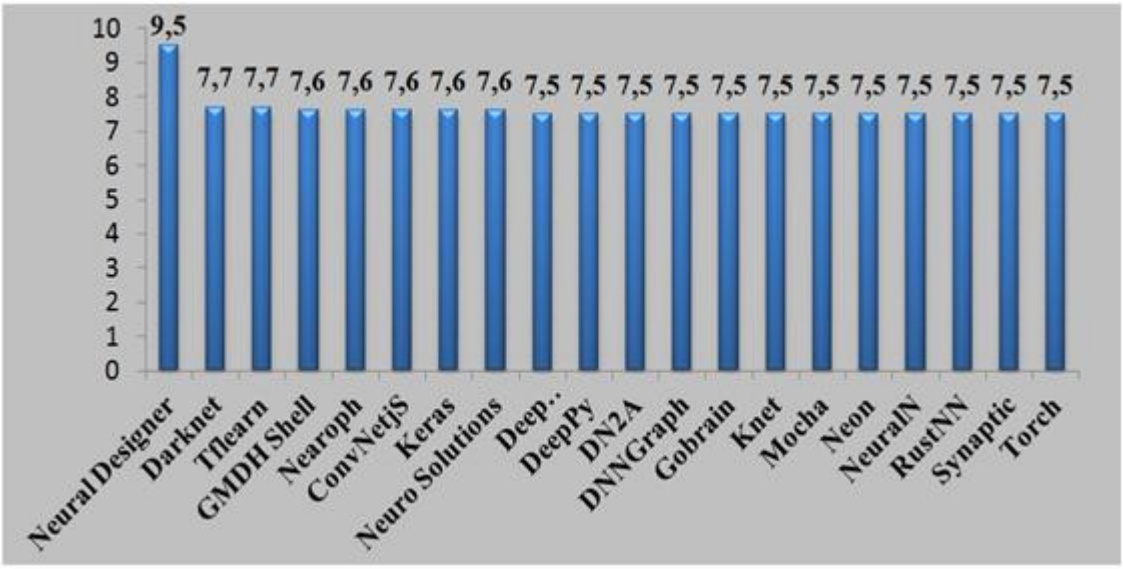

Figure-3. Ratings of neural network software in most common use in 2018.

Figure 4 refers to perceptron software technologies [47].

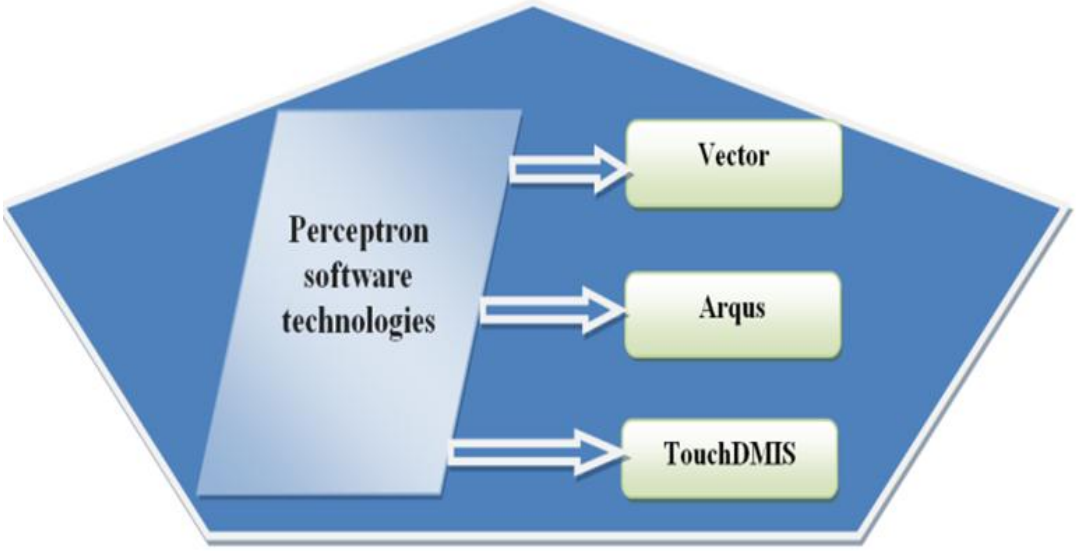

Figure-4. Perceptron software technologies.

1. Vector is the basis of perceptron's measurement techniques, providing calculation and visualization tools to convert sensor data into actionable information. The user interface module is designed to meet the specific needs of each user profile, from production to project management, enabling users to access important data quickly and directly through interlinking modules.
2. Argus is an advanced analysis program that advances perceptron's measurement tools beyond traditional historical reporting. Argus Detector runs in the background constantly analyzing data for suspicious patterns. When an upgrade becomes possible, Argus Identifier notifies the user about potential "case studies."
3. TouchDMIS is a 100% touch user interface (TUI) for the TouchCAD fast programming module, and is innovative in being the first TUI computerized maintenance management (CMM) software. The interface integrates seamlessly with both manual and DCC COORD3 touch and scanning probes, requiring just a few hours' training and offering extremely short learning curves.

The main feature of the support vector machine method is minimizing the empirical classification error at the same time as maximizing the margin (prediction interval); therefore, the method is also known as the maximal margin classifier (interval adjustment method). Its main purpose is to transfer vectors to a high-dimensional feature space, in which a hyperplane is created to divide the positive and negative vectors with the maximal margin in-between the nearest positive and negative (i.e., support) vectors to that hyperplane. Thus, the algorithm works on the assumption that the larger the interval between the support vectors, the smaller the average error of the classifier [47].

Determining a reliable method of forecasting failures is a prerequisite for building an effective fault prediction model, which depends on the characteristics of the fault data sets collected from earlier or similar software. Rathore and Kuma [48] therefore compared and evaluated fault prediction methods, and then recommended a system with which to select the most appropriate [48].

IT project managers are responsible for the schedule, evaluation, utilization, and monitoring of a project throughout its life cycle. However, as selecting the methodology is heuristic and the efficiency of the system under development is unpredictable, a projected productivity model (PPM) can help predict the factors required to achieve the targets set for the process. This, in turn, can help organizations and projects control those factors that will ensure the expected results are achieved. For a well-defined project, a PPM can determine the relationship between different variables, enabling performance to be predicted and solutions to problems implemented. This approach, however, has not been widely applied to real-world projects, but was implemented by an Indian IT company. Sharma, et al. investigated different mathematical models based on the IT sector's data and built PPMs based on Bayesian and fuzzy logic, presenting two such models as confirmation. It is expected that building PPMs will become a necessity for high maturity IT organizations in future [49].

There is a rising demand for reliable critical software systems; it is considered a key quality factor. Reliability depends on various factors, so it is important to evaluate it actively and effectively. Reliability models should be based on the important factors and are best evaluated using soft computing methods, such as fuzzy logic. Fuzzy logic can identify a specific solution for imprecise multiple factors by converting vague information into the best evaluation model. As the selection of a critical software system can be challenging, Dubey, et al. took four factors into account to evaluate reliability, the results of which they verified using defuzzification. They thus proposed a model that was effective for applications [50].

Given the increase in software cost optimization, processing the uncertainty in software cost estimation is now vital for modern organizations. Kaushik, et al. [51] introduced a new technique for software cost optimization, CUCKOO_FIS, which combines two optimization methods: CUCKOO, a swarm intelligence-based metaheuristic algorithm, and Fuzzy Inference System (FIS), based on fuzzy logic mathematics. As software cost estimation is an "approximation" of not only the cost but also the effort involved in a software project, this new technique was applied to the software cost estimation model for effort optimization and was successfully evaluated using tera-PROMISE data sets. Using swarm intelligence and fuzzy sets that work with non-algorithmic methods, this software cost estimation model proved more accurate than earlier models.

Alostad, et al. [52] used fuzzy logic concepts to build a fuzzy-based model for improving effort estimation within the Scrum framework, where the team completes it work by dividing it into a series of sprints. Several factors significantly affect the effort estimation of each task in a sprint: the team's experience, the task's complexity, its size, and the estimation accuracy, which are often, presented using linguistic quantifiers. Figure 5 illustrates the three components within the proposed model, which was developed using MATLAB.

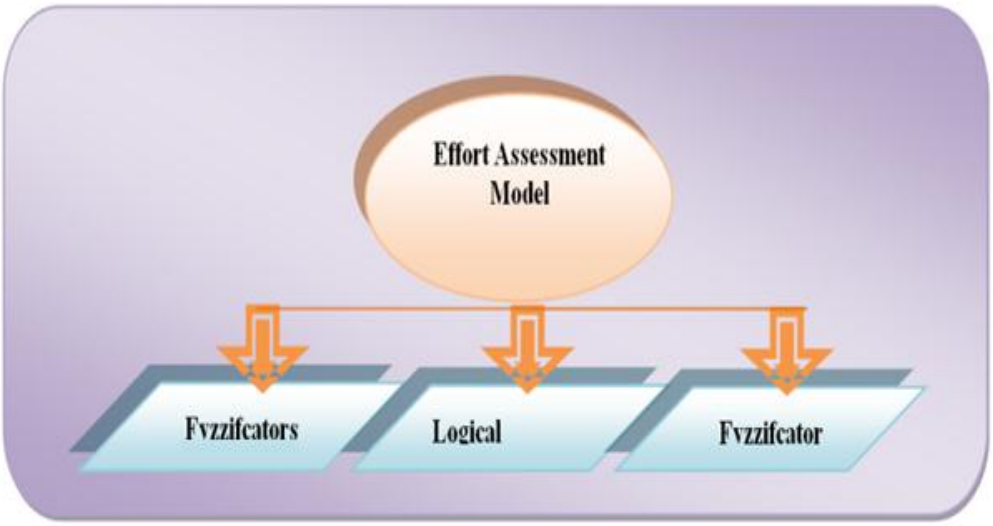

Figure-5. Effort estimation model for the Scrum framework.

The model also takes into account the feedback from a comparison between the estimated and actual efforts.

Risk identification and assessment are important activities in software project management but are complex processes, especially for new software organizations with few resources. Therefore, research undertaken by Vahidnia, et al. [53] proposed a method and prototype tool to assist software developers with risk assessment. First, they determined the risks associated with software projects, and then surveyed 86 practitioners in small organizations for their opinions, based on past projects, on the probability and impact of each risk factor. This "default" data was employed to develop the prototype tool, which, as it is used and collects additional data, will improve in accuracy for risk prediction and resolution. The methodology proved partially successful in predicting risks and estimating the probability of predetermined failures.

4. CONCLUSION

Today, computerization and data require high-quality software and tools. As such, substantial implementation of specific technologies is needed for quality assurance, reliability, speed, conformity to certain capabilities, and integrity of documentation, expansion capabilities, development, and so forth.

Therefore, this paper examined the methods for increasing software efficiency 25(effectiveness, productivity, flexibility, etc.) based on soft computing technologies and provided a broad overview of some of those methods. Consequently, it is highly probable that these technologies will expedite the use of software products and significantly reduce their cost. Moreover, it is believed that soft computing technologies will be further improved and more widely adopted in future.

Funding: This study received no specific financial support.
Competing Interests: The author declares that there are no conflicts of interests regarding the publication of this paper.

REFERENCES

[1] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Communications of The ACM*, vol. 37, pp. 77-85, 1994.

[2] X. S. Yang, Z. H. Cui, R. Xiao, A. Gandomi, and M. Karamanoglu, *Swarm intelligence and bio-inspired computation*. Amsterdam: Elsevier, 2013.

[3] D. K. Chaturvedi, *Soft Computing: Techniques and its applications in electrical engineering*. Berlin: Springer, 2008.

[4] C. M. Bishop, *Pattern recognition and machine learning*. Berlin: Springer, 2006.

[5] S. Arthur, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, 1959.

[6] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of The National Academy of Sciences*, vol. 79, pp. 2554-2558, 1982. Available at: https://doi.org/10.1073/pnas.79.8.2554.

[7] G. Felix and S. Jürgen, "LSTM recurrent networks learn simple context free and context sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, pp. 1333–1340, 2001. Available at: 10.1109/72.963769.

[8] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, pp. 277-296, 1999.

[9] C. Corinna and V. Vladimir, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995. Available at: 10.1007/BF00994018.

[10] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125-137, 2001.

[11] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical principles of fuzzy logic*. Dordrecht: Kluwer Academic, 1999.

[12] B. C. Arabacioglu, "Using fuzzy inference system for architectural space analysis," *Applied Soft Computing*, vol. 10, pp. 926-93, 2010. Available at: 10.1016/j.asoc.2009.10.011.

[13] G. Acampora, V. Loia, C. Lee, and W. Mei-Hui, *On the power of fuzzy markup language, studies in fuzziness and soft computing* Berlin Heidelberg: Springer, 2013.

[14] T. Bäck, B. F. David, and M. Zbigniew, *Handbook of evolutionary computation*. Bristol: UK: Ltd, 1997.

[15] M. Sipper, W. Fu, K. Ahuja, and J. H. Moore, "Investigating the parameter space of evolutionary algorithms," *BioData Mining*, vol. 11, pp. 1-14, 2018.

[16] P. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Proceedings of Global Trends in Signal Processing Proceedings International Conference Information Computing and Communication (ICGTSPICC), Jalgaon*, 2016.

[17] J. Cohoon, J. Kairo, and J. Lienig, *Advances in evolutionary computing*. Berlin: Springer, 2003.

[18] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65-85, 1994. Available at: https://doi.org/10.1007/bf00175354.

[19] R. Akbari and K. Ziarati, "A multilevel evolutionary algorithm for optimizing numerical functions," *International Journal of Industrial Engineering Computations*, vol. 2, pp. 419-430, 2011. Available at: https://doi.org/10.5267/j.ijiec.2010.03.002.

[20] P. Rocca, G. Oliveri, and A. Massa, "Differential evolution as applied to electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 53, pp. 38-49, 2011. Available at: https://doi.org/10.1109/map.2011.5773566.

[21] R. Balamurugan, A. Natarajan, and K. Premalatha, "Black hole optimization for biclustering microarray gene expression data," *Applied Artificial Intelligence an International Journal*, vol. 29, pp. 353–381, 2015. Available at: https://doi.org/10.1080/08839514.2015.1016391.

[22] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing*, vol. 8, pp. 239-287, 2009. Available at: https://doi.org/10.1007/s11047-008-9098-4.

[23] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, pp. 268-308, 2003. Available at: https://doi.org/10.1145/937503.937505.

[24] K. Sörensen, "Metaheuristics—the metaphor exposed," *International Transactions in Operational Research* vol. 22, pp. 3-18, 2015. Available at: https://doi.org/10.1111/itor.12001.

[25] N. Monmarché, F. Guinand, and P. Siarry, *Artificial ants*. London: Wiley-ISTE, 2010.

[26] A. Y. Zhang and W. Shuihua, "Rule-based model for bankruptcy prediction based on an improved genetic ant colony algorithm genlin," *Mathematical Problems in Engineering*, pp. 1-10, 2013. Available at: https://doi.org/10.1155/2013/753251.

[27] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems, robots and biological systems," in *NATO Advanced Workshop, Tuscany, June 26–30: Proceedings. – Italy*, 1993, pp. 703-712.

[28] M. H. Golbon-Haghighi, H. Saeidi-Manesh, G. Zhang, and Y. Zhang, "Pattern synthesis for the cylindrical polarimetric phased array radar (CPPAR)," *Progress in Electromagnetics Research*, vol. 66, pp. 87-98, 2018.

[29] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.

[30] J. Kennedy, "The particle swarm: Social adaptation of knowledge, evolutionary computation," in *Proceedings of IEEE International Conference*, 1997, pp. 303–308.

[31] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm intelligence*: Waltem: Morgan Kaufmann, 2001.

[32] P. Jackson, *Introduction to expert systems*, *ACM digital library*. Boston: Addison Wesley Longman Publishing Co, 1998.

[33] P. Jackson, *Introduction to expert systems*. Melbourne: Addison Wesley, 1998.

[34] C. T. Leondes, *Cornelius expert systems: The technology of knowledge management and decision making for the 21st century*. New York: Academic Press, 2002.

[35] S. Russell and P. Norvig, *Artificial intelligence: A modern approach*. Milan: Pearson, 2010.

[36] A. Stuart and K. Ord, *Kendall's advanced theory of statistics, distribution theory*. USA: Wiley, 2010.

[37] G. I. Ben, *Bayesian networks, encyclopedia of statistics in quality and reliability*. USA: John Wiley &Sons, 2008.

[38] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, and G. Antoniol, "Software engineering for machine-learning applications: The road ahead," *IEEE Software*, vol. 35, pp. 81-84, 2018.

[39] P. Rijwani and S. Jain, "Level of confidence in software effort estimation by an intelligent fuzzy - Neuro - genetic approach," *International Journal of Advanced Computer Science and Applications*, vol. 9, pp. 568-576, 2018.

[40] D. Appelt, C. D. Nguyen, A. Panichella, and L. C. Briand, "A machine-learning-driven evolutionary approach for testing web application firewalls," *IEEE Transactions on Reliability*, vol. 67, pp. 733-757, 2018. Available at: https://doi.org/10.1109/tr.2018.2805763.

[41] D. Guemes-Pena, C. Lopez-Nozal, R. Marticorena-Sanchez, and J. Maudes-Raedo, "Progress in artificial intelligence," vol. 7, pp. 237-247, 2018.

[42] L. Kumar, A. Tirkey, and S. K. Rath, "An effective fault prediction model developed using an extreme learning machine with various kernel methods," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 864-888, 2018. Available at: https://doi.org/10.1631/fitee.1601501.

[43] R. Malhotra and A. Sharma, "Analyzing machine learning techniques for fault prediction using web applications," *Journal of Information Processing Systems*, vol. 14, pp. 751-770, 2018.

[44] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *Journal of Systems and Software*, vol. 137, pp. 184-196, 2018. Available at: https://doi.org/10.1016/j.jss.2017.11.066.

[45] M. Van Gerven and S. Bohte, *Artificial neural networks as models of neural information processing*. Lausanne: Frontiers in Computational Neuroscience, 2018.

[46] Top 30 Artificial Neural Network Software, Retrieved from https://www.predictiveanalyticstoday.com/top-artificial-neural-network-software/, 2019.

[47] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[48] S. S. Rathore and S. Kuma, "A decision tree logic based recommendation system to select software fault prediction techniques," *Expert Systems With Applications*, vol. 71, pp. 342-357, 2017.

[49] B. Sharma, R. Nag, and M. Makkad, "Process performance models in software engineering: A mathematical solution approach to problem using industry data," *Wireless Personal Communications*, vol. 97, pp. 5367-5384, 2017. Available at: https://doi.org/10.1007/s11277-017-4783-1.

[50] S. K. Dubey, S. Singh, and H. Chaudhary, "Evaluation of reliability of critical software system using fuzzy approach," *International Journal of System Assurance Engineering and Management*, vol. 8, pp. 1327-1335, 2017. Available at: https://doi.org/10.1007/s13198-017-0603-y.

[51] A. Kaushik, S. Verma, H. J. Singh, and G. Chhabra, "Software cost optimization integrating fuzzy system and COA-Cuckoo optimization algorithm," *International Journal of System Assurance Engineering and Management*, vol. 8, pp. 1461-1471, 2017. Available at: https://doi.org/10.1007/s13198-017-0615-7.

[52] J. M. Alostad, L. R. Abdullah, and L. S. Aali, "A fuzzy based model for effort estimation in scrum projects," *International Journal of Advanced Computer Science and Applications*, vol. 8, pp. 270-277, 2017. Available at: https://doi.org/10.14569/ijacsa.2017.080939.

[53] S. Vahidnia, Ö. Ö. Tanrıöver, and I. Askerzade, "An Early Phase Software Project Risk Assessment Support Method for Emergent Software Organizations," *International Journal of Advanced Computer Science and Application*, vol. 8, pp. 105-118, 2017. Available at: https://doi.org/10.14569/ijacsa.2017.080514.