Hindawi
Scientific Programming
Volume 2021, Article ID 5532197, 14 pages
https://doi.org/10.1155/2021/5532197

Research Article

Development of Rule-Based Software Risk Assessment and Management Method with Fuzzy Inference System

Mustafa Batar,[1] **Kökten Ulaş Birant**,[2] **and Ali Hakan Işık**[3]

[1]*Graduate School of Natural and Applied Sciences, Dokuz Eylül University, 35390 İzmir, Turkey*
[2]*Department of Computer Engineering, Dokuz Eylül University, 35390 İzmir, Turkey*
[3]*Department of Computer Engineering, Burdur Mehmet Akif Ersoy University, 15030 Burdur, Turkey*

Correspondence should be addressed to Mustafa Batar; mbatar@cs.deu.edu.tr

Received 20 February 2021; Accepted 9 May 2021; Published 24 May 2021

Academic Editor: Sikandar Ali

There is an enormous budget and financial plan in software development projects, and it is required that they take a huge investment to carry on. When looked at, the costs depend on the global substantial information about software development: in 1985, $150 billion; in 2010, $2 trillion; in 2015, $5 trillion; and in 2020, over $7 trillion. Additionally, on the first new days of 2021, a day-by-day Apple Store's quantity has been approximately $500 million. In spite of the expenditures and the margins that are dramatically expanding and increasing each year, the phase of software development accomplishment is not high enough. In light of the "CHAOS" report arranged in 2015, just 17% of the software projects were finished in an opportune way, in the allotted financial plan, and as per the necessities. However, 53% of the software projects were finished in the long run or potentially over a spending plan as well as without satisfying the prerequisites precisely. In addition, software development projects were not completed and were dropped out as well in the ratio of 30%. Also, the "CHAOS" report published in 2020 has figured out that only 33% of the software projects were completed successfully all over the world. In order to cope with these unsuccessful and failure results, an effective method for software risk assessment and management has to be specified, designated, and applied. In this way, before causing trouble that has the power of preventing successful accomplishment of software development projects, software risks are able to be noticed and distinguished on time. In this study, a new and original rule set, which could be used and carried out effectively in software risk assessment and management, has been designed and developed based on the implementation of fuzzy approached technique integrated with machine learning algorithm—Adaptive Neuro-Fuzzy Inference System (ANFIS). By this approach and technique, machines (computers) are able to create several software risk rules not to be seen, not to be recognized, and not to be told by human beings. In addition, this fuzzy inference approach aims to decrease risks in the software development process in order to increase the success rate of the software projects. Also, the experimental results of this approach show that rule-based software risk assessment and management method has a valid and accurate model with a high accuracy rate and low average testing error.

1. Introduction

Risk is the likelihood of not reaching a targeted result, and also it is the probability of any event that would prevent an organization from achieving its strategic, financial, and operational objectives. In addition, risk has two main factors: the possibility of occurrence—the likelihood of not achieving a particular result or the likelihood of an undesired occurrence—and size of loss—the effects of the consequences that would arise if the risks were realized [1].

Risks come on the whole shapes and sizes; hazard experts, by and large, perceive three significant sorts: market risk is the danger that costs will move in a manner that has antagonistic outcomes of an organization; credit risk is the danger that a client, a counterparty, or a provider will neglect to meet its commitments; and operational risk is the danger that individuals, cycles, or frameworks will fall flat or that an outer occasion will contrarily affect the organization [2].

Software development projects suffer from and also are open to many risks, especially market risks, financial (credit)

risks, and technical (operational) risks. Software developers have to provide positive and especially satisfactory answers to several significant questions in order to achieve success—successful accomplishment: if the developed software fulfils the demands and the requirements of the customers or not and provides the necessary solutions to them or not; how much competition the developed software can face and cope with; whether the benefits and the gains derived from the software project exceed the development cost or not; if the project to be developed can be implemented technically and technologically or not; whether the hardware, the software, the necessary network connections, and the entire substructure will work as planned properly at the project to be developed or not; if the existing technology can meet the objectives and outputs of the software project or not; is there a possibility that the technology of the current technology becomes obsolete and old before starting to use the software to be developed; whether the security and protection system will work actively during the software project life cycle or not; and so on. The problems mentioned here have arisen and have been identified from the samples of failed software projects in the literature, and they have shown that the risks cannot be managed effectively [3].

In this context, an operating-state and reliable risk assessment and management method has been designed and developed for software projects based on the concrete, tangible, and original software risks rule set in the light of literature, fuzzy structure, and machine learning.

The proposed algorithm—ANFIS (Adaptive Neuro-Fuzzy Inference System)—for generating new original software risk rules is a methodology that consolidates the remarkable abilities of all fake insight speculations by fuzzy logic (FL), artificial neural networks (ANN), and master frameworks. The essential attribute of the ANN approach is the capacity to learn. One of the significant inconveniences of this element is that the learning results can be given in extremely huge boundary sets. Hence, it is difficult to communicate the outcomes by words. The premise of the FL is that it is extremely near the way individuals thinking as in regular dialects. However, it cannot gain proficiency with the standards itself; it is reliant on the master framework or the perspectives of individuals. So, the significance of ANFIS arises in software risk assessment and management in order to create and develop valid and meaningful risk rules [4].

There are four fundamental motivations to apply, actualize, and determine "software risk assessment and management" in the software development process as indicated by Boehm [5, 6]: to stay away from overwhelms in arranging and in a financial plan, to guarantee that the software projects run impeccably, and also to ensure that software companies are able to create their products in the direction of their necessities; to forestall duplication of inner or outer software structure or coding which is caused by inadequate or muddled necessities that are about half of software projects' costs; not to do software risk assessment and analysis in the zones that have (practically) no danger; and to design and develop a product arrangement about software projects that the client needs so as to empower the vendors to get consumer loyalty and the ideal benefits.

Up to now, several software risk parameters have been determined in order to assess and manage software development projects: productivity, engagement, attention to quality, code based knowledge and management, adherence to coding guidelines and techniques, learning and skills, personal responsibility, etc. However, there is not any universally accepted methodology to apply software risk assessment and management. There are three main reasons for this situation: firstly, each part of software creation is unique. There is no compelling reason to assemble two times the same parts of software as it might be duplicated by copying it. This makes it truly difficult to make a formal and thorough correlation between two parts of the software. Secondly, the current technology is something that changes at a truly fast phase. So, each time a methodology in respect to a certain wave of technology is dependable enough, it is for the most part as recently old. Thirdly, there is a gigantic zone for innovativeness in discovering the diverse answers for a unique issue. Because of these reasons, the technique "fuzzy structure" [7, 8] has a very convenient and proper process for defining software risks due to their nature that has no certainty—uncertainty—structure and principle. Also, software risks are defined as the probability and the severity of damage that is caused by occurring of bad or undesirable events in a system. Thus, the system suffers from strategic, financial, operational, structural, or integrity loss and damage. So, there is a need to apply and carry out an efficient "software risk assessment and management" in order to determine and recognize software risks on time before causing problems and troubles into software projects for providing successful accomplishment in the software development process.

For this study, a real application has been designed and developed based on 36 software risk rules with 23 different software risk parameters available on hand. In addition, this rule set has been adjusted for the machine learning algorithm with integrated artificial intelligence mechanism of the fuzzy structure—Adaptive Neuro-Fuzzy Inference System (ANFIS). Moreover, the software risk rule set has been converted into numerical values for the ANFIS implementation, and it has been loaded into MATLAB. Afterward, based on the ANFIS algorithm, 32 new and original software risk rules have been created from the rule set on hand. Furthermore, both the rule set on hand and also the designed and developed original software risks rule set have been valid and accurate based on the low average testing error in ANFIS configuration on MATLAB. So, it means that the fuzzy inference system has provided trustworthy results in software risk assessment and management with a high accuracy rate.

This study is organized and structured like in the following: in Related Works, a literature review and past works about the studies to deal with risks, software risks, software risk assessment, and management have been given. Moreover, in Methods, the working principle and general information about the used methods and techniques applied and implemented in the experiments have been explained. In addition, the software risks rule set—the dataset—used in the study has been given in Software Risks and Rule Set.

Also, in Software Risks Rule Development with Fuzzy Structure and Machine Learning, the obtained results (the developed rules) have been given and the experimental studies (the numeric values) have been presented. And then, in Conclusions and Future Extensions, a brief summary, concluding remarks, and future extensions have been attached in the end.

2. Related Works

The administration of undertakings is definitely not a trifling assignment. A lot of variables, both quantitative and subjective, are important to ensure venture achievement or improve the task execution. Undoubtedly, perils basically plague a whole life example of an errand, and its most noteworthy wellsprings of information are the weaknesses. The expression "hazard" gets from the early Italian "risicare," which infers "to dare" [9]. As a science, the threat was considered during the Renaissance, in the sixteenth century, essentially building up the systems of likelihood hypothesis [10].

A few researchers, explicitly in software engineering, zeroed in about the observation that hazard measure is related to unfriendly factor measures [10] utilizing as boundaries the likelihood of misfortune and the results if there should be an occurrence of misfortune, where the item is called hazard presentation. Three angles related to software risks are given and put into words by Pfleeger et al. [11] and Boehm [5]: the misfortune related to the occasion, the likelihood of event of the occasion, and how much the outcomes of the occasion can be changed, being characterized as danger presentation.

Risk management is the information, instruments, and methods to lessen dangers to an adequate level while amplifying openings [12]. Furthermore, risk management in the software development process is the act of control and evaluation of dangers that affect tasks, cycles, and results of programming [10]. A significant highlight is thought of in software development is the correspondence, particularly of specialized dangers that are regularly known yet, ineffectively imparted [13]. As per Boehm [5] who is considered the father of software risk management, "risk administration is significant uncommonly in light of the fact that it causes individuals to maintain a strategic distance from calamities, adjust and undoing of undertakings and assists with animating a circumstance of accomplishment in programming projects."

The larger part of the investigations were announced in writings about software risk management center around the recognizable proof and examination of danger—i.e., the evaluation of dangers [6, 10, 14–19]. Notwithstanding, as per Bannerman [20], hardly any examinations center around the reception of software risk management since there is mindfulness with respect to the use of its practice, however ineffectively applied.

In the most recent years, a few works identified with the utilization of computerized reasoning strategies in hazard evaluation have shown up—dark relationship [21], probabilistic terms [22], fluffy hypothesis [23, 24], Bayesian networks [25], and case-based thinking [26]. The majority of these methods use hazard factors as a wellspring of data to foresee chances. The zone of software risk management even thought to be significant actually needs a few advances, both in research and practically speaking. Different examinations show the unequivocal act of danger the executives add to execution improvement and the achievement of activities they expand [27–34].

Managers generally deal with technical risks firstly about software development projects, but they push the risks into market, financial, and several other headings to the second plan. However, these risks are very crucial for successful software development, which is so important that it cannot be ignored. Because of this reason, there is a need for an integrated risk management that covers all steps in the software development process and also addresses, analyses, and evaluates all the risks that may arise in the project lifecycle [35].

In his study, Gallivan [36] has shown the relationship between the job and the professional profession about software risk assessment and management: satisfaction and difficulty, the actual (active) performance, technical knowledge of the profession, analytical thinking skills, verbal skills, work habits, new ideas and creativity to open and reveal various special points. In addition, Sawyer and Guinan [37] have shown several points to work as a software development team to determine and recognize software risks. These issues have been team support, team loyalty, team vision, team personalities, team meeting, team members, and team leader. Also, they have tried to find answers to some questions about software risk assessment and management. These questions are software development method, code retention, code library, and working time and are related to software development documentation. Moreover, Hall et al. [38] have tried to find answers to a few questions about a few issues about software development risks. These questions have been related to software team, software project, business life, work, and personality.

In applying software risk assessment and management, Baggelaar [39] has emphasized the importance of several points in his master thesis. These important points have been abstraction, testability, coupling, modularity, templates, test coverage, error handling, and exceptional case use. In addition, software developers have tried to find out the effect of code and comment line numbers in the software development process. Furthermore, Lee et al. [40] have analyzed and evaluated software risk assessment and management in terms of personality and work habits. Also, Ting [41] has been interested in and focused on the issues of personality, working style, workload and software development process in software risk assessment and management.

Zhang et al. [42] have asked a few questions for their work and received some answers about software risks. These questions have been number of lines of code, number of comment lines, number of classes, number of samples, class relation, number of methods, degree of heritability depth, and software development issues related to the difficulty. In addition, Calikli and Bener [43] have provided a general overview of software risk assessment and management. In

their work, they have shown the effect of software developers' level of education and some points and issues in the field of software development (satisfaction level, confidence level, work experience, etc.) on software risk assessment and management.

Chilton et al. [44] have put forward several points for software risks. These points have been work-life, working habits, personality, age, and gender. Moreover, Ramler et al. [45] have tried to research and find answers to some questions about software quality in software risk assessment and management. Also, Wang and Zhang [46] have highlighted a number of important issues in software risk assessment and management. These points have been work-life, work experience, workload, education level and gender. Furthermore, in their study, Balijepally et al. [47] have tried to find answers of many questions related to personality traits in recognizing software risks.

Duarte et al. [48] have tried to find out the effects of various issues in software risk assessment and management. These issues have been timing error, size error, segmentation error, missing parts, unrelated parts, number of errors, and the number of unit tests. Also, Ehrlich and Cataldo [49] have discussed some aspects of software development in order to determine software risks. These issues have been team leader, team coordination, company management, company employees, and private life. In addition, Kelly and Haddad [50] have tried to find out the extent to how "error" has an impact into software risk assessment and management. In addition, Schröter et al. [51] have tried to answer various questions in the minds about software development risks. These questions have been related to the number of constructs, code changes, method (method) number, fixed code parts, work-life, work quality, team leader, software project documents, and software development tool. Furthermore, Westermann [52] has emphasized the importance of certain points in the software development process. Reliable code writing has been investigated about the impact of software project outputs and work style on recognizing software risks. Moreover, Calikli and Bener [53] have shown some important points in software risk assessment and management. These points are software project development plan and software team psychology.

Also, there are 13 main articles from 1998 to 2017 about risk parameters in software risk assessment and management in the literature. These have been listed in Table 1.

3. Methods

In this section, applied methods and techniques in the study—fuzzy logic (FL), artificial neural networks (ANN), and Adaptive Neuro-Fuzzy Inference System (ANFIS)—have been explained and shown in detail.

3.1. Fuzzy Logic: Fuzzy Approach. Fuzzy logic is communicated as a methodology dependent on "levels of exactness" instead of the "valid or bogus" state which is the Boolean methodology. During the 1960s, Dr. Lotfi Zadeh applied the fuzzy logic mentality firstly in his classes in the University of California at Berkeley. Fluffy hypothesis can be utilized as a method for speaking to dubiousness in building nonlinear associations with heuristic data. The hypothesis essentially works with the rationale that rather than an articulation being 0 or 1, its worth may have an esteem that can differ in this range [65].

Fuzzy approach aims to display the overall working rationale of the PC such that individuals can comprehend inside the system of rationale. A PC's rationale block gets outright contribution from the client and gives the yields TRUE or FALSE, which is equal to YES or NO outcomes. As per the fluffy rationale approach, the client's choice expresses that there are various conceivable outcomes between YES and NO. Utilizing fluffy rationale strategy, it is meant to demonstrate unsure circumstances and inappropriately characterized or complex frameworks [66].

The engineering of fuzzy logic framework comprises three principle parts as shown in the accompanying figure. Initially, it changes over framework contributions to the fluffy sets gave in the fuzzification module. The standards area depicts the circumstances that decide the yields of the framework's fluffy rationale approach. These circumstances show which articulation should be yielded against the changing info articulations of the framework. At long last, the defuzzification module changes over the fluffy set produced by the surmising motor to net worth. Along these lines, the framework yields give diverse yield esteems as indicated by the guidelines in Figure 1 [67].

3.2. Artificial Neural Networks (ANN). An artificial neural network is essentially a framework that comprises various straightforward interconnected processors called neurons, each creating a succession of genuine esteemed initiations. Such counterfeit neurons were intended to copy the genuine organic neurons' method of activity, with the end goal that input neurons get initiated by means of sensors seeing the climate. For a fake neuron, such a climate could be an n-by-m picture framework or a network comprising of packed sound. Other transitional neurons get actuated by means of weighted associations from recently associated neurons through information neurons to assigned yield neurons [68].

Inspired from humans' brains, computations in ANNs are implemented in units, known as artificial neurons, distributed over the network in layers. The inputs of a certain neuron can be collected from the external domain or from the outputs of the previous layer's neurons. To calculate the output of a neuron, all collected inputs are weighted, by multiplying each of them with a certain value assigned per each input and summed, before being passed through a nonlinear function, known as activation function. This nonlinearity provides a more flexible output that has the ability to detect more complex features. Nevertheless, additional value can be added into the inputs of a neuron to provide bias to the computations. Furthermore, the mechanism of artificial neural networks has been demonstrated in Figure 2 [69].

3.3. Adaptive Neuro-Fuzzy Inference System (ANFIS). Adaptive Neuro-Fuzzy Inference System is uncovered incorporating the ANN's learning capacity and the simplicity

Table 1: The leading articles for software risk parameters in the literature.

Reference no.	Published year	Database	The article
[54]	1998	ACM	A framework for identifying software project risks
[32]	2004	ACM	Software project risks and their effect on outcomes
[3]	2007	Emerald insight	Managing risk in software development projects: a case study
[55]	2009	ACM	Analysis of systems development project risks: an integrative framework
[56]	2009	IEEE	The application of risk matrix to software project risk management
[57]	2009	IEEE	A software project risk analysis model based on evidential reasoning approach
[58]	2009	IEEE	An approach to facilitate software risk identification
[59]	2010	IEEE	A rule-based model for customized risk identification in distributed software development project
[60]	2011	ACM	Understanding IT project risks as disturbances to digital ecosystems
[61]	2012	IEEE	Comparing risks in individual software development and standard software implementation projects: a Delphi study
[62]	2013	IEEE	Risk management system for ERP software project
[63]	2014	IEEE	Top twenty risks in software projects: A content analysis and delphi study
[64]	2017	ACM	The application of fuzzy comprehensive evaluation method in the software project risk assessment

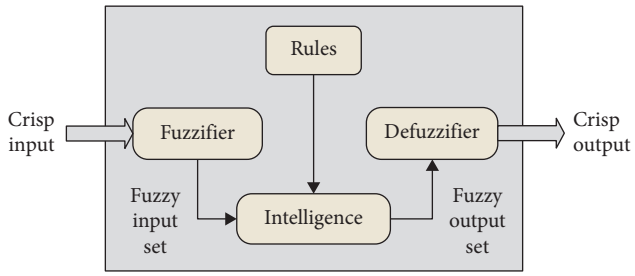

Figure 1: Fuzzy logic process.

with which the fuzzy approach can settle on choices in human capacity and give master information. While FL frameworks can give the learning and computation limit of ANN, ANN likewise gains aptitudes, for example, fuzzy control and dynamic and furnishing master information with this technique. Adaptive Network-Based Fuzzy Inference System is utilized with ANN and FL procedures to decide the qualities of the factors that will shape its own structure [70].

ANFIS works just Sugeno type models. It is as alteration of the Mamdani type demonstrating and the activities to be applied to the information are the equivalent. The main contrasts are in the yield information. In Sugeno type demonstrating, yield factors have enrollment capacities as a capacity of information sources and yield enrollment capacities must be direct or steady. ANFIS is a capacity document under the fuzzy logic toolbox in MATLAB. This program applied fluffy induction strategies to the information given to it and created suitable information models. Similarly as with other fuzzy deduction frameworks, enrollment works in ANFIS rely upon boundaries. In the event that the boundaries are changed, enrollment capacities will change and ANFIS will relegate self-enrollment capacities [71].

In the ANFIS network structure, which is framed by fuzzy standards, there are fuzzy enrollment capacities, fluffy duplication, standardization, assortment, and direct yield work layers in Sugeno type and the ANFIS structure has five-layer engineering. In ANFIS, which comprises five layers, the nerves in each layer contain similar tasks. Hubs with various shapes in layers contain capacities with various capacities. The hubs that appeared to fit as a fiddle are called versatile hubs and the boundaries of the hubs are set during the preparation of the organization. Hubs appearing as circles are fixed hubs [72].

Figure 3 shows the layers of Adaptive Neuro-Fuzzy Inference System architecture, and these layers have been explained in the following: blur layer, rule layer, normalization layer, purification layer, and total layer.

Blur layer (the layer 1): uses the generalized bell curve activation function as a membership function in dividing input values into fuzzy clusters in the blur layer.

Rule layer (the layer 2): every hub alludes to the number and rules of the Takagi-Sugeno fluffy deduction frameworks in this layer. The yield of each standard is the augmentation of enrollment degrees from the main layer.

Normalization layer (the layer 3): every hub in the haze layer concedes all hubs from 2 layers as info esteems and computes the normalized trigger level for each standard.

Purification layer (the layer 4): this layer, so-called purification layer, calculates the weighted result values of the rules formed according to the Takagi-Sugeno fuzzy inference method.

Total layer (the layer 5): it is the aggregation layer and has only one node. The output values of the nodes from the previous layer are collected to acquire the actual output value of the ANFIS system.

3.4. Software Risks and Rule Set. In order to research and determine software risk parameters, a literature review study

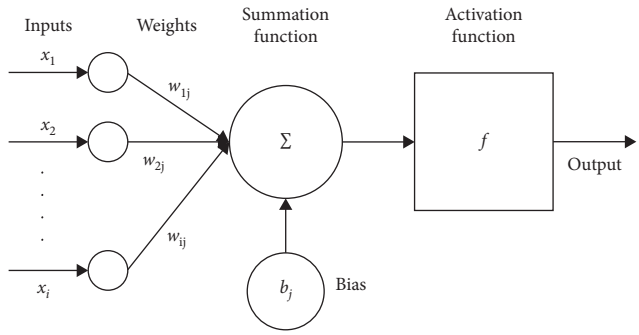

Figure 2: ANN working principle.

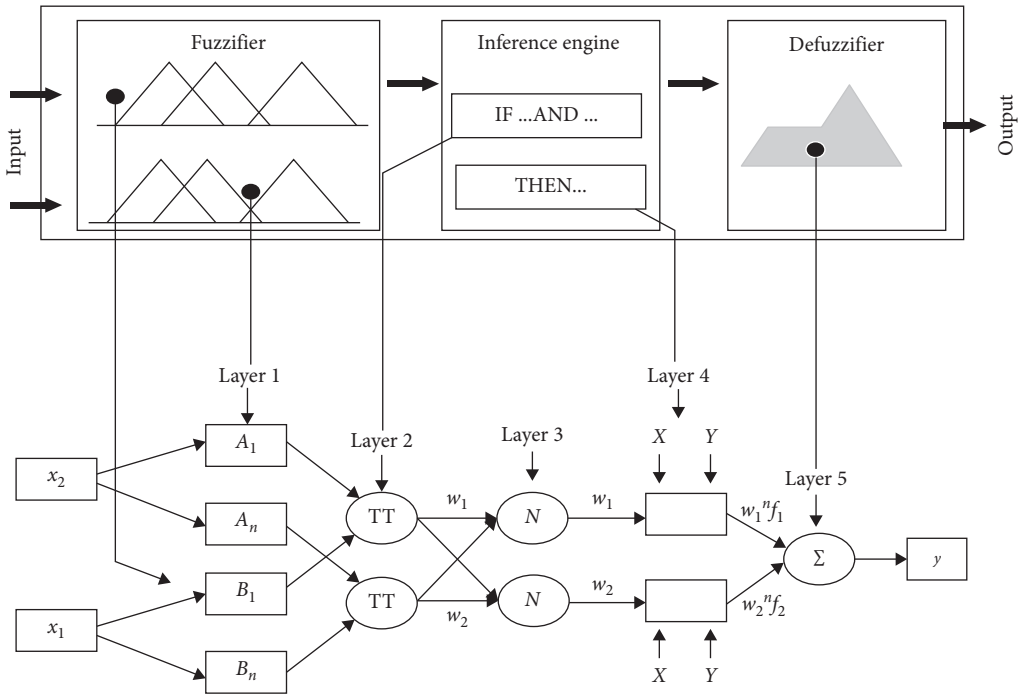

Figure 3: ANFIS architecture and its layers.

has been done. Also, this investigation has been carried out under three main contents: project risks, software project risks, and software project risks based on software developers' performance. In addition, for project risks, 50 science-indexed articles from 1978 to 2017 have been analyzed; for software project risks, 21 science-indexed articles from 1998 to 2017 have been examined; and for software project risks based on developers' performance, 17 science-indexed articles from 2001 to 2017 have been viewed and studied.

In the result of this wide literature review, for software risk parameters rule set, an IEEE article [59] published in 2010 has been chosen. According to the article [59], 19 meetings were led with specialists from 14 distinct organizations from Spain, India, and the USA in total. The specialists came from different spaces, for example, aviation, instructive programming, and custom programming advancement for the monetary business. Except for two interviewees who announced from an analyst point of view, each of them was the executive position—nine administrators—and the others were with several important positions, for example, CIO or director or supervisor. Also, each interview was reported that they have had long periods (up to twenty years) of involvement with circulated and worldwide software development projects all over the world. An assessment utilizing master interviews demonstrated that the dangers recognized by the model coordinated the genuine encounters in about 82% (trustworthy rate) of the cases. In this software risk parameters rule set, there have been 23 influencing factors and 36 different software risk parameters rules in total. These factors and rules have been shown in Table 2.

3.5. Software Risks Rule Development with Fuzzy Structure and Machine Learning. Lamersdorf et al. [59] have built and created a rule set—which consisted of 23 software risk parameters and 36 rules—as listed in Table 2. In addition, this software risk parameters rule set has been changed and converted into numerical values for design and development

Table 2: Software risks rule set table [39].

No.	Rules
1	Time zone difference ⟶ +Communication problems, lack of trust
2	Process maturity & time zone difference ⟶ −productivity drop
3	Coupling & time zone difference ⟶ +Communication problems
4	Time zone difference & (cultural difference | language difference) ⟶ +Coordination problems
5	Time zone difference & (cultural difference|language difference) & (phase = requirements) ⟶ +Quality problems, risk of project failures
6	!(Formality) & !(transparency) ⟶ +Risk of project failures
7	!(Formality) & (language difference | !(communication infrastructure)) ⟶ +Communication problems, productivity downfall
8	Language difference | cultural difference | !(personal relationships) | !(common experiences) ⟶ +Communication problems, lack of trust
9	Transparency ⟶ −Communication problems
10	!(Transparency) ⟶ +Lack of trust, quality problems
11	Size ⟶ +Travel cost overhead, −IP protection issues
12	Common experiences ⟶ −Productivity drop, coordination problems, lack of trust
13	Task coupling ⟶ +Productivity drop
14	!(Process knowledge) & size ⟶ +Communication problems
15	Language difference & cultural difference & !(common experiences) & !(personal relations) & !(process maturity) ⟶ +Risk of project failure
16	((Cultural differences & !(maturity)) | time pressure) & !(project experience) ⟶ +Communication problems
17	Maturity & common experiences ⟶ −Lack of trust
18	!(Requirements stability) & (!(communication infrastructure) | !(maturity)) ⟶ +Coordination problems
19	Process maturity ⟶ −coordination problems
20	Application knowledge ⟶ −Productivity drop
21	Technical knowledge | application knowledge | process knowledge | personal relations ⟶ −Quality problems
22	Communication infrastructure ⟶ −Productivity problems, cost overhead, quality problems, communication problems
23	!(Communication infrastructure) & (!(personal relations) | time zone difference) ⟶ +Communication problems
24	!(Communication infrastructure) & cultural difference ⟶ +Quality problem
25	Staff motivation ⟶ −Quality problems
26	!(Transparency) & time zone difference ⟶ +Productivity drop
27	!(Transparency) & !(personal relationships) ⟶ +Risk of project failures
28	Transparency ⟶ −Lack of trust
29	Coupling & number of sites ⟶ +Communication problems
30	Complexity | coupling ⟶ +Coordination problems
31	!(Cultural differences) & common experiences & communication infrastructure & process maturity ⟶ no problems
32	(Phase = coding) ⟶ −Project failure risk
33	(Phase = testing) & novelty of product & time zone difference & coupling ⟶ +Communication problems
34	Time pressure & !(personal relations) ⟶ +Communication problem
35	!(Requirements stability) & novelty of product & language difference & cultural difference ⟶ +Productivity downfall
36	Number of sites ⟶ +Coordination problem

based on a machine learning algorithm integrated with fuzzy structure—ANFIS (Adaptive Neuro-Fuzzy Inference System). Also, for the implementation of ANFIS, this numeric set has been loaded into MATLAB application for configuration. Furthermore, according to the ANFIS method, 32 new software risk rules have been implemented from the risks rule set on hand in Table 2.

In order to implement ANFIS—a machine learning approach with fuzzy artificial intelligence—on MATLAB, 36 software risk rules—at most 6 parameters in each rule in total (input & output)—on hand (Table 2) have been transformed into the values 0, 1, 2, and 3. The value 0 has meant that there is not any parameter available in that unit. In addition, the value 1 has been that the situation of the parameter is low. Moreover, the value 2 has shown that the case of the parameter is medium. Furthermore, the value 3 has determined that the state of the parameter is high. In the result of these values, the rules on hand have been arranged, adjusted, and normalized by transforming into the parameter values as shown in Table 3 (InputValues.txt in the Supplemental Files (available here)).

According to Figure 4, the membership function has been chosen for the input parameters of the rule set (Tables 2 and 3) as gauss2mf. This enrollment work is a smooth bend determined from two Gaussian participation capacities. Also, the number of memberships for the input has been given as the value 2—one unit less than the value of the state of the parameter "high" (value 3). Based on this value, the rule set has been applied into the ANFIS (Adaptive Neuro-Fuzzy Inference System) method in order to design and develop new and original software risk rules on MATLAB. Thus, 32 new rules (membership value: 2, input number: $6 - 1 = 5$, so output number: $2\hat{}5 = 32$ units) have been created from these input rules by the given inference operation and model in Figure 5.

Table 3: Numeric values of the software risks rule set in Table 2.

Rules	Values					
Rule 1	3	3	3	0	0	0
Rule 2	3	3	1	0	0	0
Rule 3	3	3	3	0	0	0
Rule 4	3	3	3	0	0	0
Rule 5	3	3	3	3	3	0
Rule 6	1	1	3	0	0	0
Rule 7	1	3	3	3	0	0
Rule 8	3	3	3	0	0	0
Rule 9	3	1	0	0	0	0
Rule 10	1	3	3	0	0	0
Rule 11	3	3	1	0	0	0
Rule 12	3	1	3	3	0	0
Rule 13	3	3	0	0	0	0
Rule 14	1	3	3	0	0	0
Rule 15	3	3	1	1	1	3
Rule 16	3	1	1	3	0	0
Rule 17	3	3	1	0	0	0
Rule 18	1	1	3	0	0	0
Rule 19	3	1	0	0	0	0
Rule 20	3	1	0	0	0	0
Rule 21	3	1	0	0	0	0
Rule 22	3	1	3	3	3	0
Rule 23	1	1	3	0	0	0
Rule 24	1	3	3	0	0	0
Rule 25	3	1	0	0	0	0
Rule 26	1	3	3	0	0	0
Rule 27	1	1	3	0	0	0
Rule 28	3	1	0	0	0	0
Rule 29	3	3	3	0	0	0
Rule 30	3	3	0	0	0	0
Rule 31	1	3	3	3	1	0
Rule 32	3	1	0	0	0	0
Rule 33	3	3	3	3	3	0
Rule 34	3	1	3	0	0	0
Rule 35	1	3	3	3	3	0
Rule 36	3	3	0	0	0	0

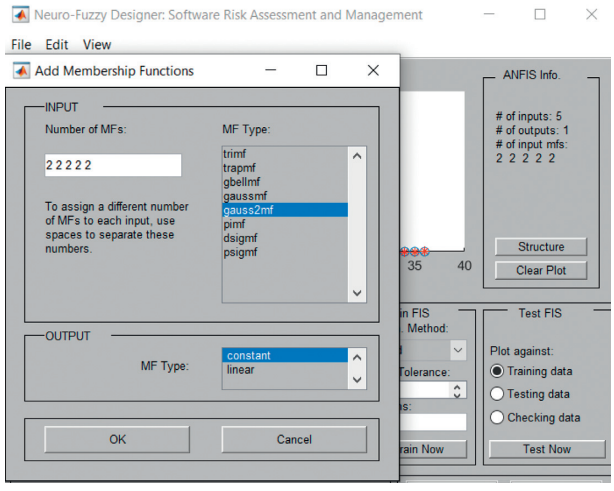

Figure 4: Set of the gauss membership function with the value 2.

The average testing error in Figure 5 has shown up very low (0.000018—18.10-6) after the implementation of ANFIS on MATLAB. That means the rule set, which has been loaded into MATLAB for rule development by ANFIS configuration, has been valid—in other words the fuzzy inference model belonging to the specific rule set has been accurate [73–78]. However, if the set were not correct or accurate, the average testing error would have been a greatly affected size [79, 80]. Furthermore, the designed and developed new software risks rule set and its numeric values have been listed and explained in the following structure.

These numeric values have been determined, assigned, and popped up semiautomatically in the ANFIS implementation on MATLAB. According to the rules, the values 0 to 1 (not included) have been that the situation of the rule parameter is low. Moreover, the values 1 to 2 (not included) have shown that the case of the rule parameter is medium. Furthermore, the values 2 to 3 (included) have meant that the state of the rule parameter is high. Based on the risk parameters in the rules (as listed in Table 2) and their values (as shown in Table 3), Adaptive Neuro-Fuzzy Inference System—ANFIS—has implemented, designed, developed, and customized original software risks rule set with their numeric values (OutputValues.txt in the Supplemental Files (available (here)) as listed in the following:

Developed Rule 1: Time zone difference is medium & Communication problems is medium & Lack of trust is medium & Process maturity is medium & Coupling is medium ⟶ Productivity drop is low.

Numeric values: 2 2 1.5 1.5 1.5 0.97

Developed Rule 2: Coupling is high & Time zone difference is high & Communication problems is medium & Cultural difference is medium & Coordination problems is high ⟶ Risk of project failures is medium.

Numeric values: 2.43 2.46 1.92 1.88 2.23 1.65

Developed Rule 3: Time pressure is high & Novelty of product is high & Cultural difference is high & Number of sites is high & Requirements stability is medium-⟶ Productivity downfall is medium.

Numeric values: 2.26 2.26 2.28 2.36 1.3 1.28

Developed Rule 4: Coupling is high & Number of sites is high & Staff motivation is low & Personal relationships is medium & Transparency is medium-⟶ Quality problems is high.

Numeric values: 2.26 2.52 0.97 1.83 1.3 2.98

Developed Rule 5: Formality is medium & Transparency is medium & Language difference is low & Common experiences is medium & Coordination problems is medium ⟶ Lack of trust is low.

Numeric values: 1.38 1.78 0.50 1.65 1.86 0.17

Developed Rule 6: Cultural difference is medium & Quality problems is medium & Communication

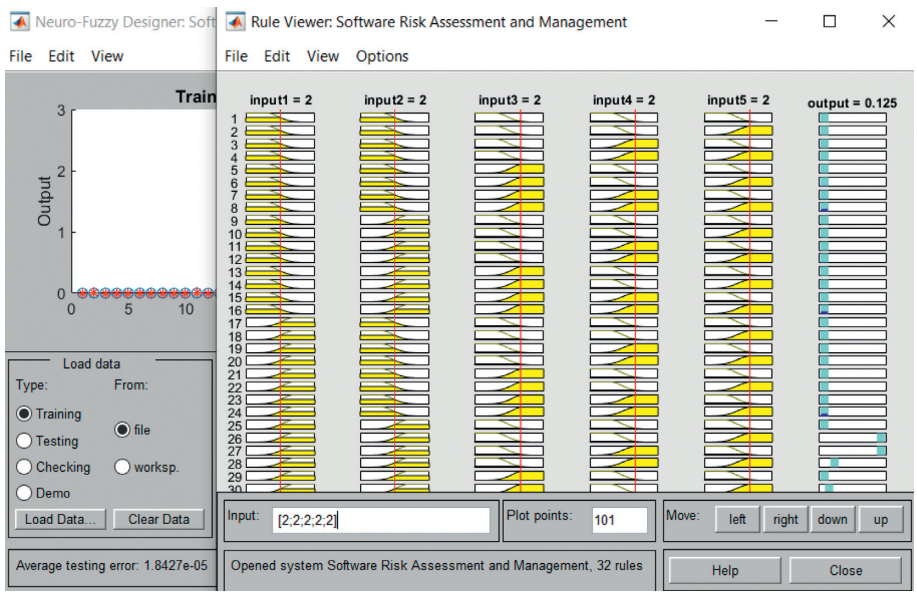

Figure 5: Development of the rules with ANFIS on MATLAB.

problems is low & Time zone difference is medium & Communication infrastructure is medium ⟶ phase = requirements is medium.

Numeric values: 1.59 1.28 0.33 1.61 1.35 1.29

Developed Rule 7: Language difference is high & Cultural difference is medium & Risk of project failures is low & Personal relationships is medium & Productivity downfall is medium ⟶ Communication problems is medium.

Numeric values: 2.32 1.71 0.59 1.13 1.43 1.94

Developed Rule 8: Coordination problems is medium & Transparency is high & Lack of trust is low & Communication infrastructure is medium & Language difference is high ⟶ Formality is medium.

Numeric values: 1.33 2.43 0.88 1.18 2.55 1.77

Developed Rule 9: Travel cost overhead is medium & Task coupling is medium & Process knowledge is low & Time pressure is high & Requirements stability is high ⟶ Maturity is medium.

Numeric values: 1.82 1.43 0.67 2.22 2.33 1.70

Developed Rule 10: Project experience is medium & IP protection issues is medium & Transparency is low & Coordination problems is low & Lack of trust is high ⟶ Quality problems is medium

Numeric values: 1.32 1.35 0.74 0.49 2.18 1.39

Developed Rule 11: Risk of project failure is medium & Quality problems is medium & Common experiences is medium & Productivity drop is low & Size is low- ⟶ Time pressure is medium

Numeric values: 1.51 1.47 1.52 0.42 0.89 1.37

Developed Rule 12: Coordination problems is medium & Process maturity is medium & Lack of trust is low & Requirements stability is low & Time pressure is medium ⟶ Task coupling is high

Numeric values: 1.75 1.93 0.84 0.17 1.44 2.22

Developed Rule 13: Travel cost overhead is medium & IP protection issues is medium & Transparency is medium & Personal relations is low & Task coupling is high ⟶ Process maturity is medium

Numeric values: 1.94 1.52 1.54 0.33 2.16 1.88

Developed Rule 14: Communication problems is medium & Quality problems is high & Productivity drop is low & Process knowledge is low & Requirements stability is medium ⟶ Common experiences is medium

Numeric values: 1.18 2.15 0.38 0.23 1.41 1.33

Developed Rule 15: Application knowledge is high & Process knowledge is high & Communication infrastructure is high & Technical knowledge is low & Staff motivation is medium ⟶ Time pressure is low

Numeric values: 2.38 2.43 2.24 0.82 1.5 0.6

Developed Rule 16: Cultural difference is medium & Coordination problems is high & Cost overhead is medium & Productivity problems is medium & Time pressure is medium ⟶ Maturity is low

Numeric values: 1.11 2.25 1.33 1.51 1.91 0.41

Developed Rule 17: Travel cost overhead is medium & Productivity drop is medium & Transparency is low & Application knowledge is medium & Technical knowledge is high ⟶ Quality problem is medium

Numeric values: 1.22 1.24 0.95 1.16 2.21 1.27

Developed Rule 18: Communication problems is medium & Requirements stability is medium & Language difference is medium & Cultural difference is medium & Communication infrastructure is low ⟶ Cost overhead is medium

Numeric values: 1.32 1.15 1.92 1.94 0.82 1.37

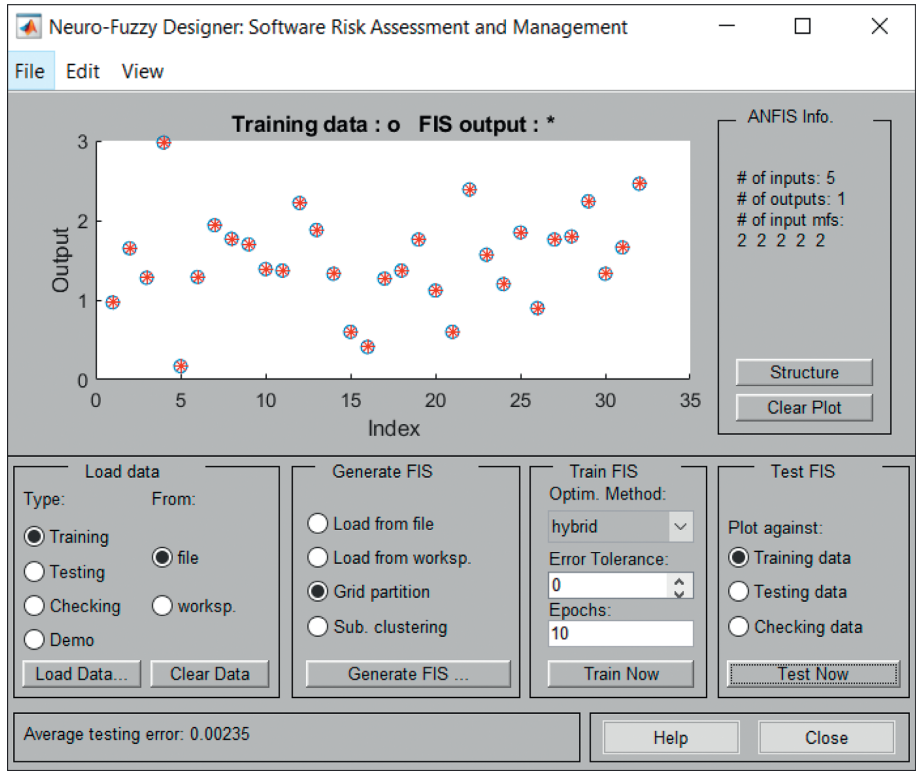

Figure 6: The result of the developed rules implementation with ANFIS on MATLAB.

Developed Rule 19: Lack of trust is high & Transparency is medium & Complexity is low & Coupling is low & Cultural differences is medium ⟶ Time pressure is medium

Numeric values: 2.38 1.86 0.47 0.98 1.97 1.76

Developed Rule 20: Number of sites is medium & Coordination problem is medium & Phase = testing is low & Process maturity is medium & Project failure risk is high ⟶ Transparency is medium

Numeric values: 1.31 1.32 0.22 1.17 2.15 1.12

Developed Rule 21: Time zone difference is medium & Novelty of product is medium & Common experiences is medium & Communication infrastructure is low & Technical knowledge is low ⟶ Process maturity is low

Numeric values: 1.13 1.14 1.33 0.65 0.85 0.6

Developed Rule 22: Phase coding is medium & Productivity downfall is high & Communication problems is high & Transparency is medium & Common experiences is low ⟶ Coordination problem is high

Numeric values: 1.17 2.17 2.18 1.95 0.63 2.39

Developed Rule 23: Productivity drop is medium & Application knowledge is high & Time pressure is high & Project experience is high & Coupling is low-⟶ Process knowledge is medium

Numeric values: 1.2 2.43 2.51 2.32 0.3 1.57

Developed Rule 24: Coupling is low & Maturity is high & Lack of trust is medium & Quality problems is low & Communication problems is low ⟶ Risk of project failures is medium

Numeric values: 0.97 2.52 1.59 0.76 0.89 1.20

Developed Rule 25: Cultural difference is medium & Novelty of product is medium & Cost overhead is low & Number of sites is medium & Task coupling is medium ⟶ Application knowledge is medium

Numeric values: 1.42 1.72 0.78 1.41 1.72 1.85

Developed Rule 26: Personal relations is medium & Productivity drop is high & Staff motivation is medium & Process maturity is low & Technical knowledge is high ⟶ Coordination problems is low

Numeric values: 1.63 2.14 1.25 0.73 2.19 0.90

Developed Rule 27: Communication problem is high & IP protection issues is high & Coupling is low & Number of sites is low & Personal relations is medium ⟶ Lack of trust is medium

Numeric values: 2.19 2.37 0.79 0.54 1.48 1.76

Developed Rule 28: Complexity is high & Quality problems is medium & Common experiences is low & Time zone difference is low & Transparency is high-⟶ Number of sites is medium

Numeric values: 2.37 1.77 0.54 0.42 2.16 1.80

Developed Rule 29: Phase coding is high & Phase = testing is medium & Communication infrastructure is medium & Requirements stability is low & Time pressure is high ⟶ Risk of project failures is high

Numeric values: 2.46 1.42 1.62 0.73 2.53 2.24

Developed Rule 30: Formality is medium & Size is medium & Maturity is low & Cultural difference is low & Lack of trust is low ⟶ Process maturity is medium

Numeric values: 1.11 1.91 0.86 0.49 0.52 1.33

Developed Rule 31: IP protection issues is high & Cultural difference is medium & Time zone difference is medium & Process knowledge is medium & Language difference is high ⟶ Lack of trust is medium

Numeric values: 2.23 1.38 1.46 1.56 2.23 1.66

Developed Rule 32: Productivity drop is high & Complexity is medium & Process maturity is low & Transparency is low & Personal relationships is medium ⟶ Coordination problem is high

Numeric values: 2.45 1.45 0.65 0.57 1.25 2.46

Figure 6 has proved the validity (the model validity, as well) of the developed rules as given above with ANFIS method on MATLAB by showing the average testing error: 0.00235—which has been too low as in Figure 5: 0.000018. Also, it could be said that the less average testing error, the more valid the machine learning fuzzy structure model (the software risks rule set) [73–78]. In addition, these results figure that the developed software risk rules have a valid structure with a high accuracy rate (low average testing error). So, they could be used and applied effectively and appropriately in software risk assessment and management according to this scientific value.

4. Conclusions and Future Extensions

36 software risk rules [59] which have been proved as a valid and recognizable model have been configured and implemented by ANFIS (Adaptive Neuro-Fuzzy Inference System)—the machine learning algorithm integrated with fuzzy artificial intelligence structure. Afterward, 32 (membership value is 2 and input numbers are 5, so $2^5 = 32$ outputs) new and original rules have been created and generated with very low average testing error in order to build the original linguistic software risks rule set. In addition, each rule has six parameters—five factors for the input and one factor for the output—in total. Furthermore, in light of this obtained data, it has pointed and stated that Adaptive Neuro-Fuzzy Inference System (ANFIS) method could work effectively and sufficiently in the area of software risk assessment and management by giving the trustworthy results on MATLAB. Moreover, this situation has stated that this implementation has given several different outputs based on the input rules and parameters by learning itself, and so, it has resulted in that machines are able to give some rules not to be seen, not to be recognized, and not to be told by people. This novel approach will provide software managers to evaluate risks in the software development process widely with the help of fuzzy inference rules without holding only to software experts' opinions. So, the success rate of software projects will increase perceptibly.

As a result of the analysis and the research about software risk assessment and management; 32 main software risk rules based on "fuzzy structure" [7, 8]—suitable with uncertainty like in the risks' nature—have been developed and figured out: if one pays attention to these 32 rules, risk in the software development process will have decreased. According to the results of this evaluation of software risk rules based on "Fuzzy Inference System," "manpower," "time," and "price" which are the main resources of the software development process will be used more effectively. Thus, the benefits of "fuzzy logic" in "software risk assessment and management" will be seen more clearly and tangible.

Peter DRUCKER—the management expert—claimed and proposed that one cannot manage the process which s/he does not measure. This explanation tells us that the software risk assessment and management have to be measured and evaluated by valid software risk rules based on the clear and objective software risk parameters in order to have an effective management process. With the contribution of Adaptive Network-Based Fuzzy Inference System (ANFIS) machine learning algorithm, a valid software risks rule set has been designed and developed, and its validation has been proven by the low average testing error of ANFIS configuration on MATLAB. According to the results of the valid software risk parameters rule set, "software developers," which are the fundamental assets of programming improvement interaction, might be utilized all the more successfully. And afterward, the advantages of the "Software Risk Assessment and Management" might be met and taken more unmistakably and more understandably in the logical zone and in the scientific area.

For the future extension of this study, a survey about software risks and their linguistic rules may be applied and taken with some software experts to get their feedbacks and their subjective evaluation (subjective validity and subjective accuracy level). Furthermore, genetic algorithms may be added into the ANFIS method (hybrid method and implementation) in order to determine the order of importance for the outputs—software risk parameters and rules.

Data Availability

The data (InputValues.txt and OutputValues.txt) used to support the findings of this study are included within the Supplementary Materials.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Supplementary Materials

The numeric values belonging to the software risk rules (36 software risk rules) in Table 2 have been attached into

InputValues.txt. These values have been used as input in the fuzzy inference system on MATLAB for developing new rules. In addition, based on the values in InputValues.txt, the values in OutputValues.txt have been popped up and assigned semiautomatically to new and original developed rules (32 software risk rules) in the fuzzy inference system on MATLAB. (*Supplementary Materials*)

References

[1] M. Smith, "The people risks," *Computer Law & Security Review*, vol. 4, no. 6, pp. 2–6, 1989.

[2] L. I. Lezzoni, "The risks of risk adjustment," *JAMA: The Journal of the American Medical Association*, vol. 278, no. 19, pp. 1600–1607, 1997.

[3] P. K. Dey, J. Kinch, and S. O. Ogunlana, "Managing risk in software development projects: a case study," *Industrial Management & Data Systems*, vol. 107, no. 2, pp. 284–303, 2007.

[4] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[5] B. Boehm, "Software risk management," in *Lecture Notes in Computer Science*, vol. 387, pp. 1–19, Springer, Berlin, Germany, 1st edition, 1989.

[6] B. W. Boehm, "Software risk management: principles and practices," *IEEE Software*, vol. 8, no. 1, pp. 32–41, 1991.

[7] S. Ali, H. Li, S. U. Khan, Y. Zhao, and L. Li, "Fuzzy multi attribute assessment model for software outsourcing partnership formation," *IEEE Access*, vol. 6, pp. 55431–55461, 2018.

[8] S. Ali, N. Ullah, M. F. Abrar, Z. Yang, and J. Huang, "Fuzzy multicriteria decision-making approach for measuring the possibility of cloud adoption for software testing," *Scientific Programming*, vol. 2020, Article ID 6597316, 24 pages, 2020.

[9] P. Gerrard and N. Thompson, *Risk-based E-Business Testing*, Artech House, Norwood, MA, USA, 1st edition, 2002.

[10] E. M. Hall, *Managing Risk: Methods for Software Systems Development*, Addison-Wesley Professional, Boston, MA, USA, 1st edition, 1998.

[11] S. L. Pfleeger, L. Hatton, and C. C. Howell, *Solid Software*, Prentice Hall, Hoboken, NJ, USA, 1st edition, 2002.

[12] K. Heldman, *Project Manager's Spotlight on Risk Management*, Jossey-Bass, San Francisco, CA, USA, 1st edition, 2005.

[13] M. J. Carr, S. L. Konda, I. Monarch, F. C. Ulrich, and C. F. Walker, *Taxonomy-based Risk Identification*, Software Engineering Institute, Pittsburgh, PA, USA, 1st edition, 1993.

[14] DoD, *U.S. Risk Management Guide for DoD Acquisition*, Defense Acquisition University, Fort Belvoir, VA, USA, 5th edition, 2003.

[15] A. J. Dorofee, J. A. Walker, C. J. Alberts, R. P. Higuera, and R. L. Murphy, *Continuous Risk Management Guidebook*, Software Engineering Institute, Pittsburgh, PA, USA, 1st edition, 1996.

[16] R. Fairley, "Risk management for software projects," *IEEE Software*, vol. 11, no. 3, pp. 57–67, 1994.

[17] J. Kontio, *Software engineering risk management: a method, improvement framework, and empirical evaluation*, Ph.D. thesis, Helsinki University of Technology, Espoo, Finland, 2001.

[18] G. Stoneburner, A. Goguen, and A. Feringa, *Risk Management Guide for Information Technology Systems and Underlying Technical Models for Information Technology Security*, National Institute of Standards and Technology, Gaithesburg, MD, USA, 1st edition, 2002.

[19] H. Van Loon, "A management methodology to reduce risk and improve quality," *IT Professional*, vol. 9, no. 6, pp. 30–35, 2007.

[20] P. L. Bannerman, "A reassessment of risk management in software projects," in *Handbook on Project Management and Scheduling*, vol. 2, pp. 1119–1134, Springer, Berlin, Germany, 1st edition, 2015.

[21] P. Qinghua, "A model of risk assessment of software project based on grey theory," in *Proceedings of 2009 4th International Conference on Computer Science & Education*, pp. 538–541, Nanning, China, July 2009.

[22] Y. Fu, M. Li, and F. Chen, "Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix," *International Journal of Project Management*, vol. 30, no. 3, pp. 363–373, 2012.

[23] J. L. Salmeron and C. Lopez, "Forecasting risk impact on ERP maintenance with augmented fuzzy cognitive maps," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 439–452, 2012.

[24] A. G. Tang and R. L. Wang, "Software project risk assessment model based on fuzzy theory," in *Proceedings of 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*, pp. 328–330, Chengdu, China, June 2010.

[25] C.-F. Fan and Y.-C. Yu, "BBN-based software project risk management," *Journal of Systems and Software*, vol. 73, no. 2, pp. 193–203, 2004.

[26] T. R. Trigo, C. Gusmão, and A. Lins, "CBR risk—risk identification method using case based reasoning," in *Proceedings of International Conference on Information Systems and Technology Management*, São Paulo, Brazil, 2008.

[27] K. De Bakker, A. Boonstra, and H. Wortmann, "Does risk management contribute to IT project success? A meta-analysis of empirical evidence," *International Journal of Project Management*, vol. 28, no. 5, pp. 493–503, 2010.

[28] W.-M. Han and S.-J. Huang, "An empirical analysis of risk components and performance on software projects," *Journal of Systems and Software*, vol. 80, no. 1, pp. 42–50, 2007.

[29] J. Jiang and G. Klein, "Software development risks to project effectiveness," *Journal of Systems and Software*, vol. 52, no. 1, pp. 3–10, 2000.

[30] J. J. Jiang, G. Klein, and R. Discenza, "Information system success as impacted by risks and development strategies," *IEEE Transactions on Engineering Management*, vol. 48, no. 1, pp. 46–55, 2001.

[31] T. Raz, A. J. Shenhar, and D. Dvir, "Risk management, project success, and technological uncertainty," *R&D Management*, vol. 32, no. 2, pp. 101–109, 2002.

[32] L. Wallace and M. Keil, "Software project risks and their effect on outcomes," *Communications of the ACM*, vol. 47, no. 4, pp. 68–73, 2004.

[33] L. Wallace, M. Keil, and A. Rai, "Understanding software project risk: a cluster analysis," *Information & Management*, vol. 42, no. 1, pp. 115–125, 2004.

[34] L. Wallace, M. Keil, and A. Rai, "How software project risk affects project performance: an investigation of the dimensions of risk and an exploratory model," *Decision Sciences*, vol. 35, no. 2, pp. 289–321, 2004.

[35] A. G. Yu, "Software crisis, what software crisis?" in *Proceedings of 2009 International Conference on Management and Service Science*, pp. 1–4, Beijing, China, September 2009.

[36] M. J. Gallivan, "The influence of system developers' creative style on their attitudes toward and assimilation of a software process innovation," in *Proceedings of the 31st Hawaii International Conference on System Sciences*, pp. 435–444, Kohala Coast, Hawaii, January 1998.

[37] S. Sawyer and P. J. Guinan, "Software development: processes and performance," *IBM Systems Journal*, vol. 37, no. 4, pp. 552–569, 1998.

[38] T. Hall, D. Wilson, A. Rainer, and D. Jagielska, "The neglected technical skill?" in *Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research: The Global Information Technology Workforce*, pp. 196–202, St. Louis, MO, USA, April 2007.

[39] H. Baggelaar, "Evaluating programmer performance visualizing the impact of programmers on project goals," M.Sc. thesis, University of Amsterdam, Amsterdam, Netherlands, 2008.

[40] K. Lee, K. Joshi, and Y. Kim, "Person-job fit as a moderator of the relationship between emotional intelligence and job performance," in *Proceedings of 2008 Computer Personnel Doctoral Consortium and Research Conference*, pp. 70–75, Charlottesville, VA, USA, April 2008.

[41] C. Ting, "The application of the function point analysis in software developers' performance evaluation," in *Proceedings of 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, Dalian, China, October 2008.

[42] S. Zhang, Y. Wang, and J. Xiao, "Mining individual performance indicators in collaborative development using software repositories," in *Proceedings of PSEC 2008: 15th Asia-Pacific Software Engineering Conference*, pp. 247–254, Beijing, China, December 2008.

[43] G. Calikli and A. Bener, "Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, pp. 1–11, Timişoara, Romania, September 2010.

[44] M. A. Chilton, B. C. Hardgrave, and D. J. Armstrong, "Performance and strain levels of it workers engaged in rapidly changing environments: a person-job fit perspective," *ACM SIGMIS Database: The Database for Advances in Information Systems*, vol. 41, no. 1, pp. 8–35, 2010.

[45] R. Ramler, C. Klammer, and T. Natschläger, "The usual suspects: a case study on delivered defects per developer," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1–4, Bolzano, Italy, September 2010.

[46] Y. Wang and M. Zhang, "Penalty policies in professional software development practice: a multi-method field study," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pp. 39–47, Cape Town, South Africa, May 2010.

[47] V. Balijepally, S. Nerur, and R. Mahapatra, "Effect of task mental models on software developer's performance: an experimental investigation," in *Proceedings of 2012 45th Hawaii International Conference on System Sciences*, pp. 5442–5451, Maui, HI, USA, January 2012.

[48] C. B. Duarte, J. P. Faria, and M. Raza, "PSP PAIR: automated personal software process performance analysis and improvement recommendation," in *Proceedings of 2012 8th International Conference on the Quality of Information and Communications Technology*, pp. 131–136, Lisbon, Portugal, September 2012.

[49] K. Ehrlich and M. Cataldo, "All-for-one and one-for-all?: a multi-level analysis of communication patterns and individual performance in geographically distributed software development," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, pp. 945–954, Seattle, WA, USA, February 2012.

[50] B. Kelly and H. M. Haddad, "Metric techniques for maintenance programmers in a maintenance ticket environment," *Journal of Computing Sciences in Colleges*, vol. 28, no. 2, pp. 170–178, 2012.

[51] A. Schröter, J. Aranda, D. Damian, and I. Kwan, "To talk or not to talk: factors that influence communication around changesets," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, pp. 1317–1326, Seattle, WA, USA, February 2012.

[52] D. Westermann, "A generic methodology to derive domain-specific performance feedback for developers," in *Proceedings of the 34th International Conference on Software Engineering*, pp. 1527–1530, Zurich, Switzerland, June 2012.

[53] G. Calikli and A. Bener, "An algorithmic approach to missing data problem in modeling human aspects in software development," in *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, pp. 1–10, Baltimore, MD, USA, October 2013.

[54] M. Keil, P. E. Cule, K. Lyytinen, and R. C. Schmidt, "A framework for identifying software project risks," *Communications of the ACM*, vol. 41, no. 11, pp. 76–83, 1998.

[55] M. Warkentin, R. S. Moore, E. Bekkering, and A. C. Johnston, "Analysis of systems development project risks: an integrative framework," *ACM SIGMIS Database: The Database for Advances in Information Systems*, vol. 40, no. 2, pp. 8–27, 2009.

[56] L. Xiaosong, L. Shushi, C. Wengun, and F. Songjiang, "The application of risk matrix to software project risk management," in *Proceedings of International Forum on Information Technology and Applications*, pp. 480–483, Chengdu, China, May 2009.

[57] N. Feng, M. Li, and H. Gao, "A software project risk analysis model based on evidential reasoning approach," in *Proceedings of 2009 WRI World Congress on Software Engineering*, pp. 224–228, Xiamen, China, May 2009.

[58] S. Khan, "An approach to facilitate software risk identification," in *Proceedings of 2009 2nd International Conference on Computer, Control & Communication*, Karachi, Pakistan, February 2009.

[59] A. Lamersdorf, J. Munch, A. F. Torre, C. R. Sánchez, M. Heinz, and D. Rombach, "A rule-based model for customized risk identification in distributed software development projects," in *Proceedings of 2010 5th IEEE International Conference on Global Software Engineering*, pp. 209–218, Princeton, NJ, USA, August 2010.

[60] J. Becker and R. W. Dai, "Understanding IT project risks as disturbances to digital ecosystems," in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pp. 137–142, San Francisco, CA, USA, November 2011.

[61] S. Hoermann, M. Aust, M. Schermann, and H. Krcmar, "Comparing risks in individual software development and standard software implementation projects: a delphi study," in *Proceedings of 2012 45th Hawaii International Conference on System Sciences*, pp. 4884–4893, Maui, HI, USA, January 2012.

[62] M. Z. Fakhar, M. Abbas, and M. Waris, "Risk management system for ERP software project," in *Proceedings of 2013 Science and Information Conference*, pp. 223–228, London, UK, October 2013.

[63] P. Sonchan and S. Ramingwong, "Top twenty risks in software projects: a content analysis and Delphi study," in *Proceedings of 2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pp. 1–6, Nakhon Ratchasima, Thailand, May 2014.

[64] W. Min, Z. Jun, and Z. Wei, "The application of fuzzy comprehensive evaluation method in the software project risk assessment," in *Proceedings of 2017 International Conference on Management Engineering, Software Engineering and Service Science*, pp. 76–79, Wuhan, China, January 2017.

[65] T. J. Ross, *Fuzzy Logic with Engineering Applications*, Wiley, Hoboken, NJ, USA, 4th edition, 2016.

[66] C. Carlsson and R. Fullér, "A fuzzy approach to real option valuation," *Fuzzy Sets and Systems*, vol. 139, no. 2, pp. 297–312, 2003.

[67] Q. Shen and A. Chouchoulas, "A rough-fuzzy approach for generating classification rules," *Pattern Recognition*, vol. 35, no. 11, pp. 2425–2438, 2002.

[68] A. D. Dongare, R. R. Kharde, and A. D. Kachare, "Introduction to artificial neural network," *International Journal of Engineering and Innovative Technology*, vol. 2, no. 1, pp. 189–194, 2012.

[69] J. Zou, Y. Han, and S.-S. So, "Overview of artificial neural networks," in *Methods in Molecular Biology*, vol. 458, pp. 14–22, Humana Press, Totowa, NJ, USA, 1st edition, 2008.

[70] N. Walia, H. Singh, and A. Sharma, "ANFIS: adaptive neuro-fuzzy inference system—a survey," *International Journal of Computer Applications*, vol. 123, no. 13, pp. 32–38, 2015.

[71] S. C. Chelgani, B. Hart, W. C. Grady, and J. C. Hower, "Study relationship between inorganic and organic coal analysis with gross calorific value by multiple regression and ANFIS," *International Journal of Coal Preparation and Utilization*, vol. 31, no. 1, pp. 9–19, 2011.

[72] M. A. Shoorehdeli, M. Teshnehlab, and A. K. Sedigh, "A novel training algorithm in ANFIS structure," in *Proceedings of 2006 American Control Conference*, pp. 6–16, Minneapolis, MN, USA, June 2006.

[73] H. Doufesh, F. Ibrahim, N. A. Ismail, and W. A. W. Ahmad, "Adaptive neurofuzzy inference system for predicting alpha band power of EEG during muslim prayer (SALAT)," *Biomedical Engineering: Applications, Basis and Communications*, vol. 28, no. 6, pp. 6–15, 2016.

[74] S. Jassar, Z. Liao, and L. Zhao, "Impact of data quality on predictive accuracy of ANFIS based soft sensor models," in *Proceedings of the World Congress on Engineering and Computer Science*, San Francisco, CA, USA, October 2009.

[75] S. Kaynak, H. Evirgen, and B. Kaynak, "Adaptive neuro-fuzzy inference system in predicting the success of student's in a particular course," *International Journal of Computer Theory and Engineering*, vol. 7, no. 1, pp. 34–39, 2015.

[76] H. Mohammed, I. Hameed, and R. Seidu, "Adaptive neuro-fuzzy inference system for predicting norovirus in drinking water supply," in *Proceedings of 2017 International Conference on Informatics, Health & Technology*, pp. 1–6, Riyadh, Saudi Arabia, February 2017.

[77] M. Neshat, A. Deli, A. Masoumi, and M. Sargolzae, "A comparative study on ANFIS and fuzzy expert system models for concrete mix design," *International Journal of Computer Science Issues*, vol. 8, no. 3, pp. 196–210, 2011.

[78] K. Salehi, S. M. Khazraee, F. S. Hoseini, and F. K. Mostafazadeh, "Laboratory biogas production from kitchen wastes and applying an adaptive neuro fuzzy inference system as a prediction model," *International Journal of Environmental Science and Development*, vol. 5, no. 3, pp. 290–293, 2014.

[79] F. H. Ismail, M. A. Aziz, and A. E. Hassanien, "Optimizing the parameters of Sugeno based adaptive neuro fuzzy using artificial bee colony: a case study on predicting the wind speed," in *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*, pp. 645–651, Gdańsk, Poland, September 2016.

[80] S. R. Nikam, P. J. Nikumbh, and S. P. Kulkarni, "Fuzzy logic and neuro-fuzzy modeling," in *Proceedings on National Conference on Recent Trends in Computing*, pp. 22–31, April 2012.