
Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi
Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering

Basılı/Printed ISSN: 1302-9304. Elektronik/Online ISSN: 2547-958X

Software Risk Assessment and Management with Rules Based on Fuzzy Approach

Bulanık Yaklaşımlı Kurallar ile Yazılım Risk Değerlendirmesi ve Yönetimi

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

[1] Dokuz Eylül University, Graduate School of Natural and Applied Sciences, Department of Computer Engineering, İzmir, TURKEY

[2] Dokuz Eylül University, Faculty of Engineering, Department of Computer Engineering, İzmir, TURKEY

[3] Burdur Mehmet Akif Ersoy University, Faculty of Engineering and Architecture, Department of Computer Engineering, Burdur, TURKEY

*Corresponding Author / Sorumlu Yazar *:* mbatar@cs.deu.edu.tr

Geliş Tarihi / Received: 07.03.2021 *Araştırma Makalesi/Research Article*
Kabul Tarihi / Accepted: 16.04.2021 *DOI:10.21205/deufmd.2021236918*
Atıf şekli/ How to cite: BATAR M., BIRANT K.U., ISIK A.H.(2021). Software Risk Assessment and Management with Rules Based on Fuzzy Approach. DEUFMD 23(69), 903-911.

Abstract

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 and etc. However, there isn't any universally accepted methodology to apply software risk assessment and management. There are three main reasons of 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 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 of 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 Approach" 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 damages that are 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 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 successfully accomplishment in software development process. In this paper, usability and efficiency of "Fuzzy Approached" linguistic and logical rules based on "Fuzzy Logic" in "Software Risk Assessment and Management" have been shown and expressed in detail.

Keywords: *Software Risk Assessment, Software Risk Management, Fuzzy Approach, Fuzzy Logic*

Öz

Günümüze kadar, yazılım geliştirme projelerini değerlendirmek ve yönetmek için çeşitli yazılım risk parametreleri belirlenmiştir: üretkenlik, taahhüt, kaliteye önem verme, koda dayalı bilgi, beceri ve değerlendirme, kodlamanın genel yapısına ve kurallarına uygunluk, öğrenme becerisi, kişisel sorumluluk bilinci, vb. Ancak, yazılım risk değerlendirmesi ve yönetimini uygulamak amacıyla dünyaca herkes tarafından kabul görmüş herhangi bir yöntem maalesef yoktur. Bu can alıcı durumun üç ana sebebi vardır: Birinci sebep, geliştirilen her bir yazılım parçası kendi içerisinde tektir. Fakat aynı yazılım parçasını geliştirmek ve ilerletmek için onu sil baştan yaratmaya gerek yoktur; elimizde var olan hali hazırdaki yazılım parçasını kopyalayarak, üzerinde oynama yaparak, değiştirerek bu durum çözülebilmektedir. Bu da, birbirine benzer iki yazılım parçası arasında hem nitelik hem de nicelik bakımından tam doğru bir karşılaştırmanın yapılamamasına neden olmaktadır. İkinci sebep, günümüz teknolojisi sürekli değişen, gelişen ve kendini yenileyen bir süreç içerisindedir. Bunun doğal bir sonucu olarak, yazılım projelerinin risk değerlendirmesi ve yönetiminin kullandığı bir teknoloji ya da teknoloji esaslı yazılım risk parametreleri dizisi çok geçmeden önemini kaybetmiş ve eskimiş duruma gelmektedir. Bunun neticesinde, bu yöntem ve parametreler işe yaramaz duruma gelmektedir. Üçüncü sebep, verilen aynı problemi çözmek için birden çok, birbirinden tamamen farklı çeşitli yöntemler geliştirilip özgünlük, kendine has olma, yaratıcılık kavramları had safhaya çıkarılabilmektedir. Bu da elle tutulur, somut verilerin yazılım risk parametrelerinin oldukça çeşitli olduğunu bizlere göstermektedir. Bu nedenlerden dolayı, "belirsizlik" kavramını içeren "Bulanık Yaklaşım" tekniği, – doğası gereği "belirsiz" yapıda olan – yazılım risk parametrelerini tanımlamak ve belirlemek için oldukça uygun bir sürece sahiptir. Ayrıca yazılım riskleri, bir sistemde kötü veya istenmeyen olayların meydana gelmesiyle ortaya çıkan kusurların olma ihtimali ve şiddeti olarak tanımlanır. Yazılım risklerinden dolayı, sistem stratejik, finansal, işlevsel (operational), yapısal veya bütünlük kaybı yaşayabilmektedir. Bu kayıpların bertaraf edilebilmesi ve yazılım geliştirme sürecinde gerçek bir başarı sağlanabilmesi için yazılım riskleri – hasara neden olmadan – zamanında belirlenmeli ve etkin bir "Yazılım Risk Değerlendirmesi ve Yönetimi" uygulanıp yürütülmelidir. Bu makalede, "Bulanık Mantık" yöntemine dayalı "Bulanık Yaklaşımlı" dilbilimsel ve mantıksal kuralların "Yazılım Risk Değerlendirmesi ve Yönetimi" alanında kullanılabilirliği ve etkinliği ayrıntılı olarak gösterilmiştir.

Anahtar kelimeler: *Yazılım Risk Değerlendirmesi, Yazılım Risk Yönetimi, Bulanık Yaklaşım, Bulanık Mantık*

1. Risk

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 basic two factors: the possibility of occurrence – the likelihood of not achieving a particular result or the likelihood of an undesired occurrence –, size of loss – the effects of the consequences that would arise if the risks were realized [1].

There are three main risk categories. Internal risks about the company: risks related to production management's effectiveness, risks related to financial management activity, risks related to the effectiveness of marketing management, risks related to in-house logistics, risks related to quality management's effectiveness, risks related to the effectiveness of human resources management, general risks about management. Risks about supply chain network: the risk that the suppliers cannot supply the input of production at the desired amount, risk of not delivering on time to suppliers, the risk that suppliers cannot achieve the desired quality standards, the risk that suppliers and distribution companies are not

able to provide the cost, risks of vendors and distribution companies with critical prescriptions for the company cannot keep up with the fast technological advances, risk of non-strategic cooperation with suppliers and distribution companies that have critical prescription for the company, the risk that distribution companies cannot reach products on time, risk of damage to the products or decrease in product quality during distribution, risks arising from the fact that an effective information network has not been established with suppliers and distribution companies, especially those with strategic priorities, or that this information network cannot be used effectively, risks that may arise from fulfilling the company's basic logistics functions either partially or completely through outsourcing. External risks: risks arising from economic uncertainties, risks of political instability, risks arising from technological developments, risks of changing legal conditions, risks created by changes in socio-economic status, risks created by increased competition and changing competition conditions, the risk of a large change in customer expectations, natural disaster risk, the risk of terrorism [2].

Risks come on the whole sizes and shapes; hazard experts for the most part perceive three significant sorts. Market risk is the danger that costs will move in a manner that has contrary 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 come up short or that an outer occasion (seismic tremor, fire, and so on) will contrarily affect the organization [3].

2. Software Risks

Building and keeping up software can be a hazardous business. Most undertakings rely upon programming – so additional cost, delays or the failure to acknowledge objectives – can have genuine results. Bigger dangers that can undermine long haul ventures require prompt consideration, and that implies putting the accentuation on danger (top 10) [4]:

Estimation and planning – The one of a kind sort of individual programming ventures makes issues for designers and administrators in assessing and booking improvement time. Continuously, screen existing activities so utilization of exercises learnt later on.

Sudden development in necessities – As a task advances, gives that are not distinguished before can make a very late obstacle to fulfilling time constraints. Attempt to plan for an impressive future from the get-go in extend and envision the most pessimistic scenario or heaviest-use situation.

Employee turnover – Every task has various designers taking a shot at it. At the point when an engineer leaves, the individual may take basic data with that person. This can postpone and now and then crash a whole venture. Guarantee to have assets where colleagues can work together and share information.

Breakdown of detail – During the underlying periods of mix and coding, prerequisites may strife. Besides, designers may locate that even the detail is indistinct or deficient.

Productivity issues – On undertakings including long courses of events, engineers will in general take things simple in any case. Accordingly, in some cases, they lose huge opportunity to finish the task. Set a practical timetable, and stick to it.

Compromising on plans – In request to stall out into the following "genuine" assignments, engineers will in general surge the plan cycle. This is a misuse of programming hours as planning is the most basic piece of programming advancement.

Gold plating – Developers now and then prefer to flaunt their abilities by adding superfluous highlights. For example, an engineer may add Flash to a fundamental login module to make it look "jazzy". Once more, this is a misuse of programming hours.

Procedural dangers – Day-to-day operational exercises may hamper because of ill-advised cycle usage, clashing needs or an absence of lucidity in obligations.

Technical dangers – Sometimes programming improvement firms lessen the usefulness of the product to make up for invades relating to high financial plans and planning. There is consistently a contention between accomplishing most extreme usefulness of the product and pinnacle execution. To make up for inordinate spending plan and timetable

overwhelms, organizations in some cases decrease the usefulness of the product.

Unavoidable dangers – These remember changes for government strategy, the out of date quality of programming or different dangers that can't be controlled or assessed. As the field of programming improvement turns out to be an ever increasing number of complex, the dangers related with it have escalated. It is essential that improvement firms around vital intending to relieve such dangers.

Software development projects on execution give data to help activities, the board examination and dynamic inside an association. In any case, these are defenseless from cost and time overwhelm alongside under-accomplishment with quality. Furthermore, high level of bugs during beginning time of preliminary and business use aren't phenomenal. Despite the fact that administrators guarantee that they deal with the software failures and issues effectively, yet there are confirmations of absence of software management (project and risk management) even by driving software developers. Software development projects experience the ill effects of market hazard, monetary danger and specialized danger. The software developers and engineers should have great responses to the accompanying inquiries to make progress: Regardless of whether the created programming satisfies the clients' interest/necessity? What amount of rivalry it is probably going to confront? Regardless of whether profits by the product outperform the expense of advancement? Is the task in fact doable? Will equipment, programming, and organizations work appropriately? Will the innovation be accessible so as to meet undertaking destinations? Is there any opportunity of the innovation getting out of date previously use? Will security framework work for the duration of its life? There are instances of prominent IT project disappointment in the literature [5].

3. Related Works about Software Risks

In his study, Gallivan have 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 revealed various special points [6]. (18 significant software risks have been determined.)

Sawyer and Guinan 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. In addition, they have tried to find answers to some questions about software risk assessment and management. These questions; software development method, code retention, code library, working time and related to software development documentation [7]. (44 significant software risks have been determined.)

Hall, Wilson, Rainer and Jagielska 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 [8]. (26 significant software risks have been determined.)

In applying software risk assessment and management, Baggelaar 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 software development process [9]. (22 significant software risks have been determined.)

Lee, Joshi and Kim have analyzed and evaluated software risk assessment and management in terms of personality and work habits [10]. (12 significant software risks have been determined.)

Thing has been interested in and focused on the issues of personality, working style, workload and software development process in software risk assessment and management [11]. (14 significant software risks have been determined.)

Zhang, Wang and Xiao first 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 method

(methods), degree of heritability depth and software development issues related to the difficulty [12]. (13 significant software risks have been determined.)

Calikli and Bener have provided a general overview of the software risk assessment and management. In their work, they have showed 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 [13]. (4 significant software risks have been determined.)

Chilton, Hardgrave and Armstrong have put forward several points for software risks. These points have been work-life, working habits, personality, age and gender [14]. (22 significant software risks have been determined.)

Ramler, Klammer and Natschläger have tried to research and find answers to some questions about software quality in software risk assessment and management [15]. (3 significant software risks have been determined.)

Wang and Zhang have highlighted a number of important issues in the software risk assessment and management. These points have been work-life, work experience, workload, education level and gender [16]. (14 significant software risks have been determined.)

In their study, Baljepally, Nerur and Mahapatra tried to find answers of many questions related to personality traits in recognizing software risks [17]. (8 significant software risks have been determined.)

Duarte, Faria and Raza 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 [18]. (10 significant software risks have been determined.)

Ehrlich and Cataldo 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 [19]. (10 significant software risks have been determined.)

Kelly and Haddad have tried to find out the extent to how "error" has an impact into software risk assessment and management [20]. (3 significant software risks have been determined.)

Schröter, Aranda, Damian and Kwan 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 [21]. (22 significant software risks have been determined.)

Westermann 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 [22]. (7 significant software risks have been determined.)

Calikli and Bener have shown some important points in software risk assessment and management. These points; software project development plan and software team psychology [23]. (4 significant software risks have been determined.)

4. Software Risk Assessment and Management

Software risk assessment is to depict the overall strategy where: One perceives threats and danger factors that can cause hurt (hazard recognizing evidence). One separates and evaluates the peril about that risk (risk assessment and danger appraisal). One chooses fitting ways to deal with discard the risk, or control the danger when the hazard can't be shed (risk control) [24].

Software risk assessment is a cautious look at the workplace to perceive those things, conditions, structures, etc. that may cause to hurt, particularly to people. Afterwards ID is made, one explores and evaluates how likely and genuine the risk is. Right when this affirmation is made, one can immediately, pick what measures should be set up to effectively deal with or cope with the underhandedness from happening [24].

The CSA Standard Z1002 "Word related prosperity and security – Hazard ID and end and danger assessment and control" uses the going

with terms: Hazard evaluation – the overall method of risk recognizing evidence, chance assessment, and peril appraisal. Danger recognizing confirmation – the route toward finding, posting, and depicting threats. Peril examination – a system for valuing the possibility of threats and choosing the component of risk. Risk evaluation – the route toward taking a gander at a normal peril against given danger measures to choose the tremendousness of the risk. Peril control – exercises executing danger evaluation decisions [24].

Software risk management is one of the most significant occupations for a venture supervisor. You can think about a hazard as something that you would lean toward not to have occur. Dangers may compromise the task, the product that is being created, or the association. Hazard the board includes envisioning dangers that may influence the undertaking plan or the nature of the product being created, and afterward making a move to maintain a strategic distance from these dangers. Dangers can be arranged by kind of hazard (specialized, hierarchical, and so forth.). A reciprocal arrangement is to characterize risks as per what these dangers influence [25]:

Venture dangers influence the undertaking calendar or assets. A case of a task hazard is the loss of an accomplished framework designer. Finding a supplanting engineer with proper aptitudes and experience may take quite a while; thus, it will take more time to build up the product structure than initially arranged.

Item risks impact the product's design, development, execution or quality. A case of an item chance is the disappointment of a bought segment to proceed true to form. It will impact the framework's general implementation and execution so that is more slow than anticipated.

Business dangers influence the association getting or creating the product. For example, a contender presenting another item is a business hazard. The presentation of a serious item may imply that the suppositions made about deals of existing programming items might be unduly idealistic.

In addition, software risk assessment and management process (the steps) has been showed and illustrated in Figure 1 in the following.

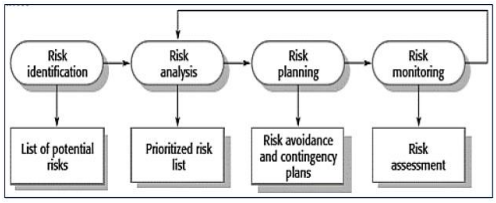

Figure 1. Software risk management process

5. 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 1960s, Dr. Lotfi Zadeh applied the fuzzy logic mentality firstly in his classes in University of California at Berkeley. Fluffy hypothesis can be utilized for as a methods 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 [26].

The participation work is a graphical portrayal used to speak with the impact of each contribution of the "Fuzzy Approach" in the info field. The info field is commonly characterized as a widespread set that can communicate all the circumstances that can happen in a framework. Fluffy rationale standardizes the info articulations to a weight, at that point characterizes the connections between the contributions to effectively display the framework, coming about in a yield esteem. The characterized rules are characterized as the weighting component to decide the impacts of information and yield articulations on fluffy rationale yield sets of the end-product. Making, reviewing and joining capacities produces a fluffy rationale yield that turns out effectively for the framework. All info and yield articulations in fluffy rationale have distinctive participation capacities. The guidelines contain a bunch of characterized decides that are performed utilizing ALSO or AND numerical administrators. Fluffy rationale changes over characterized input articulations into phonetic qualities and fluffy sets. Fluffy rationale has various strategies which are named as Takagi-Sugeno technique and Mamdani strategy. The Tagaki-Sugeno technique proposed by Takagi and Sugeno is the fluffy rationale strategy used to determine fluffy

rationale information and yield articulations in nonlinear frameworks. In the Takagi-Sugeno model, the information on hand, yield articulations are characterized by IF-THEN standards and rules. As indicated by the principles, the yield can be determined with the assistance of a basic recipe. Then again, in Mamdani fluffy rationale technique, numerous rules must be characterized to conform to the fuzzification of participation capacities [27].

"Fuzzy Approach" attempts to model the general 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 furthermore, NO. Utilizing fluffy rationale strategy, it is meant to demonstrate unsure circumstances, inappropriately characterized or complex frameworks [28].

The architecture of "Fuzzy Approach" based on "Fuzzy Logic" comprises of three principle parts as appeared 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 yield 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 a net worth. Along these lines, the framework yields give diverse yield esteems as indicated by the guidelines [29]. Furthermore, Figure 2 in the following has figured out and illustrated fuzzy logic steps and its working mechanism/principle.

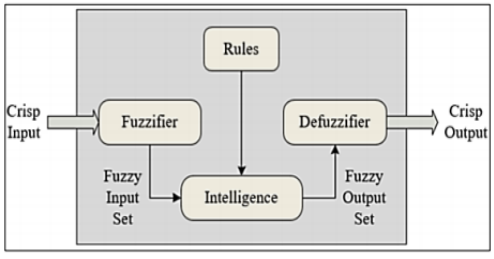

Figure 2. Fuzzy logic process

6. Software Risk Rules Based on Fuzzy Approach

The relation of developed methods to software development method: Using some methods or models which have been designed and implemented before by software developers during software development process shows that there is the property of "Reusability" in software risk assessment and management. This property is the main factor of applying a software development method in software progress since a development method provides software developers to use analyzed, tried and practiced methods and models which make their works easy. That means the property of "Reusability" decrease the cost of the software project in terms of working force and completion time. So, risk in software development process will decrease.

The relation of user requirements to developed methods: A software development project makes the customers' works easy with the contribution of software developers who work methodically based on the software development process. In addition, they can master the user needs more than others, and design and develop a software project which meets the user requirements and the project outputs almost completely. So, risk in software development process will decrease.

The relation of "Exception Handling" to "Error Handling": Applying the property of "Exception Handling" in software developement directs software developers to use the property "Error Handling". Also, the property "Exception Handling" is the main condition of the property "Error Handling". That means if a software developer would like to do error handling operation while programming, s/he has to do exception handling operation before that. So, risk in software development process will decrease.

The relation of software quality to reusability: It can be indicated that paying attention to the quality in the software project by software developers directs them to use the property "Reusability" in software development. Moreover, the software developers who pay attention to the quality while designing and implementing a program, can also apply the property of "Reusability" features easily at the same time since software quality depends on a software development method and this

development process bring to reuse designed and developed methods and models with it. So, risk in software development process will decrease.

The relation of reliable code writing to practical program: In order to ensure reliable operation of the product that will be produced during the software development process and to give the desired results, it requires the reliable writing of the program of the software, which means that the codes of the program are developed according to the reliable structure. Furthermore, if a software can be executed, this may be understood and used. So, risk in software development process will decrease.

The relation of developed methods to software quality: It can be shown that using models which have been determined before in software development directs software developers' companies to guarantee the quality of the software. In addition, the models and methods which have been designed and developed before is accepted as a main factor of software quality since these models and methods are required from software development methods which have an aim of rising the quality in software products. So, risk in software development process will decrease.

7. Conclusion

There are four fundamental motivations to apply, actualize and determine "Software Risk Assessment and Management" in software development process as indicated by Boehm [30,31]: To stay away from overwhelms in arranging and in financial plan, and 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. To design and develop a product arrangement about software projects that the client needs so as to empower the venders to get the consumer loyalty and the ideal benefits.

As a result of the analysis and the research about software risk assessment and management; six main software risk rules based on "Fuzzy Approach" – is suitable with uncertainty like in the risks' nature – have been figured out: *the relation of developed methods to software development method, the relation of user requirements to developed methods, the relation of "Exception Handling" to "Error Handling", the relation of software quality to reusability, the relation of reliable code writing to practical program, the relation of developed methods to software quality.* If one pays attention these 6 rules (relations based on "Fuzzy Logic"), risk in software development process will decrease. According to the results of this evaluation of software risk rules based on "Fuzzy Approach", "manpower", "time" and "price" that are the main resources of 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.

References

[1] Smith, M. 1989. The people risks, Computer Law & Security Review, vol. 4, p. 2-6. DOI: 10.1016/0267-3649(89)90002-2

[2] Renn, O. 2004. Perception of risks, Toxicology Letters, vol. 149, p. 405-413. DOI: 10.1016/j.toxlet.2003.12.051

[3] Lezzoni, L. K. 1997. The risks of risk adjustment, JAMA Journal of the American Medical Association, vol. 278, p. 1600-1607. DOI: 10.1001/jama.278.19.1600

[4] Arnuphaptrairong, T. 2011. Top ten lists of software project risks: Evidence from the literature survey. International MultiConference of Engineers and Computer Scientists, 16-18 March, Hong Kong, 1-6.

[5] Dey, P. K., Kinch, J., Ogunlana, S. O. 2007. Managing risk in software development projects: A case study, Industrial Management & Data Systems, vol. 107, p. 284-303. DOI: 10.1108/02635570710723859

[6] Gallivan, M. J. 1998. The influence of system developers' creative style on their attitudes toward and assimilation of a software process innovation. Thirty-First Hawaii International Conference on System Sciences, 6-9 January, Kohala Coast, 435-444.

[7] Sawyer, S., Guinan P. J. 1998. Software development: Processes and performance, IBM Systems Journal, vol. 37, p. 552-569. DOI: 10.1147/sj.374.0552

[8] Hall, T., Wilson, D., Rainer, A., Jagielska, D. 2007. The neglected technical skill? ACM SIGMIS CPR Conference on Computer Personnel Research: The Global Information Technology Workforce, 19-21 April, St. Louis Missouri, 196-202.

[9] Baggelaar, H. 2008. Evaluating programmer performance visualizing the impact of programmers on project goals. M.Sc. Thesis, University of Amsterdam.

[10] Lee, K., Joshi, K., Kim, Y. 2008. Person-job fit as a moderator of the relationship between emotional intelligence and job performance. ACM SIGMIS CPR Conference on Computer Personnel Doctoral

Consortium and Research, 3-5 April, Charlottesville VA, 70-75.

[11] Thing, C. 2008. The application of the function point analysis in software developers' performance evaluation. 4th International Conference on Wireless Communications, Networking and Mobile Computing, 12-17 October, China, 1-4.

[12] Zhang, S., Wang, Y., Xiao, J. 2008. Mining individual performance indicators in collaborative development using software repositories. 15th Asia-Pacific Software Engineering Conference, 3-5 December, China, 247-254.

[13] Calikli, G., Bener, A. 2010. Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance. 6th International Conference on Predictive Models in Software Engineering, 12-13 September, Romania, no. 10.

[14] Chilton, M. A., Hardgrave, B. C., Armstrong, D. J. 2010. Performance and strain levels of it workers engaged in rapidly changing environments: A person-job fit perspective, ACM SIGMIS Database, vol. 41, p. 8-35. DOI: 10.1145/1719051.1719053

[15] Ramler, R., Klammer, C., Natschläger, T. 2010. The usual suspects: A case study on delivered defects per developer. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 16-17 September, Italy, no. 48.

[16] Wang, Y., Zhang, M. 2010. Penalty policies in professional software development practice: A multi-method field study. 32nd ACM/IEEE International Conference on Software Engineering, 1 May, Cape Town South Africa, 39-47.

[17] Balijepally, V., Nerur, S., Mahapatra, R. 2012. Effect of task mental models on software developer's performance: An experimental investigation. 45th Hawaii International Conference on System Science, 4-7 January, Hawaii, 5442-5451.

[18] Duarte, C. B., Faria, J. P., Raza, M. 2012. PSP PAIR: Automated personal software process performance analysis and improvement recommendation. Eighth International Conference on the Quality of Information and Communications Technology, 3-6 September, Portugal, 131-136.

[19] Ehrlich, K., Cataldo, M. 2012. All-for-one and one-for-all?: A multi-level analysis of communication patterns and individual performance in geographically distributed software development. ACM 2012 Conference on Computer Supported Cooperative Work, 11-15 February, Washington, 945-954.

[20] Kelly, B., Haddad, H. M. 2012. Metric techniques for maintenance programmers in a maintenance ticket environment, Journal of Computing Sciences in Colleges, vol. 28, p. 170-178. DOI: 10.5555/2382887.2382915

[21] Schröter, A., Aranda, J., Damian, D., Kwan, I. 2012. To talk or not to talk: Factors that influence communication around changesets. ACM 2012 Conference on Computer Supported Cooperative Work, 11-15 February, Washington, 1317-1326.

[22] Westermann, D. 2012. A generic methodology to derive domain-specific performance feedback for developers. 34th International Conference on Software Engineering, 2-9 June, Zurich, 1527-1530.

[23] Calikli, G., Bener, A. 2013. An algorithmic approach to missing data problem in modeling human aspects in software development. 9th International Conference on Predictive Models in Software Engineering, 9 October, Baltimore Maryland, no. 10.

[24] Kumar, C., Yadav, D. K. 2015. A probabilistic software risk assessment and estimation model for software projects, Procedia Computer Science, vol. 54, p. 353-361. DOI: 10.1016/j.procs.2015.06.041

[25] Lyytinen, K., Mathiassen, L., Ropponen, J. 1996. A framework for software risk management, Journal of Information Technology, vol. 11, p. 275-285. DOI:10.1057/jit.1996.2

[26] Ross, T. J. 2016. Fuzzy Logic with Engineering Applications. 4th edition. John Wiley & Sons Inc., United Kingdom, 580p.

[27] Hájek, P. 1998. Metamathematics of Fuzzy Logic. TREN, vol. 4, Kluwer Academic Publishers, Springer, Dordrecht, 299p.

[28] Carlsson, C., Fuller, R. 2003. A fuzzy approach to real option valuation, Fuzzy Sets and Systems, vol. 139 p. 297-312. DOI: 10.1016/S0165-0114(02)00591-2

[29] Shen, Q., Chouchoulas, A. 2002. A rough-fuzzy approach for generating classification rules, Pattern Recognition, vol. 35, p. 2425-2438. DOI: 10.1016/S0031-3203(01)00229-1

[30] Boehm, B. W. 1989. Software Risk Management. IEEE Press, New York, 455p.

[31] Boehm, B. W. 1991. Software risk management: principles and practices, IEEE Software, vol. 8, p. 32-41. DOI: 10.1109/52.62930