DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

RISK ASSESSMENT IN SOFTWARE DEVELOPMENT BY FUZZY INFERENCE APPROACH

by
Mustafa BATAR

July, 2021
İZMİR

RISK ASSESSMENT
IN SOFTWARE DEVELOPMENT
BY FUZZY INFERENCE APPROACH

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Engineering

by
Mustafa BATAR

July, 2021
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"RISK ASSESSMENT IN SOFTWARE DEVELOPMENT BY FUZZY INFERENCE APPROACH"** completed by **MUSTAFA BATAR** under supervision of **ASSIST. PROF. DR. KÖKTEN ULAŞ BİRANT** and **ASSOC. PROF. DR. ALİ HAKAN IŞIK** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Assist. Prof. Dr. Kökten Ulaş BİRANT

Supervisor

---	---
Prof. Dr. Hasan SELİM	Assoc. Prof. Dr. Semih UTKU
Thesis Committee Member	Thesis Committee Member
---	---
Assoc. Prof. Dr. İsmail KIRBAŞ	Assoc. Prof. Dr. Emre ÇOMAK
Examining Committee Member	Examining Committee Member

Prof. Dr. Özgür ÖZÇELİK

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank all who helped me to make and do this study. In particular, I want to express gratefulness to the following ones:

Firstly, I should thank to the greatest Turkish revolutionist and anti-imperialist leader who is the founder of Republic of Turkey – Ghazi M. Kemal ATATÜRK, for having education in the light of science in the universities. He is my pathfinder.

To my parents – Ertuğrul BATAR & Şerife BATAR, for their support and for teaching me to see, to read, to learn and to think. They are the meaning of my life.

To my wife – Ayşe ŞİMŞEK BATAR, for her brilliant comments, suggestions and making me encouraged during the development of the thesis. If she did not be with me, I could not complete my thesis. She is my heart to live. She is everything of me.

To my first supervisor, Asst. Prof. Dr. Kökten Ulaş BİRANT, for his guidance and contributions during my studies and trust in my work. During the process of writing my thesis, he was sometimes a professor of me, sometimes a brother of me, and sometimes a friend of me as well as always, he became fair, and every time treated fairly to me. He is the one who affects my life positively in every sense.

To my second supervisor, Assoc. Prof. Dr. Ali Hakan IŞIK, for his contributions, his mainstreaming and useful suggestions through this study.

To my thesis monitoring committee, Prof. Dr. Hasan SELİM and Assoc. Prof. Dr. Semih UTKU, for their insightful comments and encouragement about my thesis work.

Lastly and most morally, to my football team (club), Beşiktaş – also known as Karakartal, for teaching me to having conscience, and to live honorably and honestly by this motto: “Play with Honour, Win Rightly.” It is the metaphor in my life.

Mustafa BATAR

RISK ASSESSMENT IN SOFTWARE DEVELOPMENT BY FUZZY INFERENCE APPROACH

ABSTRACT

Software projects have a huge money related weight and need to put resources into high volumes. When seen costs dependent on the global substantial information on PC programming; it was $150 billion out of 1985, it was $2 trillion out of 2010, and it ignored $5 trillion after 2016. Additionally, in the time of 2018, simply a day by day giro of Apple Store was about $250 million. In spite of the costs and ventures that are dramatically expanding each year, the phase of effective advancement of the product ventures isn't high. In light of the "CHAOS" report arranged in 2016, just 17% of the product improvement ventures were finished in an opportune way, in the allotted financial plan and as per the necessities. 53% of the ventures were finished over the long run or potentially over spending plan as well as additionally without satisfying the prerequisites precisely. 30% of the product ventures can't have been finished in the improvement stage. For that software development projects with such high expenses and low success rate could have a better quality structure, a risk assessment and management approach has to be determined for better software risk assessment and management methodology. In this doctoral thesis, a rule set, which has consisted of 32 original software risk parameters rules, about software risk assessment and management has been shown, and how to determine, get, design and develop has been explained step by step based on the fuzzy approach technique integrated with machine learning algorithm – ANFIS.

Keywords: Software engineering, software risk assessment, software risk management, machine learning, fuzzy approach, ANFIS

YAZILIM GELİŞTİRMEDE BULANIK ÇIKARIM YAKLAŞIMI YOLUYLA RİSK DEĞERLENDİRMESİ

ÖZ

Yazılım geliştirme projeleri maddi olarak büyük bir götürüye sahiptir ve yüksek miktarlarda yatırıma ihtiyaç duymaktadır. Bilgisayar yazılımı ile ilgili uluslararası somut verilere dayanan maliyetlere bakıldığı zaman; 1985 yılında 150 milyar dolardır, 2010 yılında 2 trilyon dolardır ve 2016 yılından sonra da 5 trilyon doları geçmektedir. Buna ek olarak, 2018 yılında, Apple Store'un bir günlük cirosu bile yaklaşık 250 milyon dolardır. Her yıl katlanarak artan bu maliyetlere, harcamalara ve yatırımlara rağmen; yazılım projelerinin başarılı bir şekilde geliştirilme oranı çok da yüksek değildir. 2016 yılında hazırlanan "CHAOS" raporuna göre, geliştirilen yazılım projelerinin sadece %17'si tam zamanında, ayrılan bütçede ve verilen isteklere uygun bir şekilde tamamlanmıştır. Projelerin %53'ü ise süre ve bütçe aşımıyla ve ayrıca gereksinimleri tam doğru biçimde karşılamadan bitirilmiştir. Yazılım projelerinin %30'u ise, geliştirilme evresinde tamamlanamayıp iptal edilmiştir. Bu kadar yüksek maliyetli ve bu denli düşük başarı oranına sahip yazılım geliştirme projelerinin daha kaliteli ve daha başarılı bir yapıya sahip olabilmesi için, daha iyi bir yazılım risk değerlendirmesi ve yönetim metodu elde edilerek etkin bir risk değerlendirme ve yönetim yaklaşımının belirlenmesi gerekir. Bu etkin yaklaşımın bir sonucu olarak, yazılım geliştirme projelerinde sorunlar daha ortaya çıkmadan önce, yazılım riskleri oluşturabilecek bazı problemler zamanında fark edilebilip tespit edilebilecektir. Bu doktora tez çalışmasında, ANFIS makine öğrenmesi yöntemi ile birlikte, bulanık mantık yaklaşım tekniğine bağlı olarak yazılım risk değerlendirmesi ve yönetimi ile ilgili 32 yazılım risk kuralını içeren yeni ve orijinal bir kural seti oluşturulmuştur. Ayrıca, bu kural setinin nasıl elde edildiği gösterilmiş ve geliştirme bölümünün tüm evreleri adım adım detaylı bir şekilde anlatılmıştır.

Anahtar kelimeler: Yazılım mühendisliği, yazılım risk değerlendirmesi, yazılım risk yönetimi, makine öğrenmesi, bulanık yaklaşım, ANFIS

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1.1 General Information about Risks

Risk is the probability of not arriving at a designated result, and furthermore the likelihood of any occasion would keep an association from accomplishing its vital, monetary and functional destinations. Moreover, hazard has fundamental two parameters: the chance of event – the probability of not accomplishing a specific outcome or the probability of an undesired event –, size of misfortune – the impacts of the results which emerge where the dangers were figured it out or not (Smith, 1989).

Dangers and hazards come in all sizes and shapes. Also, hazard experts by and large perceive three significant sorts. Market hazard is the danger that costs will move in a manner that has adverse outcomes of an organization; credit hazard is the danger that a client, a counterparty or a provider will neglect to meet its commitments; and functional danger is the danger which individuals, cycles or frameworks will fall flat or that an outer occasion (tremor, fire, and so forth) will contrarily affect the organization (Lezzoni, 1997).

There are three basic risk categories. Internal risks about the company: risks related to production management's effectiveness, risks about financial management activity, risks about the effectiveness of marketing management, risks about in-house logistics, risks about quality management's effectiveness, risks about 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 (Renn, 2004).

1.2 General Information about Software Risks

Building and keeping up software can be a hazardous business. Most endeavours depend on 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 the executives: 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 (Arnuphaptrairong, 2011).

1.3 Purpose of the Thesis

Application based software development projects provide information and knowledge to support and maintain operations, management analysis, decision making processes and mechanisms in an organization or an administration. However, they are highly vulnerable against time and cost imbalance and also against quality measures of success that have perhaps relatively less importance. In addition, the presence of high level critical and number of errors is really important and crucial at early period of experimental and commercial use of software projects. Although managers allege that they can manage the issues, which they are experiencing, very efficiently, deficiencies and hurdles in software development management are indicated by

leading and expert software developers' subjective evaluation from all over the world (Efe & Demirors, 2019).

Programming improvement projects experience the ill effects of and furthermore are available to numerous dangers, particularly market chances, monetary dangers and specialized dangers. Programming designers need to give positive and particularly palatable responses to a few huge inquiries to make progress – achievement: if the created programming satisfies the requests and the prerequisites of the clients or not and gives the essential answers for them or not, how much contest the created programming can face, and adapt to, regardless of if the advantages and the additions got by the product project surpass the improvement cost or not, whether the undertaking to be created can be executed in fact and innovatively or not, if the equipment, the product, the fundamental organization associations and the whole sub-construction will fill in as arranged appropriately at the task to be created or not, whether the current innovation can meet the destinations and yields of the product project or not, is there a likelihood that the innovation of the current innovation become old and old prior to beginning to utilize the product to be created, if the security and insurance framework will work effectively during the product project life cycle or not, and so on. In addition, the issues referenced here have emerged and distinguished by the examples of bombed programming projects in the writing and these demonstrate that the dangers can't be overseen successfully (Dey, et al., 2007).

Administrators by and large arrangement with specialized dangers right off the bat in programming advancement projects, yet they push the dangers under market, monetary and a few different headings to the subsequent arrangement. Nonetheless, these dangers are significant for fruitful programming advancement, which is critical to such an extent that it can't be overlooked. Due to this explanation, there is a requirement for an incorporated danger the board that covers all means in the product advancement measure and furthermore addresses, examinations and assesses every one of the dangers that may emerge in the venture lifecycle (Yu, 2009).

In this doctoral dissertation, it has been designed and developed an operating-state risk assessment and management method for software projects based on the concrete and tangible data (original software risk parameters rule set) in the light of literature review study, machine learning, fuzzy structure and software experts. Also, in this study, rule-based model for software risk assessment has been developed with contribution of new and original software risk rules and parameters by Artificial Intelligence. According to this model, it has been seen that fuzzy structured machine learning algorithm – ANFIS – can create software risk rules without any human factor.

1.4 Main Contributions of the Thesis

In this doctoral dissertation, nine software risk assessment and management approaches, techniques and methods – Fault Tree Analysis, Decision Tree, Probabilistic Risk Analysis, Event Tree, Failure Modes and Effects Analysis, Hazard Analysis Critical Control Point, Root Cause Analysis, Risk Ranking and Filtering, and Hierarchical Holographic Modeling – have been introduced and explained in detail besides the key factor – fuzzy approach. In addition, that this fuzzy technique based on artificial intelligence and machine learning – including Adaptive Neuro-Fuzzy Inference System (ANFIS) comprised of fuzzy logic and Artificial Neural Networks (ANN) – is more useful and more effective for software risk evaluation in comparison with the others has been shown in detail by a demo, a real application and a survey.

The demo has been developed and designed with contribution of a 15-software-risk-parameter set. Moreover, this set has consisted of 50 software risk parameters created by the contribution of literature review. In this review, there have been 50 science citation indexed articles from 1978 to 2017 for project risks; 21 science citation indexed articles from 1998 to 2017 for software project risks; and 17 science citation indexed articles from 2001 to 2017 for software project risks based on developers' performance. In addition, 18 original linguistic risk rules have been created and written based on the selected 15-software-risk-parameter for the demo. Also, this demo has been developed by Python programming language (for user interface design) and MATLAB (for fuzzy logic structure, operation and development).

The genuine application has been planned and created dependent on 23 programming hazard boundaries and 36 principles from an IEEE article distributed in 2010 (Lamersdorf et al., 2010). Moreover, this standard set has been changed over into mathematical qualities for the execution of ANFIS. A while later, this numeric set has been stacked into MATLAB for ANFIS design. Besides, in light of ANFIS technique, 32 new principles have been made from the standard set close by. The present circumstance has expressed that this execution has given a few unique yields dependent on the info rules by learning itself, thus, it has been come about that machines can give a few guidelines not to seen, perceived and told by the people.

The survey has been prepared in order to show subjective accuracy ratings of 32 automatically derivated (developed) new rules (explained in the paragraph above) among software companies in real world. Furthermore, this survey has been applied with the help of Google Forms on internet. Also, it has been has been conducted by 108 different software experts from 76 different software companies (start-up and small companies, big and corporate companies, public enterprises, private institutions, national companies, international companies, universities, etc.). In the light of this survey, the developed rules' validity and accuracy levels and percentages have been determined and found out according to the software experts' opinions.

1.5 Organization of the Thesis

This study includes seven main chapters and the remaining of the thesis has been organized as follows:

In Chapter 2, literature review about "software risks", "software risk parameters", "software risk assessment", "software risk management", "software risk assessment and management system" and some important artificial intelligence methods based on fuzzy approach (especially, ANFIS) have been given in detail.

In Chapter 3, detailed information about "software", "software projects", "software management", "software project management", "software risk assessment", "software

risk management" and several "software risk assessment and management techniques" – Fault Tree Analysis, Decision Tree, Probabilistic Risk Analysis, Event Tree, Failure Modes and Effects Analysis, Hazard Analysis Critical Control Point, Root Cause Analysis, Risk Ranking and Filtering, and Hierarchical Holographic Modeling – have been given.

In Chapter 4, the original software risk parameters set has been explained in detail, and also, a demo application based on fuzzy logic method of these parameters has been given and told by showing related captures from the implementation and development.

In Chapter 5, it has been explained in detail how new software risks rule set have been formed and how to be designed and developed. In addition, it has been shown step by step how to derivate and generate an original new rule set based on Adaptive Neuro-Fuzzy Inference System (ANFIS) machine learning method on MATLAB with high accuracy rate.

In Chapter 6, a survey study about the subjective evaluation of the developed rules (32 new software risk parameters rules in total) has been explained and analysed according to the software experts' opinions in order to show their subjective validation.

In Chapter 7, the conclusion remarks of the work have been shown, and the future works of the study have been given.

CHAPTER TWO
LITERATURE REVIEW

The organization of endeavors isn't certainly a silly issue. A ton of parameters, both quantitative and abstract, is imperative in order to guarantee adventure accomplishment or further develop the errand execution. Point of fact, dangers basically infest a whole life example of an undertaking, and its most prominent wellsprings of information are the weaknesses. The expression "hazard" gets from the early Italian "risicare", which suggests "to dare" (Gerrard & Thompson, 2002). As a science, the risk was considered in sixteenth century, during the Renaissance, essentially setting up the systems of probability speculation (Hall, 1998).

A couple of makers, expressly in Software Engineering, focused into the perception that danger measure is identified with unpleasant factor measures (Hall, 1998), using as limits the probability of disaster and the outcomes if there ought to be an event of setback, where the thing is called risk show. Pfleeger et al. (2001) and Boehm (1989) prescribe that three points are identified with programming peril: the hardship identified with event, the probability of occasion of the event, and the amount that can change the results of the event, being described as threat show that the aftereffect of the likelihood by the adversity.

Threat the leaders is the utilization of aptitudes, data, instruments, and strategies to diminish perils to a satisfactory level while intensifying openings (Heldman, 2010). Besides, the threat the leaders in programming adventures is the demonstration of assessment and control of risks that impact errands, cycles, and consequences of programming (Hall, 1998). A critical feature be considered in programming adventures is the correspondence, especially of particular risks, that are routinely known at this point, inadequately bestowed (Carr et al., 1993). According to Barry Boehm, pondered the father of threat the leaders in programming adventures, "hazard organization is critical extraordinarily considering the way that it makes people avoid cataclysms, change, and fixing of endeavors, and helps with invigorating a condition of achievement in programming projects" (Boehm, 1989).

The greater part of the investigates announced in writings about programming hazard the executives center around the recognizable proof and examination of danger – i.e., the evaluation of dangers (Boehm, 1991; DoD, 2006; Dorofee et al., 1996; Fairley, 1994; Hall, 1998; Kontio, 2001; Stoneburner et al., 2003; Van Loon, 2007). Notwithstanding, as per Bannerman (2015), hardly any examinations center around the reception of danger the executives in programming improvement ventures, because there is a mindfulness with respect to the use of its practice, but it is ineffectively carried out and applied.

In the most recent decades, a few works identified with the utilization of computerized reasoning strategies in hazard evaluation have showed up, for example, dark relationship (Qinghua, 2009), probabilistic terms (Fu et al., 2012), fluffy hypothesis (Salmeron & Lopez, 2012; Tang & Wang, 2010), Bayesian networks (Fan & Yu, 2004), and case-based thinking (Trigo et al., 2008). The majority of these methods use hazard factors as a wellspring of data to foresee chances. The zone of danger the executives in programming ventures, even thought to be significant, actually needs a few advances, both in research and practically speaking. Different examinations show that the unequivocal act of danger the executives adds to execution improvement and expands the achievement of activities (De Bakker et al., 2010; Han & Huang, 2007; Jiang & Klein, 2000; Jiang et al., 2001; Raz et al., 2002; Wallace & Keil, 2004; Wallace et al., 2004a; Wallace et al., 2004b).

Generally, hazard factors are the source and reason for each peril (Brasiliano, 2009; Silva, 2011). Also, these are the inward and outer elements which potentially impact or contribute the event of the danger. In this point of view, the danger is the entirety of a lot of danger factors that whenever appeared, comprise a risk. Accordingly, hazard factors add to decreasing the reflection level of a danger. For instance, a specific association creates programming without following a turn of events philosophy, or a coding standard, significantly less there are preparing approaches. It can be attested that these are terms that lead to the improve danger – the danger can be perceived as a lot of parameters which cause a specific risk.

A danger factor alone can't enough total an incentive in venture the board; it just makes sense when there is an away from of likelihood of event and the related effects and outcomes. Thus, the utilization of grouping of dangers is significant for better recognizable proof of dangers. Risk Breakdown Structure (RBS) – Figure 2.1 in the following – is an intriguing device, because it is a danger situated gathering, that composes in an organized way, groups, and characterizes the presentation of the recognized venture hazards, giving a progressive structure of possible danger parameters (Hillson, 2002). One known proposition of RBS for programming advancement ventures is the hazard scientific classification via Carr et al. (1993).

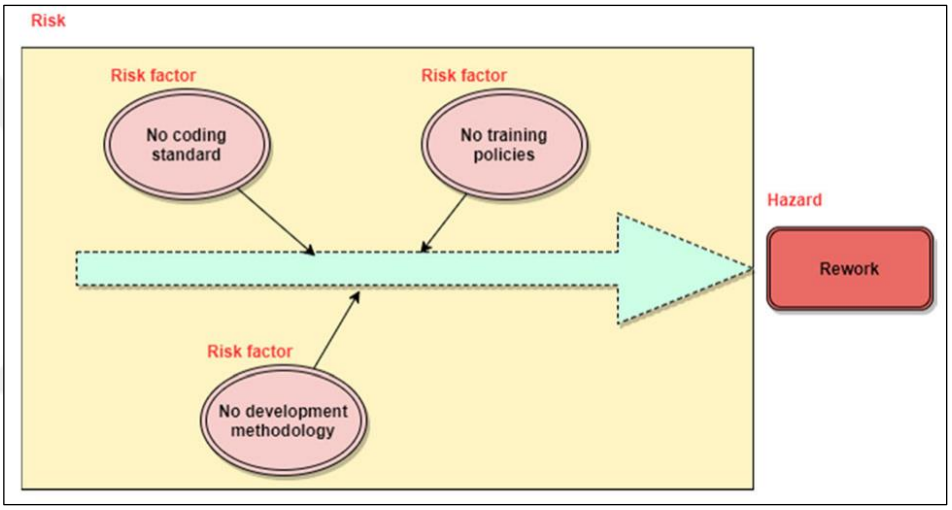

Figure 2.1 Risk mechanism (Hillson, 2002)

In banking and money space and area, Zanganeh (2011) exhibits viability of ANFIS in liquidation forecast on an informational collection comprising of budgetary proportions of non-bankrupt and bankrupt companies in Tehran. In addition, these utilized two methodology for choosing prescient factors; with no include determination technique and with T-measurement highlight determination technique. The investigation of observational outcomes demonstrates that ANFIS model beats Logistic Regression. In another examination Giovanis (2012) applied two methodologies for forecast of financial downturn or extension periods in USA; first including Logit and Probit models while the second using Adaptive Neuro-Fuzzy Inference System (ANFIS) with Gaussian and summed up ringer enrollment capacities. An informational collection comprising of tests between 1950 and 2006

shows that the ANFIS model beats the Logit and Probit model and henceforth demonstrates that neurofuzzy model gives a superior and more solid indications of monetary emergency.

Chen (2013) embraced molecule swarm improvement procedure to get proper boundary adjustments in subtractive grouping by incorporating the Adaptive Network based Fuzzy Induction System (ANFIS) model to anticipate business disappointments. Tests led into organizations recorded on the Taiwan Stock Exchange Corporation (TSEC) shows that the model is better than different models, with lower mean total rate blunder and root mean squared mistake. Moreover, their decision demonstrates a proposed mixture technique in order to give better outcomes about anticipating possible budgetary pain of an organization. As per Behbood (2011) an exact budgetary early admonition model is alluring for choice producers and controllers in any association having a place with monetary industry. They introduced a model for money related cautioning consolidating fluffy deduction framework in the learning capacity of neural organization to anticipate money related status of an association with Fuzzy Case-Based Reasoning (FCBR). A versatile fluffy standard base is produced to anticipate monetary status of target case. On the off chance that the case is anticipated to fall flat, the FCBR is utilized to discover comparative endure cases. The model encourages chief by giving money related choices to change specific highlights as organization objectives in forthcoming year to keep away from future monetary trouble.

Iraji et al. (2012), in their work, planned a savvy framework to isolate and order understudies as indicated by their learning furthermore, execution. The order is done through two techniques viz. LVQ organizations and ANFIS strategy. Both the techniques consider understudy's learning factor for the characterization measures. It is seen that ANFIS outflanks LVQ technique. The point of grouping understudies is to recognize grant understudies and to spot frail understudies or on the other hand liable to bomb understudies so appropriate coaching assets can be apportioned to them. The arrangement of understudies may be useful in arranging key projects for improving understudies' exhibition. Comparable idea can be found in approach for expectation

of understudies' exhibitions and grouping proposed by Inyang (2013). They encountered that graduates with less evaluations face troubles in getting work and henceforth the establishment needs to distinguish and bunch these understudies at a beginning phase during the examinations so that a presentation improvement plan can be created for them. Their strategy depends on mix of FCM, k-implies calculations and ANFIS. Creators introduced reasons for decision of every method and attempted to introduce a cross breed approach which demonstrates palatable outcomes to be appropriate for the expectation and bunching of understudies dependent on their execution level.

Considering comparative boundaries Catherine et al. (2013) planned a neuro-fluffy based model to decide the students' learning inclination. They utilized neuro-fluffy master thinking to assess student learning capacity comparative with the learning ideas measures and put away in the student profile during the association of the students with the framework. It is seen that the model could distinguish and order 96% into their individual learning inclinations while 4% couldn't be ordered effectively due to conflicting information reactions design from the students. The model is required to be utilized by the teacher to exhortation the student on the most proficient method to create frail learning style.

For enormous ventures, you should archive the results of the hazard the executives' procedure in a hazard the board plan. This ought to incorporate a conversation of the dangers looked by the task, an investigation of these dangers, and data on how you intend to deal with the hazard in the event that it appears to probably be an issue. The hazard the executives' procedure is an iterative procedure that proceeds all through a task. Hazard the board in dexterous improvement is less formal. A similar principal exercises should in any case be followed and hazards examined, despite the fact that these may not be officially archived. Light-footed improvement diminishes a few dangers, for example, dangers from prerequisites changes. Nonetheless, lithe advancement additionally has a drawback. Due to its dependence on individuals, staff turnover can affect the task, item, and business (McManus, 2003).

Gallivan (1998), in his examination, have shown the connection between the work and the expert calling about programming hazard appraisal and the board: fulfillment and trouble, the genuine (dynamic) execution, specialized information on the calling, insightful reasoning abilities, verbal abilities, work propensities, novel thoughts and innovativeness to open and uncovered different exceptional focuses. (18 critical programming hazards have been resolved.) Also, Sawyer, & Guinan (1998) have shown a few focuses to function as a product advancement group to decide and perceive programming chances. These issues have been group support, group dependability, group vision, group characters, group meeting, colleagues and group pioneer. Moreover, they have attempted to discover answers to certain inquiries regarding programming hazard appraisal and the executives. These inquiries; programming improvement strategy, code maintenance, code library, working time and identified with programming advancement documentation. (44 critical programming hazards have been resolved.)

Hall, Wilson, Rainer, & Jagielska (2007) have attempted to discover answers to a couple of inquiries regarding a couple of issues about programming improvement chances. These inquiries have been identified with programming group, programming project, business life, work and character. (26 critical programming hazards have been resolved.) Moreover, in applying programming hazard appraisal and the board, Baggelaar (2008) has underlined the significance of a few focuses in his lord postulation. These significant focuses have been deliberation, testability, coupling, measured quality, layouts, test inclusion, blunder taking care of and outstanding case use. What's more, programming engineers have attempted to discover the impact of code and remark line numbers in programming improvement measure. (22 huge programming hazards have been resolved.)

Lee, Joshi, & Kim (2008) have dissected and assessed programming hazard evaluation and the board as far as character and work propensities. (12 critical programming hazards have been resolved.) Furthermore, Thing (2008) has been keen on and zeroed in the terms of character, responsibility, working style, and

programming improvement measure about programming hazard appraisal and the board. (14 huge programming hazards have been resolved.)

Zhang, Wang, & Xiao (2008) first have posed a couple of inquiries in the work and got a few responses about programming hazards. In addition, these inquiries have been number of lines of code, number of remark lines, number of classes, number of tests, class connection, number of technique (strategies), level of heritability profundity and programming advancement issues identified with the trouble. (13 critical programming chances have been resolved.) Furthermore, Calikli, & Bener (2010) have given an overall outline of the product hazard evaluation and the board. In their work, they have showed the impact of programming designers' degree of training and a few focuses and issues about programming advancement (fulfillment level, certainty level, work insight, and so on) on programming hazard appraisal and the executives. (4 critical programming chances have been resolved.)

Chilton, Hardgrave, & Armstrong (2010) have advanced a few focuses for programming hazards. These focuses have been work-life, working propensities, character, age and sexual orientation. (22 huge programming hazards have been resolved.) Moreover, Ramler, Klammer, & Natschläger (2010) have attempted to research and discover answers to certain inquiries regarding programming quality in programming hazard appraisal and the executives. (3 huge programming chances have been resolved.) Also, Wang, & Zhang (2010) have featured various significant issues in the product hazard evaluation and the executives. These focuses have been work-life, work insight, responsibility, training level and sex. (14 huge programming chances have been resolved.)

Balijepally, Nerur, & Mahapatra (2012), in their examination, attempted to discover answers of numerous inquiries identified with character characteristics in perceiving programming hazards. (8 critical programming hazards have been resolved.) Likewise, Duarte, Faria, & Raza (2012) have attempted to discover the impacts of different issues in programming hazard appraisal and the executives. They have been timing blunder, size mistake, division mistake, missing parts, random parts, number of blunders and

the quantity of unit tests. (10 critical programming chances have been resolved.) Also, Ehrlich, & Cataldo (2012) have talked about certain parts of programming improvement to decide programming chances. These issues have been group pioneer, group coordination, organization the executives, organization workers and private life. (10 critical programming chances have been resolved.)

Kelly, & Haddad (2012) have attempted to discover the degree to how "blunder" has an effect into programming hazard appraisal and the board. (3 critical programming hazards have been resolved.) Furthermore, Schröter, Aranda, Damian, & Kwan (2012) have attempted to address different inquiries in the personalities about programming advancement chances. These inquiries have been identified with the quantity of builds, code changes, strategy (technique) number, work-life, fixed code parts, work quality, group pioneer, programming project reports and programming improvement device. (22 huge programming hazards have been resolved.) Also, Westermann (2012) has accentuated the significance of specific focuses in the product advancement measure. Dependable code composing has been researched in the effect of programming project yields and work style about perceiving programming chances. (7 critical programming hazards have been resolved.) Moreover, Calikli, & Bener (2013) have shown some significant focuses in programming hazard evaluation and the executives. These focuses; programming project improvement plan and programming group brain science. (4 huge programming hazards have been resolved.)

CHAPTER THREE
TERMS AND TECHNIQUES ABOUT SOFTWARE RISK MANAGEMENT

3.1 Software

PC programming, or simply writing computer programs, is an arrangement of data or PC rules that prompt the PC how to work. This is instead of actual gear, from which the system is gathered and truly plays out the work. In programming designing and programming building, PC writing computer programs is all information dealt with by PC structures, ventures and data. PC programming joins PC tasks, libraries and related non-executable data, for instance, online documentation or progressed media. PC gear and programming require each other and neither can be basically used isolated (Shapiro, 2000).

The greater part of writing computer programs is written in critical level programming vernaculars. They are easier and progressively beneficial for programming engineers since they are closer to standard tongues than machine dialects. Critical level tongues are changed over into machine language using a compiler or an interpreter or a mix of the two. Programming may moreover be written in a low-level low level registering develop, which has strong correspondence to the PC's machine language rules and is changed over into machine language using a building specialist (Powers & Turk, 2012).

In 2000, Fred Shapiro, a clerk at the Yale Graduate School, appropriated a letter revealing that John More staggering Tukey's 1958 paper "The Educating of Cement Mathematics" contained the soonest known usage of the articulation "programming" found in a chase of JSTOR's electronic annals, beginning before the OED's reference by two years. This drove various to recognize Tukey for composing the term, particularly in tributes disseminated that identical year, notwithstanding the way that Tukey never ensured credit for any such coinage. In 1995, Paul Niquette affirmed he had at first generated the term in October 1953, regardless of the way that he couldn't find any records supporting his case. The most reliable known circulation of the

articulation “programming” in a structure setting was in August 1953 by Richard R. Carhart, in a Rand Organization Exploration Memorandum.

3.2 Software Projects

Programming instruments are moreover programming as undertakings or applications that item planners (in any case called designers, coders, developers or programming engineers) use to make, investigate, keep up (for instance improve or fix), or regardless support programming. Writing computer programs is written in at any rate one programming tongues; there are many programming lingos in presence, and each at any rate one execution, all of which includes its own game plan of programming contraptions. These instruments may be tolerably autonomous tasks, for instance, debuggers, compilers, linkers, interpreters, and substance devices, which could be combined together in order to complete successfully an endeavour; or they may shape a fused improvement condition (IDE), which joins a ton or the whole of the value of such free devices. IDEs may do this by either conjuring the significant individual instruments or by re-completing their handiness in another way. An IDE can make it less difficult to do express endeavours, for instance, glancing in records in a particular errand. Many programming language executions give the decision of using both individual gadgets and an IDE (Rook, 1986).

PC programming must be “stacked” into the PC’s accumulating, (for instance, the hard drive or memory). At the point when the item has stacked, the PC can execute the item. This incorporates passing headings from the application programming, through the system programming, to the gear which at last gets the direction as machine code. Each direction causes the PC to do an action – moving data, finishing a count, or changing the control stream of bearings. Data improvement is regularly beginning with one spot in memory then onto the following. Now and again it incorporates moving data among memory and registers which enable quick data access in the CPU. Moving data, especially a great deal of it, can be costly. Thusly, this is sometimes kept up a vital good ways from by using “pointers” to data. Counts join direct assignments, for instance, expanding the assessment of a variable data segment. Dynamically complex

computations may incorporate various assignments and data segments together (Robles et al., 2005).

3.3 Software Management

The individuals working in a product association are its most noteworthy resources. It is costly to select and hold great individuals, and it is up to programming administrators to guarantee that the designers dealing with a venture are as beneficial as could be expected under the circumstances. In fruitful organizations and economies, this profitability is accomplished when individuals are regarded by the association and are allotted obligations that mirror their aptitudes and experience. It is significant that product venture directors comprehend the specialized issues that impact crafted by programming improvement. Tragically, be that as it may, great programming engineers are not in every case great individuals' chiefs. Programming engineers frequently have solid specialized aptitudes yet may do not have the milder abilities that empower them to persuade and lead a task advancement group. As a venture chief, you ought to know about the potential issues of individuals the board and should attempt to create individuals the executives' aptitudes. There are four basic factors that impact the connection between an administrator and the individuals that the individual in question oversees (Stellman & Greene, 2005) as given in the following.

Consistency: All the individuals in an undertaking group ought to be treated in a practically identical manner. Nobody anticipates that all prizes should be indistinguishable, however individuals ought not to feel that their commitment to the association is underestimated.

Regard: Different individuals have various abilities, and chiefs should regard these distinctions. All individuals from the group ought to be allowed a chance to make a commitment. Now and again, obviously, you will find that individuals just don't fit into a group and they can't proceed, yet it is significant not to form a hasty opinion about them at a beginning time in the task.

Consideration: People contribute adequately when they feel that others hear them out and assess their proposition. It is essential to build up a workplace where all perspectives, even those of the least experienced staff, are thought of.

Trustworthiness: As a chief, you ought to consistently speak the truth about what is working out in a good way and what is going seriously in the group. You ought to likewise speak the truth about your degree of specialized information and be eager to concede to set up with more information when vital. On the off chance that you attempt to conceal obliviousness or issues, you will in the end be discovered and will lose the regard of the gathering.

These requirements are organized in a progression of levels. The lower levels of this chain of command speak to central requirements for food, rest, etc., and the need to have a sense of safety in a domain. Social need is worried about the need to feel some portion of a social gathering. Regard need speaks to the need to feel regarded by others, and self-acknowledgment need is worried about self-awareness. Individuals need to fulfill lower-level needs, for example, hunger before the more dynamic, more significant level needs. Individuals working in programming improvement associations are not normally ravenous, parched, or truly compromised by their condition. Subsequently, ensuring that people groups' social, regard, and self-acknowledgment needs are fulfilled is generally significant from an administration perspective (Maslow, 1954) as explained in the following.

To fulfill social needs, you have to give individuals time to meet their colleagues and give spots to them to meet. Programming organizations, for example, Google give social space in their workplaces for individuals to get together. This is generally simple when the entirety of the individuals from an improvement cooperation in a similar spot, however, progressively, colleagues are not situated in a similar structure or even a similar town or state. They may work for various associations or from home more often than not. Person to person communication frameworks and video chatting can be utilized for remote correspondences, yet my involvement in these frameworks is that they are best when individuals definitely know one another. You ought to

mastermind some up close and personal gatherings right off the bat in the venture so individuals can legitimately connect with different individuals from the group. Through this immediate collaboration, individuals become some portion of a social gathering and acknowledge the objectives and needs of that gathering.

To fulfill regard needs, you have to show individuals that they are esteemed by the association. Open acknowledgment of accomplishments is a straightforward and powerful method of doing this. Clearly, individuals should likewise feel that they are paid at a level that mirrors their abilities and experience.

At long last, to fulfill self-acknowledgment needs, you have to give individuals duty regarding their work, appoint them requesting (however not feasible) undertakings, and give chances to preparing and improvement where individuals can upgrade their aptitudes. Preparing is a significant rousing impact as individuals like to increase new information and learn new abilities (Biswas et al., 2007). In addition, software management process has been demonstrated in Figure 3.1 in the following.

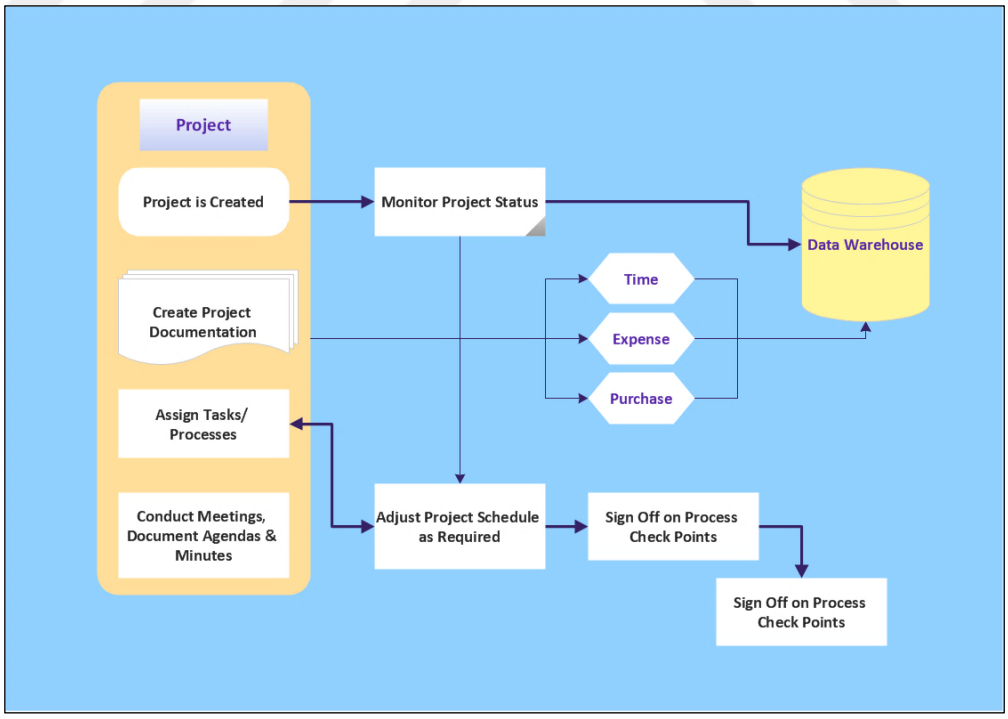

Figure 3.1 Software management process (Stellman & Greene, 2005)

3.4 Software Project Management

Programming venture the executives is a fundamental piece of programming building. Activities should be overseen in the fields that proficient programming building is consistently dependent upon hierarchical financial plan and timetable imperatives. The task director's responsibility is to guarantee that the product venture meets and conquers these limitations just as conveying top notch programming. Great administration can't ensure venture achievement. Be that as it may, awful administration for the most part brings about venture disappointment: The product might be conveyed late, cost more than initially evaluated, or neglect to meet the desires for clients. The achievement measures for venture the executives clearly differ from undertaking to extend, in any case, for most activities, significant objectives are: to convey the product to the client at the concurred time; to keep generally costs inside spending plan; to convey programming that meets the client's desires; to keep up a reasonable and well-working advancement group. These objectives are not exceptional to programming building but rather are the objectives of all building ventures. Notwithstanding, programming building is not the same as different kinds of designing in various manners that make programming the executives especially testing. A portion of these distinctions (Jurison, 1999) are introduced in the following.

The item is impalpable: A supervisor of a shipbuilding or a structural designing undertaking can see the item being created. On the off chance that a timetable slips, the impact on the item is noticeable – portions of the structure are clearly incomplete. Programming is impalpable. It can't be seen or contacted. Programming venture directors can't see improvement by taking a gander at the antique that is being developed. Or maybe, they depend on others to create proof that they can use to survey the advancement of the work.

Enormous programming ventures are regularly "one-off" ventures: Each huge programming improvement venture is extraordinary on the grounds that each condition where programming is created is, somehow or another, not the same as all others. Indeed, even supervisors who have a huge group of past experience may think

that it is hard to foresee issues. Besides, fast mechanical changes in PCs and correspondences can make experience old. Exercises gained from past activities may not be promptly transferable to new undertakings.

Programming forms are variable and association explicit: The designing procedure for certain kinds of framework, for example, scaffolds and structures, is surely known. Be that as it may, various organizations utilize very unique programming improvement forms. We can't dependably anticipate when a specific programming process is probably going to prompt advancement issues. This is particularly evident when the product venture is a piece of a more extensive frameworks building venture or when totally new programming is being created.

As a result of these issues, it isn't astonishing that some product ventures are late, over-budget, and delayed. Programming frameworks are frequently new, extremely intricate, and actually inventive. Calendar and cost invades are likewise basic in other designing activities, for example, new vehicle frameworks, that are unpredictable and creative. Given the challenges in question, it is maybe surprising that such a large number of programming ventures are conveyed on schedule and to financial plan. It is difficult to compose a standard set of working responsibilities for a product venture supervisor. The activity differs enormously relying upon the association and the product being created. Probably the most significant components that influence how programming ventures are overseen are (Liu & Horowitz, 1989).

3.5 Software Risk Assessment

Peril evaluation is a term used to depict the overall strategy or procedure where you: One perceives threats and danger factors that can cause hurt (hazard recognizing evidence). One separates and evaluates the peril related with that risk (chance 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 (chance control). A peril assessment is a cautious look at your workplace to perceive those things, conditions, structures, etc. that may cause hurt, particularly to people. After ID is made, you

explore and evaluate how likely and genuine the risk is. Right when this affirmation is made, you can immediately, pick what measures should be set up to effectively take out or control the underhandedness from happening. 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 (Kumar & Yadav, 2015).

3.6 Software Risk Management

Hazard the executives is one of the most significant occupations for a venture supervisor. You can think about a hazard as something that you’d 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 dangers (Lyytinen et al., 1996) as given in the following.

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 chances affect the execution or quality of the product being created. A case of an item chance is the disappointment of a bought segment to proceed true to form. Also, it might affect the total execution of the model so it 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 demonstration of a serious issue might see that the suppositions made about deals of existing programming items might be unduly idealistic.

Obviously, these hazard classifications cover. An accomplished specialist’s choice to leave a venture, for instance, presents an undertaking hazard on the grounds that the product conveyance calendar will be influenced. It definitely requires some investment for another task part to comprehend the work that has been done, so the individual in question can't be promptly profitable. Thus, the conveyance of the framework might be postponed. The departure of a colleague can likewise be an item hazard in light of the fact that a substitution may not be as experienced thus could make programming mistakes. At last, losing a colleague can be a business chance in light of the fact that an accomplished designer’s notoriety might be a basic factor in winning new agreements. For enormous tasks, you should record the aftereffects of the hazard examination in a hazard register alongside a result investigation. This sets out the results of the hazard for the venture, item, and business. Successful hazard the executives makes it simpler to adapt to issues and to guarantee that these don't prompt unsuitable financial plan or calendar slippage. For little undertakings, formal hazard recording may not be required, however the task administrator ought to know about them. The particular dangers that may influence a task rely upon the undertaking and the authoritative condition where the product is being created. Notwithstanding, there are additionally basic dangers that are autonomous of the kind of programming being created. These can happen in any product improvement venture. Programming hazard the executives is significant on account of the intrinsic vulnerabilities in programming advancement. These vulnerabilities come from inexactly characterized prerequisites, necessities changes because of changes in client needs, challenges in evaluating the time and assets required for programming improvement, and contrasts in singular aptitudes. You need to foresee dangers, comprehend their effect on the task, the item, and the business, and find a way to maintain a strategic distance from these dangers. You may need to draw up emergency courses of action so that, if the dangers do happen, you can make prompt recuperation move. A layout of the procedure of hazard

the executives (this layout has been showed in Figure 3.2 in the following) includes a few phases (Fairley, 1994) as explained in the following.

Hazard recognizable proof: You should distinguish conceivable venture, item, and business dangers.

Hazard examination: You ought to evaluate the probability and outcomes of these dangers.

Hazard arranging: You should make arrangements to address the hazard, either by staying away from it or by limiting its impacts on the task.

Hazard checking: You ought to normally evaluate the hazard and your arrangements for chance moderation and reconsider these plans when you become familiar with the hazard.

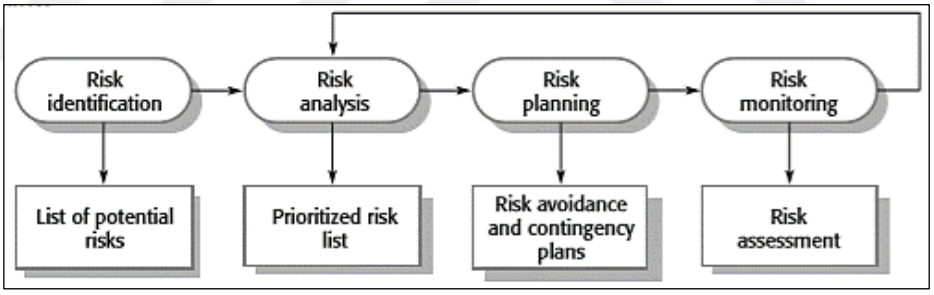

Figure 3.2 Risk management process (Lyytinen et al., 1996)

3.7 Several Software Risk Assessment and Management Techniques

The Fault Tree Analysis (FTA) was first introduced by Bell Laboratories and is a champion among the most by and large used systems in structure enduring quality, reasonability and prosperity assessment. It's anything but a deductive approach used to choose the various mixes of hardware and programming disillusionments and human goofs that could cause undesired events (suggested as best events) at the system level. The deductive assessment begins with an overall end, by then undertakings to choose the unequivocal explanations behind the choice by building a reasoning layout

called a fault tree. This is generally called embracing a best down methodology. The guideline inspiration driving the fault tree assessment is to assist with perceiving likely explanations behind system disillusionments before the failure truly occur. It can similarly be used to evaluate the probability of the best event using symptomatic or quantifiable systems. These checks incorporate system quantitative reliability and suitability information, for instance, dissatisfaction probability, frustration rate and fix rate. Resulting to completing a FTA – given in Figure 3.3 in the following –, you can focus your undertakings on upgrading structure security and faithful quality (Tanaka et al., 1983).

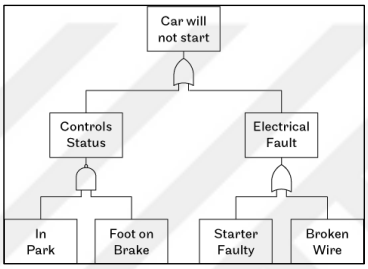

Figure 3.3 An example of fault tree analysis (Tanaka et al., 1983)

A Decision Tree is a choice assistance device that utilizes a tree-like model of choices and their potential results, including chance occasion results, asset expenses, and utility. It is one approach to manage show a figuring that simply contains restrictive control clarifications. Choice trees are regularly utilized in practices research, unequivocally in choice assessment, to assist with perceiving a method bound to achieve an objective, yet of course are an outstanding device in AI (Olaru & Wehenkel, 2003). Also, an example of decision tree has been indicated in Figure 3.4 in the following.

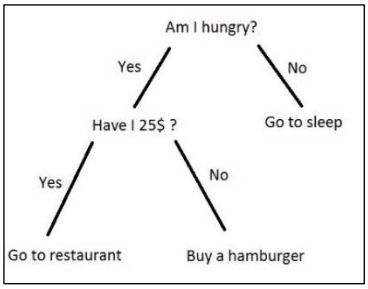

Figure 3.4 Tree like model for decision tree (Olaru & Wehenkel, 2003)

Probabilistic Risk Analysis (PRA) is precise and irrefutable assumptions for stream models and impurity transport over a broad assortment of environmental streams are broadly tedious because of intricacies introduced by savage streams, spatial calculations and typical weaknesses. The shortfall of sufficient site depiction, computational resources, and here and there, the lack of conceptualizations and logical models of critical actual miracles obfuscate exhibiting tries further. These issues present a main shortfall of feeling about any expected stream and cast inquiries on the attainability of procuring a single deterministic conjecture of a contamination circumstance (Bedford & Cooke, 2001).

An Event Tree is an inductive descriptive diagram wherein an event is bankrupt down using Boolean reasoning to take a gander at a successive game plan of resulting events or results. For example, event tree assessment is a vital portion of nuclear reactor security designing. An event tree shows progression development, course of action end states and gathering unequivocal conditions across after some time (Kenarangui, 1991). Moreover, an example of event tree has been showed in Figure 3.5 in the following.

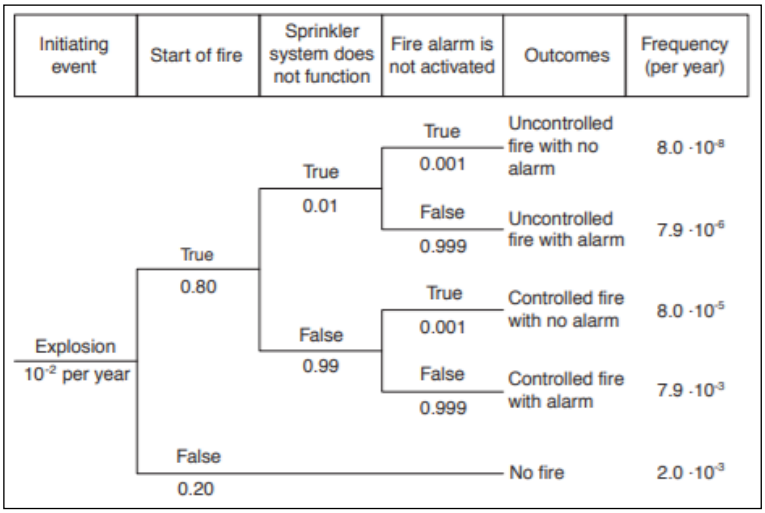

Figure 3.5 Forward procedure for event tree mechanism (Kenarangui, 1991)

Failure Modes and Effects Analysis (FMEA) is a little by little methodology for recognizing each and every possible dissatisfaction in an arrangement, a collecting or get together cycle or a thing or organization. “Disillusionment modes” implies the

ways or modes, where something may miss the mark. Dissatisfactions are any errors or deformations, especially ones that impact the customer and can be potential or certified. “Effects assessment” suggests thinking about the results of those mistake. Disillusionments are coordinated by how certified their results are, the methods by which routinely they occur and how viably they can be recognized. The purpose behind the FMEA is to take actions to clear out or decrease frustrations, starting with the most raised need ones (Gilchrist, 1993).

Hazard Analysis Critical Control Point (HACCP) is an organization system wherein food prosperity is tended to through the examination and control of natural, engineered, and actual dangers from rough material age, acquisition and managing, to collecting, allotment and use of the finished thing (Hulebak & Schlosser, 2002).

Root Cause Analysis (RCA) is a system for basic deduction used to perceive the fundamental issues. A parameter is seen as a fundamental driver if ejection thereof from the terms accuse of progression shields the last undesirable come about because of rehashing; whereas a causal parameter is one that impacts an event’s outcome, anyway isn’t a hidden driver. Despite the way that clearing a causal parameter could benefit an outcome, it doesn’t go on its rehash with sureness (Andersen & Fagerhaug, 2006). Furthermore, an example of root cause has been demonstrated in Figure 3.6 in the following.

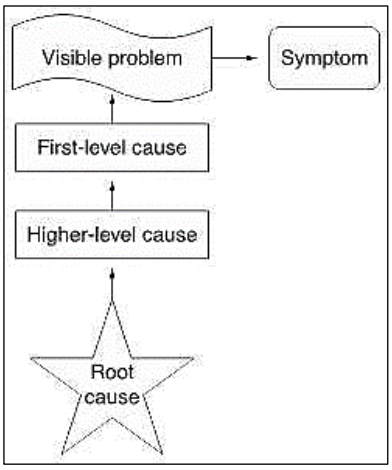

Figure 3.6 Effect of root cause (Andersen & Fagerhaug, 2006)

Risk Ranking and Filtering is a champion among the most generally perceived help methodologies used for Risk Management. This procedure is generally called “Relative Risk Ranking”, “Danger Indexing” and “Peril Matrix and Filtering”. Its will likely give more sharpened focus to the fundamental perils inside a structure – ordinarily, from an immense and complex game plan of danger circumstances. Peril Ranking and Filtering works by isolating overall risk into danger portions and surveying those fragments and their individual duties to as a rule threat (Ezell et al., 2000).

The Center for Risk Management of Engineering Systems at the University of Virginia developed the procedure of Hierarchical Holographic Modeling (HHM) during the 1980s as a peril examination instrument. Planned to discover the stunning interdependencies of essential establishments, HHM was made to overcome the trouble of a single model's ability to address every one of the huge pieces of an erratic structure. In showing colossal extension, complex systems, various satisfactory mathematical and hypothetical models emerge, each addressing a substitute, yet precise viewpoint. The utilization of HHM grants the progression of various, generous models that "get the exemplification of the various estimations, dreams, and perspectives of establishment structures." Hierarchical recommends the need to understand chances as they show up at various estimations in a chain of command (e.g., chances at the approach of designs level, the individual framework level, the sub-structure level and the bit level). Holographic recommends the need to will risk according to alternate points of view (e.g., financial, security, geographic, specific, and so forth) The showing approach is sorted out to regulate both holographic and distinctive evened out contemplations and uses a formalized procedure (Risk Filtering and Ranking Methodology) to mastermind dangers (Haimes, 1981). In addition, an example of hierarchical holographic modeling has been given in Figure 3.7 in the following.

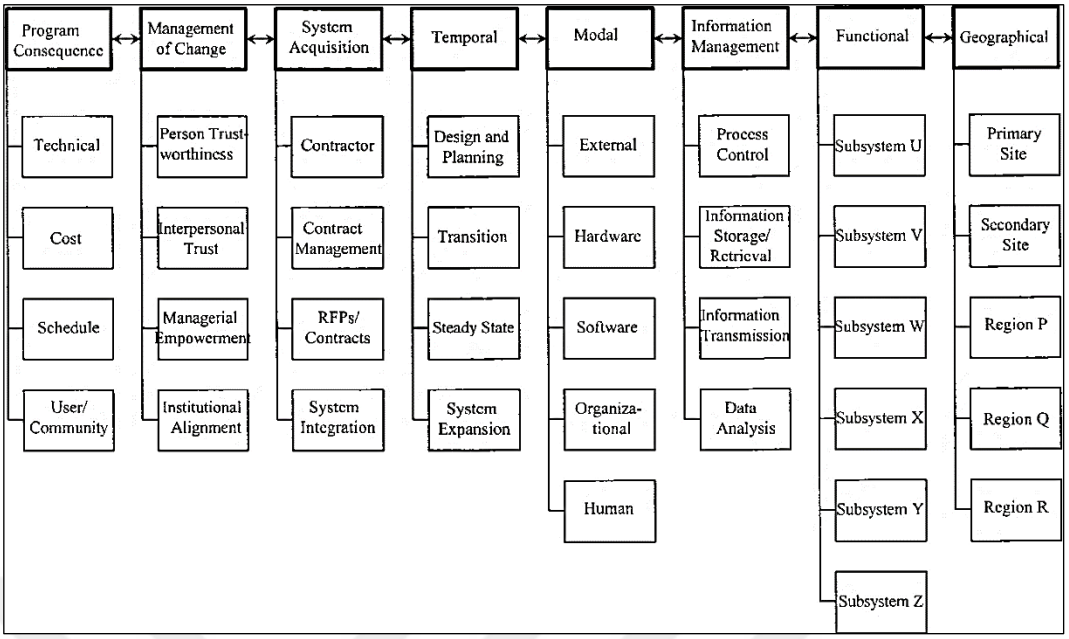

Figure 3.7 Structure of hierarchical holographic modeling (Haimes, 1981)

CHAPTER FOUR
SOFTWARE RISK PARAMETERS SET AND ITS IMPLEMENTATION

Programming hazards are fundamental for adventures. Programming improvement adventures are not particular as attempted masterminding is done with least information. Regardless, the degree of peril changes with unconventionality, size and region. Degree creep, nonappearance of appreciation of issues, dubious necessities and nonattendance of resources, hardware, frameworks organization and security terms are a part of the essential peril segments in programming progression adventures. Thusly, there is a need to manage threat about programming improvement. Despite of the way that researchers and master have made on peril the leaders about programming improvement, close to work have been done to remember every one of the concerned accomplices for supervising risk and joining the threat the board cycle with a widely inclusive approach. Various frameworks of programming peril the leaders have been proposed; some of that oversee only the application part and a couple of models oversee risk completely covering application, affiliation and between affiliation levels. Regardless, there is a deficiency of composing in programming improvement peril the board from developers' perspective (Coleman, 2002).

4.1 Preparation of The Risk Set

In order to prepare an original software risk parameters set, a literature review study has been done for about a term. This review has been done under three main subjects: project risks, software project risks and software project risks based on software developer's performance. In addition, for project risks, 50 science citation indexed articles from 1978 to 2017 have been analysed; for software project risks, 21 science citation indexed articles from 1998 to 2017 have been examined; and for software project risks based on developers' performance, 17 science citation indexed articles from 2001 to 2017 have been viewed and studied. In the result of this study and investigation, 13 main articles attracted to the attention for forming and creating a software risk parameters set. These articles have been listed in the Table 4.1 in detail in the following.

Table 4.1 The main articles for software risk parameters set

No	Published Year	Database	The Article
1	1998	ACM	Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. *A framework for identifying software project risks.*
2	2004	ACM	Wallace, L., & Keil, M. *Software project risks and their effect on outcomes.*
3	2007	emerald insight	Dey, P. K., Kinch, J., & Ogunlana, S. O. *Managing risk in software development projects: A case study.*
4	2009	ACM	Warkentin, M., Moore, R. S., Bekkering, E., & Johnston, A. C. *Analysis of systems development project risks: An integrative framework.*
5	2009	IEEE	Xiaosong, L., Shushi, L., Wengun, C., & Songjiang, F. *The application of risk matrix to software project risk management.*
6	2009	IEEE	Feng, N., Li, M., & Gao, H. *A software project risk analysis model based on evidential reasoning approach.*
7	2009	IEEE	Khan, S. *An approach to facilitate software risk identification.*
8	2010	IEEE	Lamersdorf, A., Munch, J., Torre, A. F., Sánchez, C. R., Heinz, M., & Rombach, D. *A rule-based model for customized risk identification in distributed software development projects.*
9	2011	ACM	Becker, J., & Dai, R. W. *Understanding IT project risks as disturbances to digital ecosystems.*
10	2012	IEEE	Hoermann, S., Aust, M., Schermann, M., & Krcmar, H. *Comparing risks in individual software development and standard software implementation projects: A delphi study.*
11	2013	IEEE	Fakhar, M. Z., Abbas, M., & Waris, M. *Risk management system for ERP software project.*
12	2014	IEEE	Sonchan, P., & Ramingwong, S. *Top twenty risks in software projects: A content analysis and Delphi study.*
13	2017	ACM	Min, W., Jun, Z., & Wei, Z. *The application of fuzzy comprehensive evaluation method in the software project risk assessment.*

Based on these articles listed in the table above (Table 4.1), a software risk parameters set – 50 different parameters in total - have been built under 6 main titles: Organizational Environment (4 parameters), Users (5 parameters), Requirements (12 parameters), Project Complexity (9 parameters), Team (6 parameters), and Planning and Control (14 parameters). These titles and their parameters have been listed and written in the Table 4.2 in the following.

Table 4.2 The unique software risk parameters set

Titles	Software Risk Parameters
Organizational Environment	Change in organizational management during the project
	Corporate politics with negative effect on project
	Unstable organizational environment
	Organization undergoing restructuring during the project
Users	Lack of adequate user involvement
	Failure to identify all stakeholders
	Lack of cooperation from users
	Lack of appropriate experience of the user representative
	User resistance
Requirements	Lack of frozen system requirements
	Systems requirements not adequately identified
	Unclear or misunderstanding system requirements
	New and/or unfamiliar subject matter for the system
	Lack of analysis for change of requirements
	Change extension of requirements
	Lack of report for requirements
	Poor definition of requirements
	Ambiguity and/or change of requirements
	Unclear customer requirements
	Requirement creep
	Unable to meet user requirements
Project Complexity	Project involved the use of new technology
	High level of technical complexity
	Immature technology
	Project involves the use of technology that has not been used in prior
	Low software performance
	Problem with new technology

Table 4.2 continues

	Inadequate infrastructure
	Resource insufficiency
	Inadequate design
Team	Inexperienced team members
	Inadequately trained development team members
	Team members lack specialized skills required by the project
	Inefficient team capability
	Conflict among team members
	Staff turnover
Planning and Control	Lack of an effective project management methodology
	Project progress not monitored closely enough
	Not managing change properly
	Inadequate project planning
	Project milestones not clearly defined
	Lack of effective project management skills
	Ineffective communication
	Lack of control over consultants, vendors and subcontractors
	Inappropriate development process
	Unrealistic schedule
	Optimistic resource planning
	Lack of executive involvement
	Unrealistic budgeting
	Lack of law enforcement

4.2 A Demo Based on The Unique Risk Set with Fuzzy Logic Method

Software risk assessment and management might be characterized as the arranged and standard control of the issues and inconveniences that may happen while planning and building up the product. Thusly, this demo has been created by avoiding the significant expense of the improvement cycle and following the product life cycle in the most (to the extent) exact way. In this application, a product hazard appraisal and the board framework with coordinated fluffy methodology (fluffy rationale) has been arranged, planned and created. As a finish of this work, programming venture administrators might have the option to come to the most precise (to the extent and dependent on the decided danger boundaries set) result by utilizing and applying this

framework as opposed to managing long plans (burn-through basically parcel of time), and also, this framework may spare “man”, “time”, and “value”, which are driving the two sources of info and yields of programming advancement instrument hazard boundaries (Birant et al., 2019).

4.2.1 Fuzzy Logic Method

Fuzzy logic is conveyed like a philosophy reliant upon "levels of precision" rather than the "legitimate or sham" state which is the Boolean system. The chance of fleecy reasoning was first top tier by Dr. Lotfi Zadeh of the University of California at Berkeley during the 1960s. Feathery theory can be used for as a strategies for addressing irregularity about creating nonlinear relationship with heuristic data. The theory is basically about the reasoning that instead of enunciation being 0 or 1, its value may have a regard that can vary in this rang (Ross, 2017).

Cushy reasoning undertakings to show the general working reasoning of the PC with the end goal that people can understand inside the arrangement of reasoning. A PC's reasoning square gets altogether commitment from the customer and gives the yields TRUE or FALSE, which is equivalent to YES or NO results. According to the fleecy reasoning methodology the customer's decision communicates that there are different possible results between YES moreover, NO. Using cushioned reasoning procedure, it is intended to exhibit uncertain conditions, improperly described or complex systems (Carlsson & Fuller, 2003).

The designing of soft reasoning system contains three rule parts as showed up in the going with (as given Figure 4.1 in the following). At first, this converts over structure commitments to the cushioned sets gave in the fuzzification module. The norms region portrays the conditions that choose the results of the structure’s cushy reasoning methodology. These conditions show which explanation ought to be yield against the changing data enunciations of the system. Finally, the defuzzification module converts in the cushy set delivered by the inferring engine to a total assets.

Thusly, the structure yields give assorted yield regards as shown by the rules (Shen & Chouchoulas, 2002).

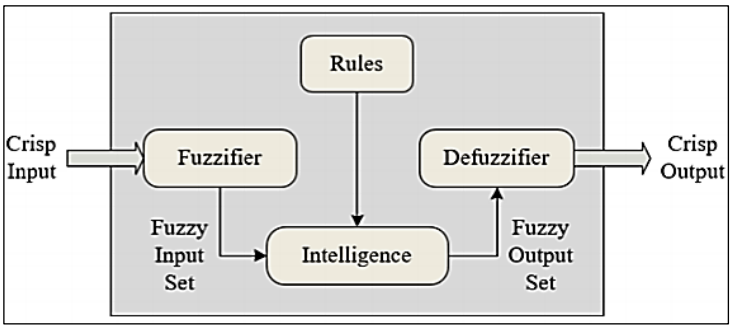

Figure 4.1 Steps of fuzzy logic process (Ross, 2017)

There are six unique rules about risks in the software development process based on the fuzzy logic method – fuzzy structure – in the following.

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 analysed, 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 the 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 the software development process will decrease.

The relation of "Exception Handling" to "Error Handling": Applying the property of "Exception Handling" in software development 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 the 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 the 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 the 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 the software development process will decrease.

4.2.2 The Demo Implementation

The demo has been developed and designed with the assistance of blend of "MATLAB" (for usefulness and activity) and "Python" programming language (for UI). Additionally, in this study, there are 18 etymological principles (the unique sensible guidelines) about danger evaluation and the executives with 15 diverse danger boundaries (decided a lot by writing survey as indicated by master suppositions listed and explained in the Table 4.2) in view of fluffy rationale for programming improvement ventures (Birant et al., 2019). Moreover, in the Table 4.4 in the following, the original linguistic software risks rule set (18 rules in total), which has been built and formed based on the parameters listed in the Table 4.2, has been shown, and also the abbreviations (15 software risk factors in total) used in the rules have been given and defined in the Table 4.3.

Table 4.3 The abbreviations of the rule parameters

No	Abbreviation	Definition
1	IDP	Inappropriate development process
2	IC	Ineffective communication
3	UMSR	Unclear or misunderstanding system requirements
4	PMNCD	Project milestones not clearly defined
5	LEPMM	Lack of an effective project management methodology
6	IPP	Inadequate project planning
7	ITM	Inexperienced team members
8	PIUNT	Project involved the use of new technology
9	ITDTM	Inadequately trained development team members
10	NUSMFS	New and/or unfamiliar subject matter for the system
11	LAUI	Lack of adequate user involvement
12	HLTC	High level of technical complexity
13	COMDP	Change in organizational management during the project
14	CPNEP	Corporate politics with negative effect on project
15	FIAS	Failure to identify all stakeholders

Table 4.4 The original linguistic rule set

No	The Original Rule
1	If “IDP” is medium and “IC” is medium and “UMSR” is high, then risk is high.
2	If “IDP” is low and “IC” is medium and “UMSR” is medium, then risk is medium.
3	If “IDP” is low and “IC” is low and “UMSR” is low, then risk is low.
4	If “PMNCD” is high and “LEPMM” is medium, then risk is high.
5	If “PMNCD” is medium and “LEPMM” is low, then risk is medium.
6	If “PMNCD” is low and “LEPMM” is low, then risk is low.
7	If “IPP” is medium and “ITM” is high, then risk is high.
8	If “IPP” is low and “ITM” is medium, then risk is medium.
9	If “IPP” is low and “ITM” is low, then risk is low.
10	If “PIUNT” is high and “ITDTM” is medium and “NUSMFS” is medium, then risk is high.
11	If “PIUNT” is medium and “ITDTM” is low and “NUSMFS” is medium, then risk is medium.
12	If “PIUNT” is low and “ITDTM” is low and “NUSMFS” is low, then risk is low.
13	If “LAUI” is medium and “HLTC” is high, then risk is high.
14	If “LAUI” is low and “HLTC” is medium, then risk is medium.
15	If “LAUI” is low and “HLTC” is low, then risk is low.
16	If “COMDP” is medium and “CPNEP” is high and “FIAS” is medium, then risk is high.
17	If “COMDP” is low and “CPNEP” is medium and “FIAS” is medium, then risk is medium.
18	If “COMDP” is low and “CPNEP” is low and “FIAS” is low, then risk is low.

In this application, the input and output ranges according to the fuzzy logic method have been determined about software risk assessment and management methodology as given in Table 4.5 in the following.

Table 4.5 Input and output ranges in the demo

Input Ranges (out of 10)			Output Ranges (out of 1)		
low	medium	high	low	medium	high
[0 2 4]	[3 5.5 8]	[6 8 10]	[0 0.2 0.4]	[0.3 0.5 0.7]	[0.6 0.8 1]

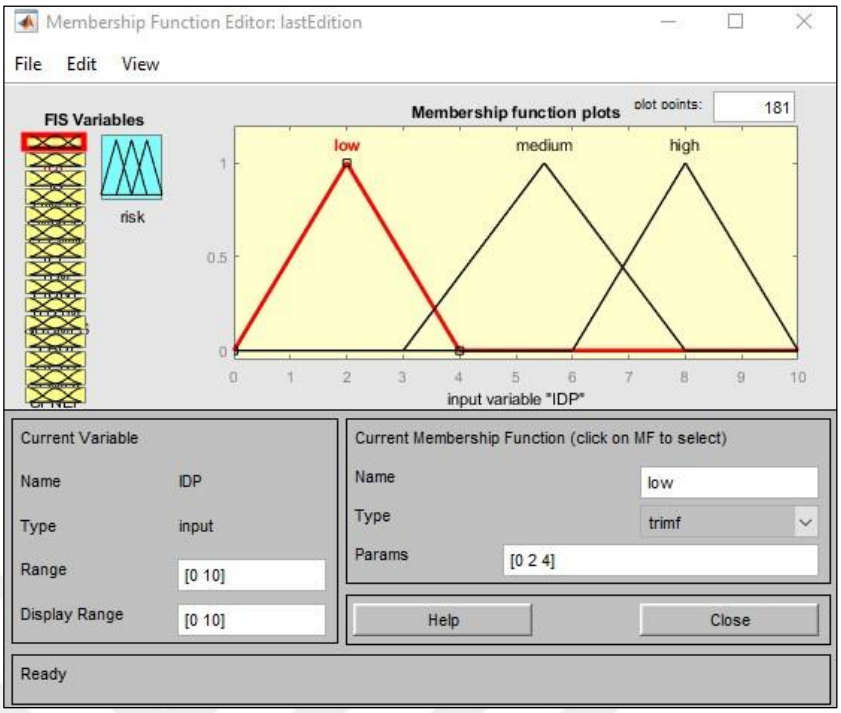

Figure 4.2 Input levels in fuzzy logic on MATLAB

According to the Figure 4.2, there has been a membership function belonging to an input in the fuzzy logic method implementation in the demo on MATLAB. Also, this functions has been formed by trimf type – behaved as a triangle function – in the application. In addition, three levels have been defined in the demo – low, medium and high. Moreover, for the low input, the minimum value has been 0, and the maximum value has been 4. Furthermore, the top belonging value for the input has been 2. Additionally, the values between 3 and 8 have belonged to the medium input. Also, the value of 5.5 has given the maximum belonging ratio for the medium level. Moreover, the minimum value has been 6, and the maximum value has been 10 for the high input. In addition, for the top belonging of the high level input, the value has had to be 8 in the implementation in the fuzzy logic demo.

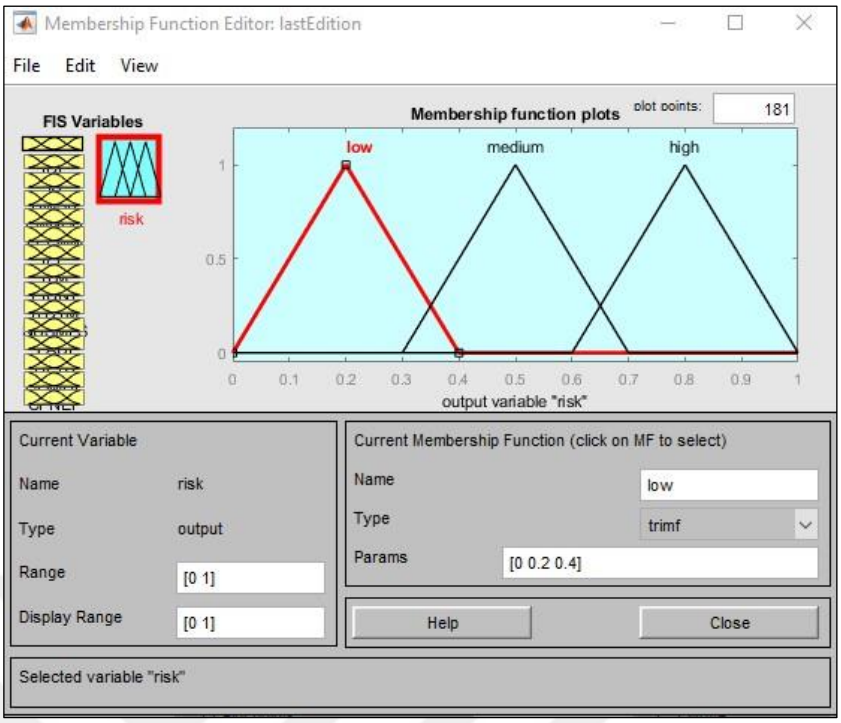

Figure 4.3 Output levels in fuzzy logic on MATLAB

Based on the Figure 4.3, a membership function has been shown and structured for the output in the fuzzy logic demo on MATLAB. Also, this function has been built by trimf type in the implementation as in the input membership function shown in the Figure 4.2. In addition, three levels (like the input levels) have been defined in the application for the output – low, medium and high. Moreover, for the low output, the minimum value has been 0, and the maximum value has been 0.4. Furthermore, the top belonging value for the output has been 0.2. Additionally, the values between 0.3 and 0.7 have belonged to the medium output. Also, the value of 0.5 has given the maximum belonging ratio for the medium level of the output. Moreover, 0.6 has been the minimum value, and 1 has been the maximum value for the high output. In addition, the value has had to be 0.8 in the implementation in the fuzzy logic demo on MATLAB for the top belonging of the high level output.

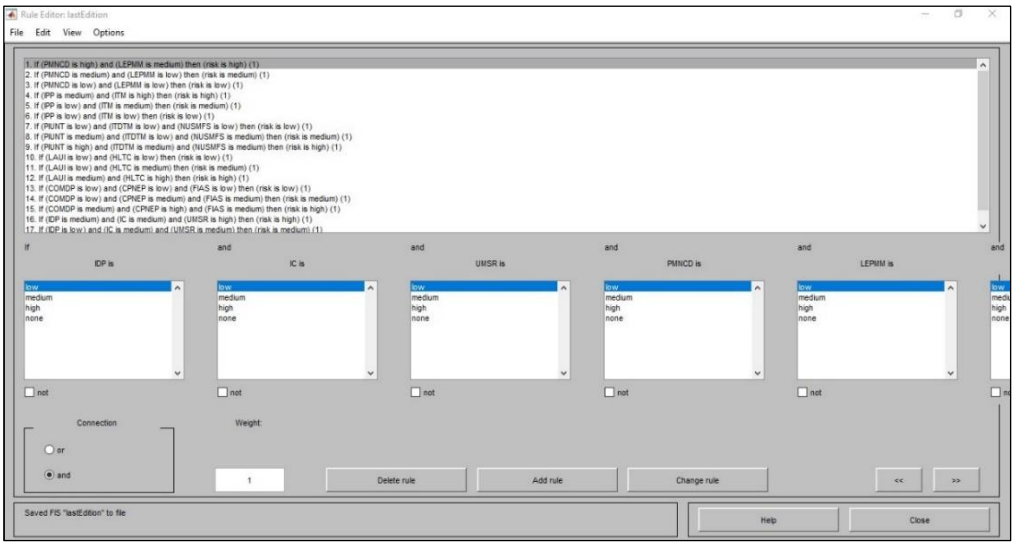

Figure 4.4 The linguistic rules on MATLAB

Figure 4.4 has showed the linguistic rules – declared and listed in the Table 4.4 – and the rule parameters – given in the Table 4.3 – belonging to the fuzzy logic method on MATLAB. These rules have been added into the implementation by using "Rule Editor", also the rule factors have been built and formed by adding variable in the fuzzy logic method on MATLAB. In addition, based on the values of the input rule parameters, fuzzy logic structure have given the result as an output value 0 to 1 – so has been interpreted that software risk is low, medium or high.

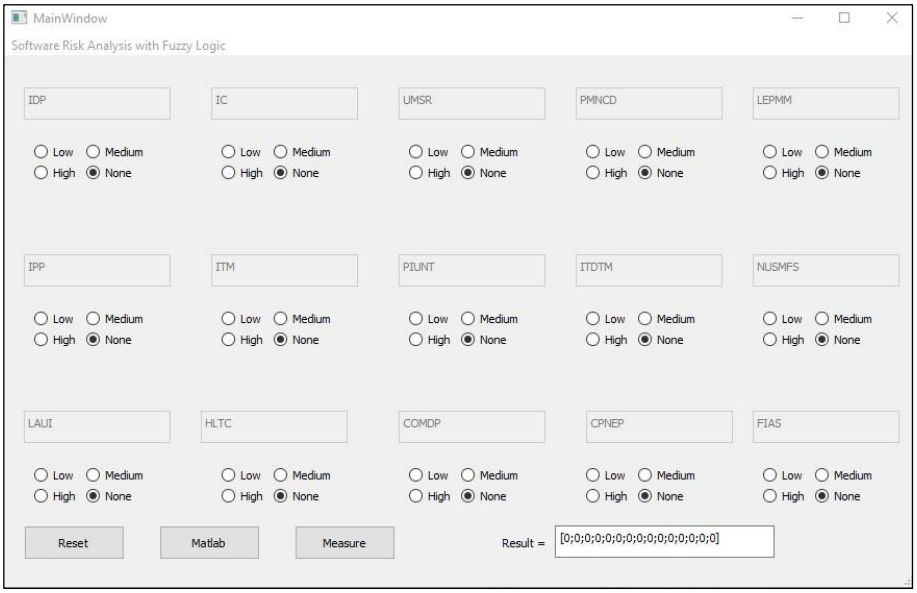

Figure 4.5 User interface for getting input on Python

An user interface has been designed and developed as given in the Figure 4.5 in the PyQT platform with Python programming language. Also, this has been implemented in order to get inputs from the users by clicking “low”, “medium” or “high” for ease of use. In addition, these inputs have been converted into numerical values as given in the Table 4.5 – low[0 2 4], medium[3 5.5 8] and high[6 8 10]. Accordig to these values, “low” has been converted into the value 2, “medium” has been changed into the value “5” (5.5 has been converged to 5), and “high” has been formed into the value 8. Finally, these numerical values of the rule parameters have been transformed into the fuzzy logic implemetation, and this method have given the results in the “Rule Viewer” on MATLAB as seen in the Figure 4.6 in the following.

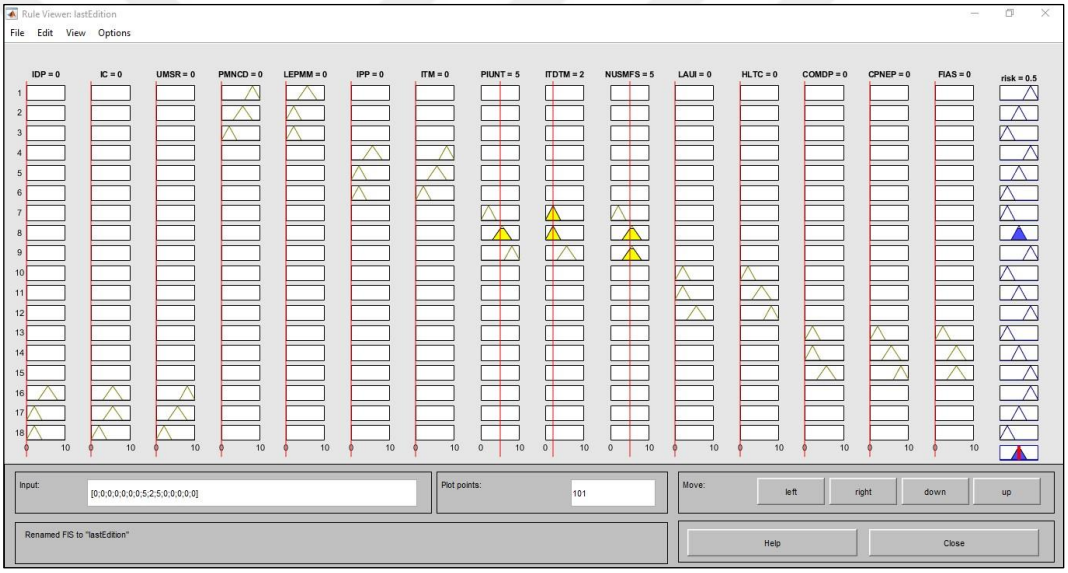

Figure 4.6 An example of a rule output on MATLAB

Based on the structure in the Figure 4.6, the “Rule Viewer” in the fuzzy logic method on MATLAB have given the result of an original risk parameter rule (listed in the Table 4.4): If *“project involved the use of new technology (PIUNT)”* is medium (value: 5) and *“inadequately trained development team members (ITDTM)”* is low (value: 2) and *“new and/or unfamiliar subject matter for the system (NUSMFS)” is* medium (value: 5), then *risk* is medium (value: 0.5).

CHAPTER FIVE
SOFTWARE RISKS RULE SET AND ITS DERIVATION

Risks are about vulnerability. On the off chance that individuals put a structure around that vulnerability, at that point they adequately de-hazard the undertaking, and furthermore that implies they can move significantly more unhesitatingly to accomplish the task objectives. By distinguishing and dealing with a complete rundown of undertaking chances, terrible astonishments and hindrances can be diminished and brilliant open doors found. The danger the board cycle additionally assists with settling issues when they happen in light of the fact that those issues have been conceived and plans to treat them have just been created and concurred. Individuals evade rash responses and going into "putting out fires" mode to amend issues that might have been envisioned. This makes for more joyful, less focused on venture groups and partners (Adler & Dumas, 1984).

5.1 Preparation of The Software Risk Parameters Rules

By and large, 19 gatherings were driven with experts from 14 unmistakable associations in India, USA and Spain. Also, the software experts came from various different spaces, for instance, flight, educational programming and custom programming headway for the financial business. By the exception of two interviewees who reported according to an investigator perspective, all of them came from the leaders positions, with 9 being adventure overseers and others standing firm on situations, for instance, quality chief, thing manager, or CIO. All that interviewees could report from a couple (up to 20) extensive stretches of inclusion with flowed and overall programming improvement adventures. An evaluation using ace meetings showed that the perils perceived by the model composed the real experiences in about 82% of the cases. In this product hazard boundaries rule set, there have been 23 impacting parameters and 36 distinct principles altogether (Lamersdorf et al., 2010). These rules have been recorded and displayed in Table 5.1 in the in the following.

Table 5.1 Software risk parameters rule set

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

Table 5.1 continues

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

5.2 Derivation of the Rule Set with ANFIS Method

Lamersdorf et al. fabricated and made a standard set – comprised of 23 programming hazard boundaries and 36 principles – in 2010 as clarified in the above segment 5.1 and recorded in the Table 5.1. Moreover, this product hazard boundaries rule set has been changed over mathematical qualities for determination and multiplication dependent on an AI calculation coordinated with man-made consciousness – ANFIS (Adaptive Neuro-Fuzzy Inference System). Likewise, for the execution of ANFIS, this numeric set has been stacked into MATLAB application for design. Besides, as per ANFIS strategy, 32 new guidelines have been derivated from the dangers rule set available.

5.2.1 ANFIS – Adaptive Neuro-Fuzzy Inference System

Adaptive neuro-fuzzy inference system is a system that combines the astounding capacities of all phony understanding theories, for instance, Fuzzy Logic (FL), Artificial Neural Networks (ANN) and expert structures. The fundamental quality of the ANN approach is the ability to learn. One of the huge burdens of this component is that the learning results can be addressed in incredibly colossal limit sets.

Henceforth, it is hard to impart the results by words. The reason of the FL is that it is amazingly close to the manner in which people thinking as in standard vernaculars. In any case, it can't acquire capability with the actual principles, it is dependent on ace structure or the viewpoints on people. Here, the meaning of adaptable neural fleecy reasoning is emerged (Jang, 1993).

The versatile neural fluffy rationale approach is uncovered incorporating the benefits of ANN's learning capacity and the simplicity with which the fluffy rationale can be settle on choices in human capacity and give master information. While FL frameworks can give the learning and computation limit of ANN, ANN likewise gain aptitudes, for example, fluffy control, dynamic and furnishing master information with this technique. Versatile Neural fluffy control framework is utilized ANN and FL procedures to decide the qualities of the factors that will shape its own structure (Walia et al., 2015).

ANFIS works just Sugeno type models. It is as modification of the Mamdani type exhibiting and the exercises to be applied to the data are the same. The principle contrasts are in the yield data. In Sugeno type illustrating, yield factors have enlistment limits as a limit of data sources and yield enlistment limits should be immediate or consistent. ANFIS is a limit archive under the Fuzzy Logic Toolbox in MATLAB. This program is applied feathery enlistment techniques to the data given to it and is made reasonable data models. Additionally likewise with other soft derivation systems, enlistment works in ANFIS depend upon limits. If the limits are changed, enlistment limits will change and ANFIS will consign self-enlistment limits (Chelgani et al., 2011).

In the ANFIS network structure – has been exhibited in Figure 5.1 in the following –, which is outlined by cushioned principles, there are cushy enlistment limits, feathery duplication, normalization, arrangement and direct yield work layers in Sugeno type and ANFIS structure has five layer designing. In ANFIS, which includes five layers, the nerves in each layer contain comparable assignments. Center points with different shapes in layers contain limits with different limits. The center points seemed perfectly

healthy are called adaptable centers and the limits of the center points are set during the arrangement of the association. Center points showed up as circles are fixed centers (Shoorehdeli et al., 2006).

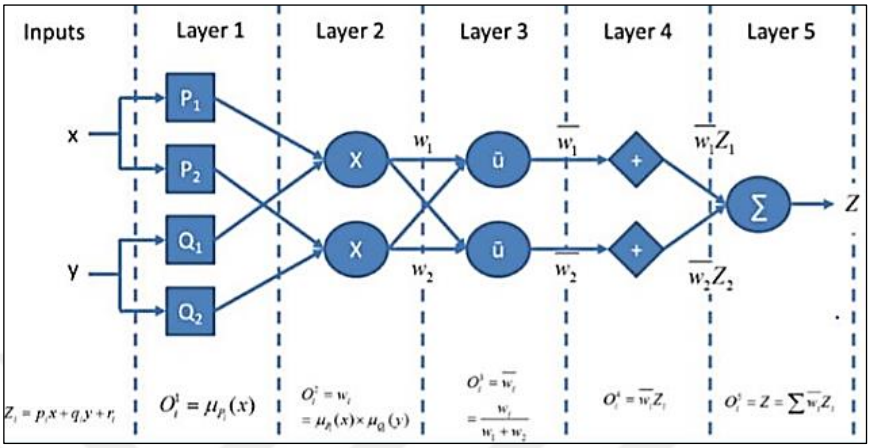

Figure 5.1 ANFIS architecture and layers (Walia et al., 2015)

5.2.2 ANFIS Application for Building New Rule Set

In order to implement ANFIS – a machine learning algorithm – on MATLAB, the rule set on hand has to be converted into numeric values. For this reason, 36 software risk rules (listed and explained in the Table 5.1) have been transformed into the values 0, 1, 2 and 3. Also, the value 0 has meant that there isn't 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 showed that that the case of the parameter is medium. Furthermore, the value 3 has been determined that the state of the parameter is high. In the result of these values, the rules on hand have been transformed into Table 5.2.

Table 5.2 Numeric values for the rule set

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

Table 5.2 continues

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

These values (Table 5.2) of the rule set (Table 5.1) on hand has been loaded into MATLAB (Figure 5.3 and Figure 5.4) as a txt file as given in the Figure 5.2 in the following in order to be trained and to learn new rules, and then to put out and generate new original rules (Figure 5.5) based on the derivation principle of machine learning

algorithm integrated with artificial intelligence approach – ANFIS (Adaptive Neuro-Fuzzy Inference System).

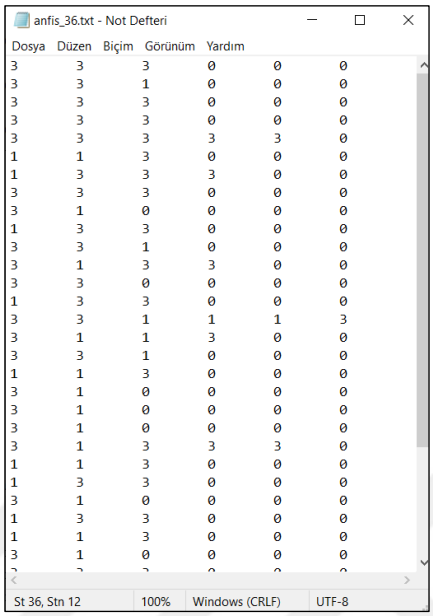

Figure 5.2 A txt file for numeric values of the rules

Figure 5.2 has belonged to the txt file of the rule set's numerical values as given in the Table 5.2. This file has been formed and prepared in order to load the rule set to implement ANFIS machine learning method on MATLAB for the derivation of the rules.

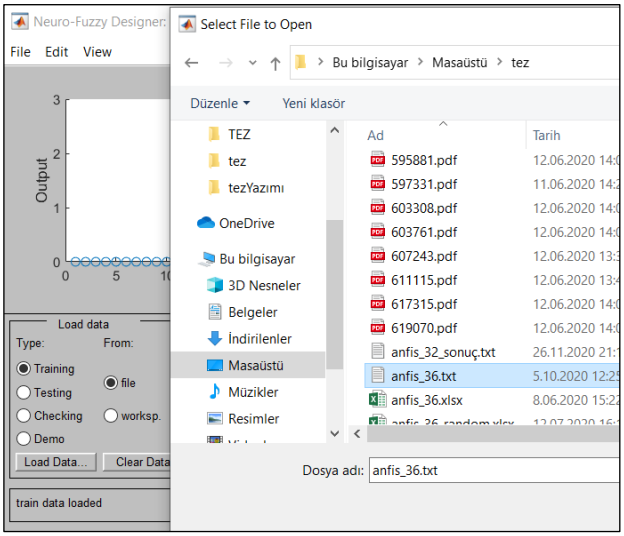

Figure 5.3 Loading the txt file on MATLAB

Figure 5.3 has demonstrated that the txt file shown in the Figure 5.2 has been loaded into the ANFIS implementation on MATLAB by clicking “Load Data” on the application. So, the rule set, which has been listed in the Table 5.1, has been got ready for generation process.

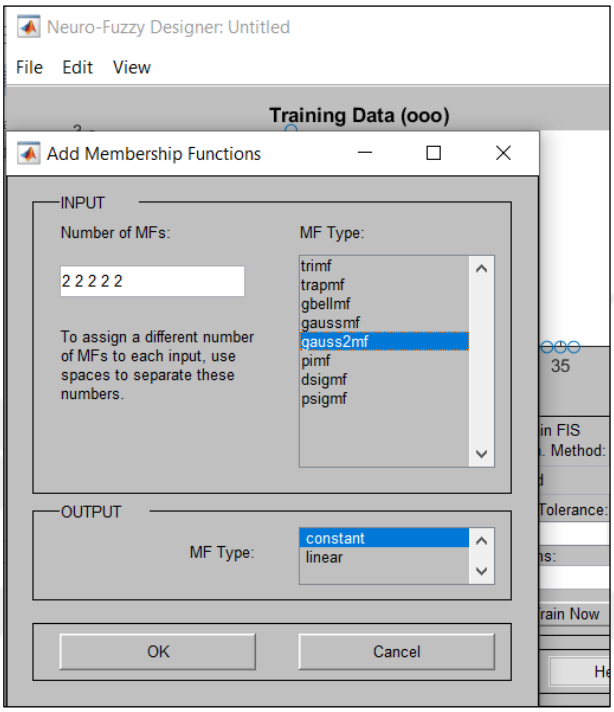

Figure 5.4 Set of the gauss membership function with the value 2

According to the Figure 5.4, the membership function has been chosen for the input parameters of the rule set (Table 5.1) as gauss2mf. This membership function is a smooth curve calculated from two Gaussian membership functions. Also, the number of membership for the input has been given as the value 2 – one unit less than the value of the state of the parameter (high) explained in the section 5.2.2. Based on this value, the rule set has been applied into the ANFIS (Adaptive Neuro-Fuzzy Inference System) method in order to generate new original software risk rules on MATLAB. Thus, 32 new rules (membership value is 2 and input numbers are 5, so 2^5=32 units) have been derivated from these input rules as given in the Figure 5.5 in the following.

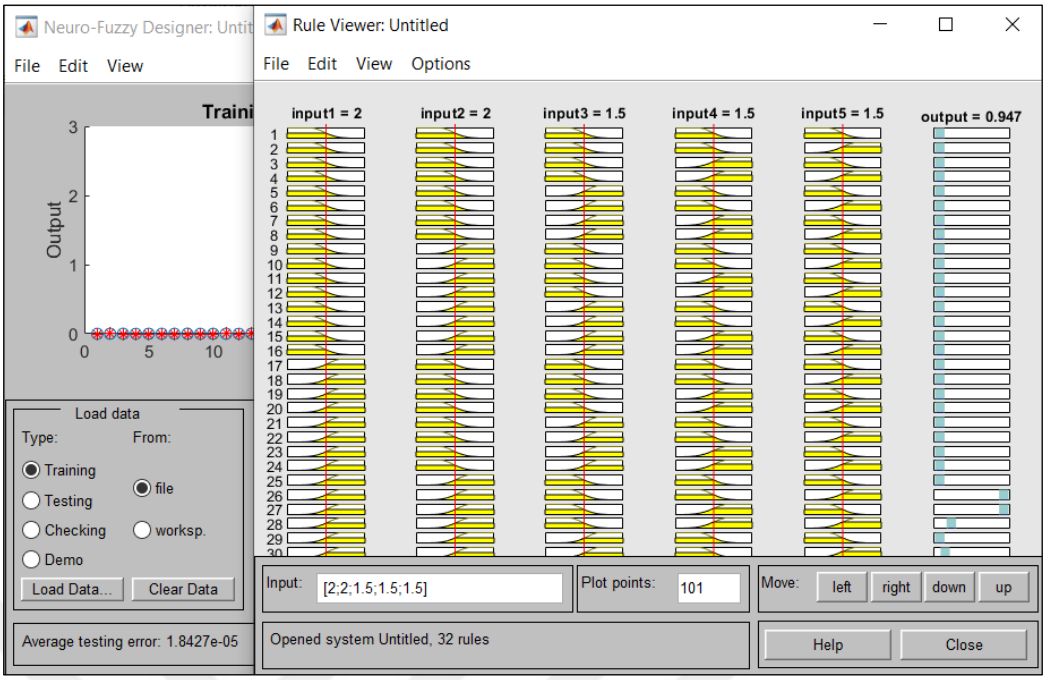

Figure 5.5 Derivation of the rules with ANFIS

The average testing error has showed up very low (0.000018) after the implementation of ANFIS in MATLAB as given in the Figure 5.5. That means the rule set, which has been loaded into MATLAB for derivation of the rule by ANFIS configuration, has been valid and consistent – in other words the fuzzy inference model belonging to the specific rule set has been accurate and valid (Doufesh et al., 2016; Jassar et al., 2009; Kaynak et al., 2015; Mohammed et al., 2017; Neshat et al., 2011; Salehi et al., 2014). However, if the set weren't correct or accurate, the average testing error will have been affected-size (great). In addition, in order to understand this situation better, another txt file has been formed and written by random values 0, 1, 2, 3 with the same size (same number of columns and rows) showed in the Table 5.2 and given in the Figure 5.2. Afterward, this file has been loaded into MATLAB for ANFIS implementation. Furthermore, this txt file has been placed in the Figure 5.6 and the result of the implementation of this txt file has been shown in the Figure 5.7 in the following.

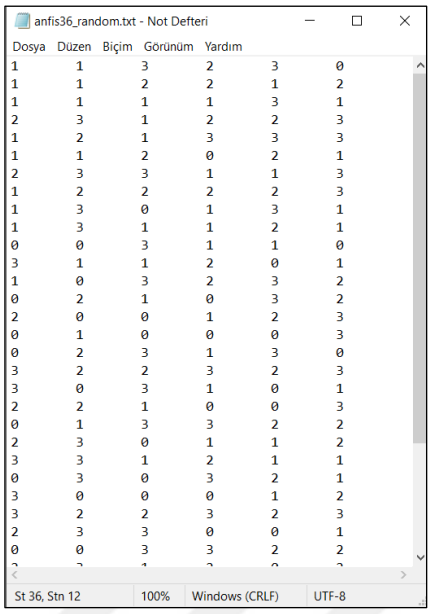

Figure 5.6 A txt file with random values

Figure 5.6 has belonged to the txt file with random values in order to compare the results of the ANFIS implementation on MATLAB with the txt file (Figure 5.2), which has included numerical values (Table 5.2) of the rule set (Table 5.1), in respect to average testing errors. This new txt file has been formed and prepared in order to load the random and invalid rule set to implement ANFIS machine learning method with integrated artificial intelligence on MATLAB for the rule generation.

Figure 5.7 The result of the random values with ANFIS

According to the result of ANFIS implementation of the txt file with random values as given above in the Figure 5.6, this txt file hasn't had a valid rule set because of the great average testing error – 0.274022 (Figure 5.7). In the light of this case study, it can be easily seen that the first risk parameters rule set as listed in the Table 5.1 has been accurate and valid based on the scientific numeric value – average testing error: 0.000018. But, the second rule set's (Figure 5.6) average testing error 10000 (10^4) times more than the first rule set's, so that has not been recognizable and not valid according to the scientific findings and articles (Doufesh et al., 2016; Jassar et al., 2009; Kaynak et al., 2015; Mohammed et al., 2017; Neshat et al., 2011; Salehi et al., 2014).

Based on the confidential and reliable rule data set as listed in the Table 5.1 and explained in the section 5.1, 32 (membership value is 2 and input numbers are 5, so 2^5=32 units) new original rules have been formed and generated with ANFIS (Adaptive Neuro-Fuzzy Inference System) method on MATLAB. In addition, all these rules have six parameters – 5 parameters for the input and 1 parameter for the output. Moreover, this new rule set has been valid according to the tiny (very very low) average testing error (Doufesh et al., 2016; Jassar et al., 2009; Kaynak et al., 2015; Mohammed et al., 2017; Neshat et al., 2011; Salehi et al., 2014). Furthermore, the derivated software risk parameters rule set and its numeric values have been listed and explained in the Table 5.3 in the following. These numeric values have been determined, assigned and popped up automatically in the ANFIS implementation on MATLAB. Accordingly in the section 5.2.2, 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 showed that 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.

Table 5.3 The derivated rules and their numerical values

No	Derivated Rules & Numeric Values
Derivated 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

Table 5.3 continues

Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated Rule 6	Cultural difference is medium & Quality problems is medium & Communication 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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

Table 5.3 continues

Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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

Table 5.3 continues

Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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

Table 5.3 continues

Derivated 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
Derivated 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
Derivated 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
Derivated 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
Derivated 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

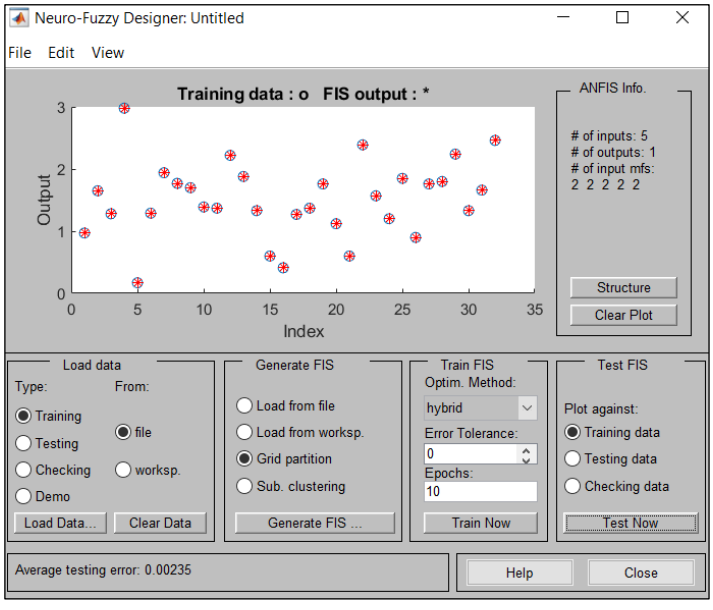

Figure 5.8 The result of the derivated rules implementation with ANFIS

Figure 5.8 has proved the validity (the model validity, as well) of the derivated rules as given in the Table 5.3 with ANFIS method on MATLAB by showing the average testing error: 0.00235 – that has been too low as in the Figure 5.5: 0.000018. So, it could be said that the less average testing error, the more valid the fuzzy inference model (the software risks parameter set) (Doufesh et al., 2016; Jassar et al., 2009; Kaynak et al., 2015; Mohammed et al., 2017; Neshat et al., 2011; Salehi et al., 2014).

All in all, 36 programming hazard rules (Lamersdorf et al., 2010) which have been demonstrated as a legitimate and unmistakable model (clarified exhaustively in the above passages in this segment 5.2.2), have been designed and carried out by ANFIS (Adaptive Neuro-Fuzzy Inference System) – the AI calculation incorporated with man-made brainpower structure. A while later, 32 (participation esteem is 2 and information numbers are 5, so 2^5=32 units) new unique principles have been made and created with next to no average testing blunder to fabricate the first etymological danger boundaries rule set. Also, each standard has six boundaries – five variables for the information, one factor for the yield – altogether. Besides, in the light of this got information, it has pointed and expressed that Adaptive Neuro-Fuzzy Inference System (ANFIS) strategy could work successfully and adequately in the space of programming hazard evaluation and the executives by giving the dependable outcomes on MATLAB. This strategy has been run with the commitment of “Neuro-Fuzzy Designer” (Figure 5.9 in the following) on MATLAB.

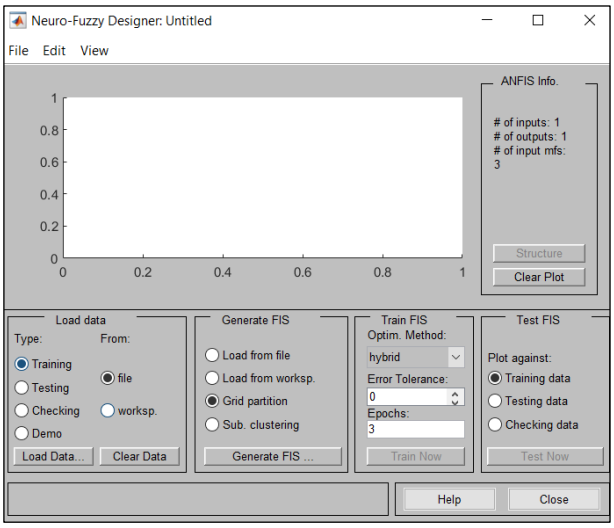

Figure 5.9 Neuro-fuzzy designer on MATLAB

CHAPTER SIX
SURVEY STUDY

The survey has been prepared in order to show accuracy ratings of automatically derivated (designed and developed) new (original) rules (explained in the chapter five) – 32 original software risk parameters rules – among software companies in the real world according to the subjective evaluation and analysis of software experts. In the light of this survey, the develop rules' subjective validity and subjective accuracy levels – in percentages and in numbers – have been found out and shown.

6.1 Preparation of The Survey

There have been 32 new generated rules – listed and explained in detail in the chapter five – in the original set of the risk assessment and management methodology based on the derivation of ANFIS (Adaptive Neuro-Fuzzy Inference System) implementation on MATLAB. In addition, each rule has been converted into a table structure with different pages (next by next style) to be understood better and more easily by the survey respondents. However, at first, these rules have been written in plain text form with same page (from top to the bottom – downwards style) that has been read more difficult and have taken more time from the participants in the survey. Moreover, there have been six parameters in total in every rule – for the input, there have been five parameters; for the output, there has been just one parameter. Moreover, the rules have been evaluated based on the multiple choice with three alternatives – "Agree", "Disagree", and "No idea". The table forms of the derivated rules (defined in detail in the chapter five) have been given and shown with the selection alternatives, and also these rules' plain text forms have been figured to put a finer point on that with their explanations in the following – for the plain text template, Figure 6.1 has been captured; for the table structure, Figure 6.2 has been captured.

Derivated Rule 1 (Figure 6.1): 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

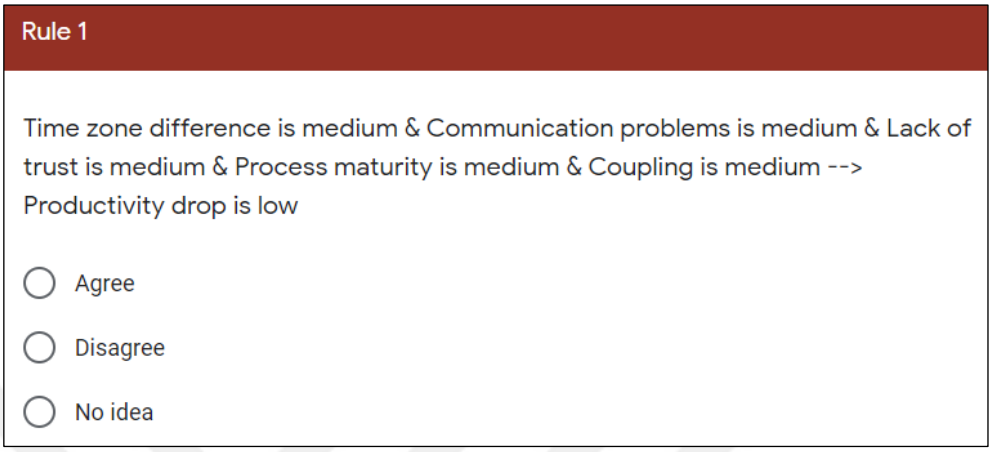

Figure 6.1 Plain text structure of the derivated rule 1

Derivated Rule 32 (Figure 6.2): Productivity drop is high & Complexity is medium & Process maturity is low & Transparency is low & Personal relationships is medium → Coordination problem is high

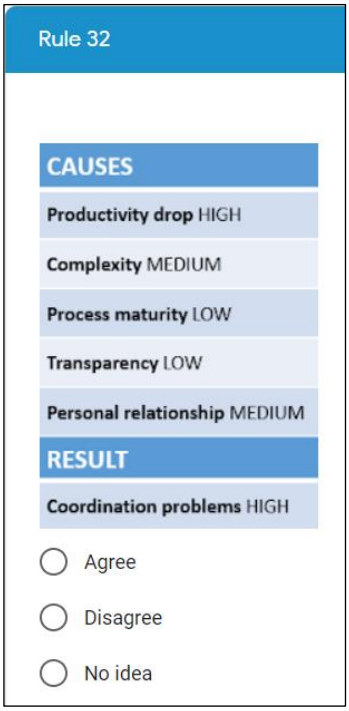

Figure 6.2 Table form of the derivated rule 32

6.2 Carrying out The Survey

The table matrix formed rules given in the Figure 6.2 have had main two parts: causes and result (causes (input parameters) → result (output parameters)). They have been put and showed in a questionnaire. Furthermore, this questionnaire has been prepared and written in Google Forms on internet, and so it could be done and conducted more easily as well as could be sent, given and taken more and more people faster in a short time. Moreover, these 32 procedures (deviated rules – original software risk parameters rule set – as in the chapter 5) have been evaluated by 108 software experts (all of them are *senior* software developers) from 76 different companies (freelance unit, start-up and small companies, big and corporate companies, public enterprises, private institutions, national companies, international companies, universities, etc.) for showing their subjective analysis and subjective validity/ accuracy levels of these rules based on their own opinions.

The companies whose software experts have conducted the survey have been listed in the Table 6.1 in the following.

Table 6.1 The companies list in the survey

No	The Company
1	1001 App
2	Accenture
3	Akbank
4	ARETE consulting
5	ASELSAN
6	AvivaSA
7	Aydın Adnan Menderes University
8	Bakırçay University
9	Bartın University
10	Bayzat
11	Bitlis Eren University
12	Borda Technology
13	BorgWarner
14	Camdalio Software
15	Cesa Software
16	CFM Cooling and Automation Inc.
17	Cross Turkey
18	DefineX
19	Dept Agency
20	DEU – The Office Of Career Planning & Alumni Relations
21	Deytek

Table 6.1 continues

22	Ecospend Technologies Limited
23	Ege University Hospital
24	Egebis Information Systems
25	Emakina.TR
26	enibra Software and Consultancy
27	ETCBASE
28	Freelance Unit
29	Garanti BBVA Technology
30	GE Aviation
31	Getir Retail Logistics Corporation
32	Guven Seker - kidandmore.com
33	Halkbank
34	Happy Game Company
35	HAVELSAN
36	Honeywell
37	Innobo Labs
38	Intertech Information Technology
39	Invent Analytics
40	i2i Systems
41	Istanbul Technical University
42	Izmir Metropolitan Municipality
43	Izmir Katip Celebi University - Atatürk Training and Research Hospital
44	KAL - ATM Software
45	2th District Office of Highways
46	Kent College: Izmir Private School
47	KoçSystems
48	KuveytTürk Participation Bank
49	Logo Software
50	Maroteknology
51	Metrohm Autolab BV
52	Migros Trade Corporation
53	MobileUpp
54	Neocrea Software Solutions
55	Nesine.com
56	OREDATA
57	Proto Software
58	PruvaSoft Information Technologies
59	Recep Tayyip Erdogan University
60	SEBIT Education and Information Technologies Inc.
61	Serenity Software
62	Sgk Directorate of Service Delivery
63	Sitecore
64	SoftTech
65	Stackpole International
66	Supply Chain Wizard, LLC
67	trendyol.com
68	TURK Financial Technology
69	Turkcell Technology
70	Turk Telekom
71	Turkiye İş Bank
72	VeriPark
73	Vestel
74	Vodafone
75	Yapı Kredi
76	Yeşilırmak Electricity Distribution Corporation

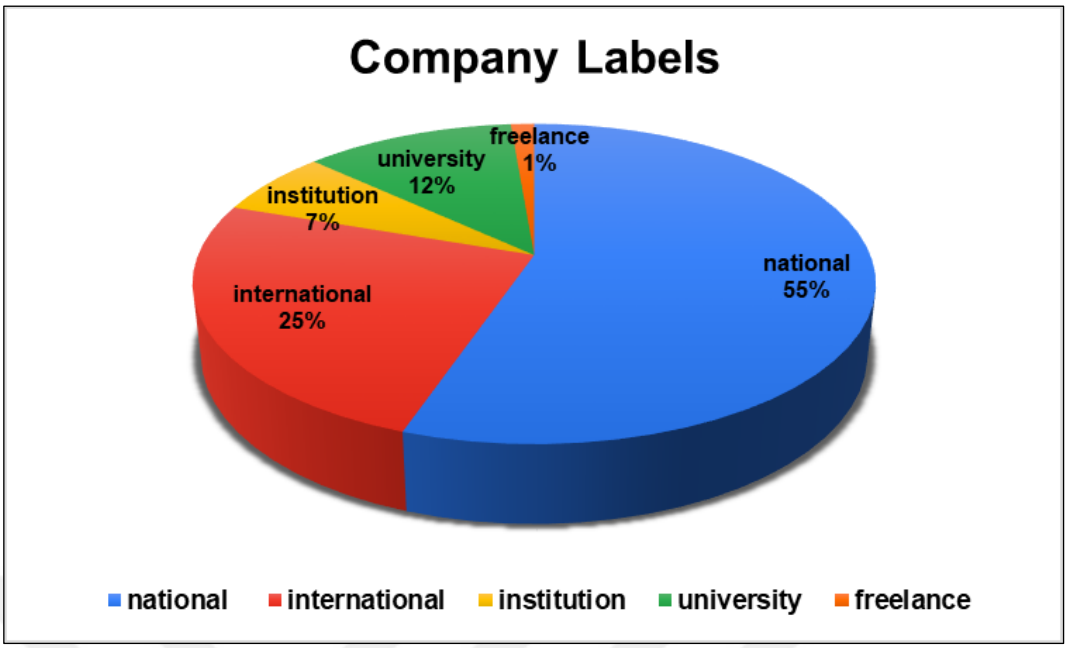

Figure 6.3 The company labels in the survey

When it has been looked at the companies (listed in the table 6.1 – # of these: 76), whose software developers and experts have taken the survey, the percentage distribution of these have been as shown in the Figure 6.3. According to the findings in this figure, about 55 percent of the companies have been a national firm – 42 companies. In addition, the company labels have been composed of international in the percentage of 25 – 19 firms. Also, about 7% of these companies have been related to an institution (a municipality, a hospital, a school, etc) – 5 institutions. Moreover, these company labels have consisted of universities in the ratio of about 12 percent – 9 universities: Aydın Adnan Menderes University, Bakırçay University, Bartın University, Bitlis Eren University, Dokuz Eylül University, Ege University, Istanbul Technical University, İzmir Katip Celebi University and Recep Tayyip Erdogan University. Furthermore, about 1 percent of the given labels is freelance – working freely.

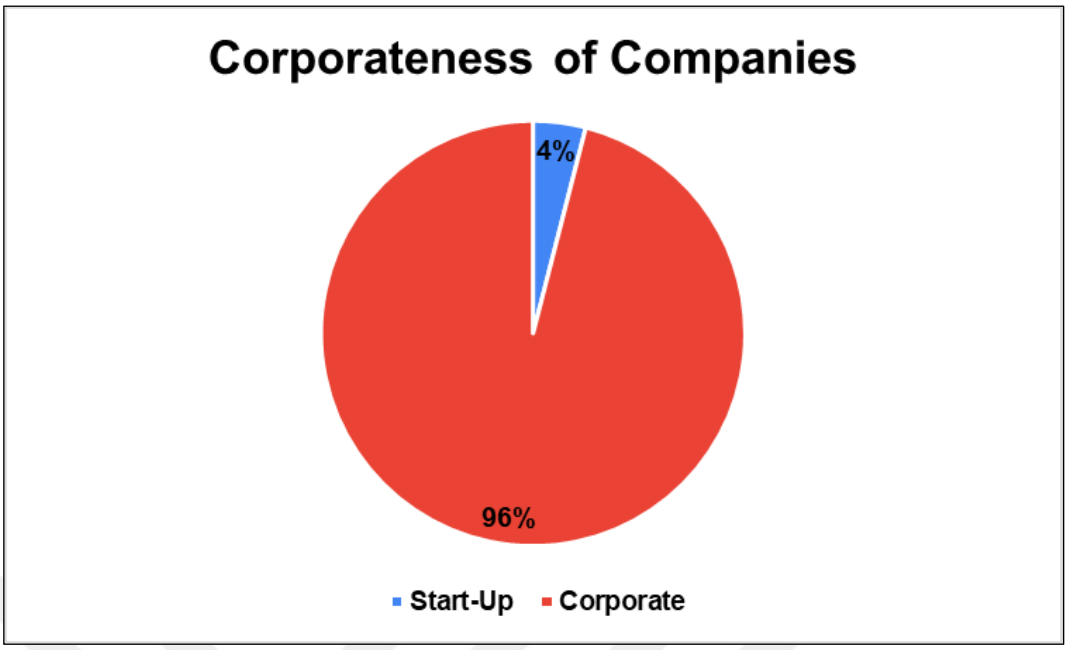

Figure 6.4 Corporateness of the companies in the survey

The companies which have placed in the survey have been formed by 76 different companies in total as listed in the table 6.1. Also, a company label has been "freelance" as given in the Figure 6.3. Because of that, these firms have been taken as the number 75 – one unit less than total number of companies in the survey. In addition, about 4 percent of these companies have been in the start-up (3 out of 75), and about 96 percent of the firms in the survey have been more corporate (72 out of 75) regarding the company establishment year, the number of employees, the duties done, etc. Thus, it could be said that great majority of the software developers, who have conducted the survey, have worked in a company that has had a corporate identity – in other words, these experts have had to work in a discipline, in a working principle, and in a specific and certain working culture. In the result of this situation, they might have looked at the survey questions and rules from the point view of this experience – this might have made the survey results more reliable and more trustworthy in an experimental way.

The frequency and percentages of the derivated rules' – listed and explained in detail in the chapter five – subjective validation according to the software experts' opinions and answers (Agree, Disagree, No idea) in the given survey via Google Forms on internet have been given in the Table 6.2 in the following.

Table 6.2 The answers' detail of the derivated rules in the survey

Derivated Rule	Agree (f)	Disagree (f)	No i. (f)	TOTAL (f)	Agree (%)	Disagree (%)	No i. (%)	TOTAL (%)
1	73	33	2	108	68	30	2	100
2	69	38	1	108	64	35	1	100
3	54	49	5	108	50	45	5	100
4	82	25	1	108	76	23	1	100
5	65	35	8	108	60	32	8	100
6	68	16	24	108	63	15	22	100
7	74	28	6	108	69	26	5	100
8	60	25	23	108	56	23	21	100
9	46	43	19	108	43	40	17	100
10	64	41	3	108	59	38	3	100
11	64	31	13	108	59	29	12	100
12	52	37	19	108	48	34	18	100
13	71	21	16	108	66	19	15	100
14	50	44	14	108	46	41	13	100
15	62	39	7	108	58	36	6	100
16	64	28	16	108	59	26	15	100
17	71	29	8	108	66	27	7	100
18	81	12	15	108	75	11	14	100
19	58	39	11	108	54	36	10	100
20	54	29	25	108	50	27	23	100
21	67	29	12	108	62	27	11	100
22	91	8	9	108	84	7	9	100
23	60	36	12	108	56	33	11	100
24	55	51	2	108	51	47	2	100
25	75	12	21	108	70	11	19	100
26	61	37	10	108	57	34	9	100
27	55	39	14	108	51	36	13	100
28	47	19	42	108	44	17	39	100
29	73	26	9	108	68	24	8	100
30	65	22	21	108	60	20	20	100
31	69	26	13	108	64	24	12	100
32	61	37	10	108	57	34	9	100

According to the Table 6.2, *the most agreed rule* with 91 “agree” out of 108 answers and 84 out of 100 percent in total (also, 8 “disagree” answers with 7% (percentage) and 9 “no idea” answers with about 9% (percentage)) has been “Derivated Rule 22” in the following:

Phase = coding is medium & Productivity downfall is high & Communication problems is high & Transparency is medium & Common experiences is low → Coordination problem is high

“The derivated rule 22” has expressed that coordination problems in a software development projects have depended on the five main parameters: coding process in the development, productivity of the software development mechanism, communication among software developers in the project, transparent project progress, and common experiences of developers to each other in the software development project.

Based on the results as shown in the Table 6.2, *the most unrecognized (rejected) rule* with 51 “disagree” out of 108 responses and 47 out of 100 percent in total (as well, 55 “agree” responses with 51% (percentage) and 2 “no idea” responses with about 2% (percentage)) has stated “Derivated Rule 24” in the following.

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

“The derivated rule 24” has been interpreted that software development project failures could have been influenced less from the five given input parameters: coupling/coupling tasks in the software development process, maturity level of the project progress, trust level among developers to each other in the project, quality issues in the development, and communication structure in the software project development progress. In addition 51% of the respondents, who have taken the survey, have admitted the state of software project failure has been medium. But, it could be commented that 47% of the software experts, who have been the participants of the

questionnaire have had an opinion that the case of the project failure in the software development process might have been low instead of medium by being affected less.

Based on the results as shown in the Table 6.2, *the most dismissed rule* with 49 "disagree" out of 108 responses and 45 out of 100 percent in total (as well, 54 "agree" responses with 50% (percentage) and 5 "no idea" responses with about 5% (percentage)) has stated "Derivated Rule 3" in the following.

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

"The derivated rule 3" has been commented that productivity of the software development project could have been affected much more from the given five parameters: time pressure during the software development, novelty of software product in the project, cultural differences among software developers working in the project, the number of sites (teams, groups, work, package, module, etc), and the stability of requirements of the software development project. In addition 54% of the software developers, who have taken the survey, have confirmed the productivity downfall situation has been medium. However, it could be said that 45% of the participants of the questionnaire have had an opinion that the case of productivity downfall in the software development process might have been high instead of medium.

According to the findings as given in the Table 6.2, *the most uncertain/unclear/indefinite rule* with 42 "no idea" out of 108 replies and 39 out of 100 percent in total (moreover, 47 "agree" replies with 44% (percentage) and 19 "disagree" replies with about 17% (percentage)) has pointed "Derivated Rule 28" in the following.

Complexity is high & Quality problems is medium & Common experiences is low & Time zone difference is low & Transparency is high → Number of sites is medium

"The derivated rule 28" has stated that number of sites (working groups, working packages, working modules, team, etc.) have been the result of the five influencing factors: complexity of the software development project, quality of the software product, common experiences among software developers in the project, time zone areas of the developers, and transparency of the software development mechanism. But, remarkable number (42 "no idea" answers with about 39% (percentage)) of the software experts, who have conducted the survey, have had no opinion (no idea) about the parameter "number of sites". It could be commented that they haven't known what the site is/means. Because of that, this parameter "number of sites" might have been defined in order to clarify their minds in the questionnaire to take more reliable feed-backs from the participants in the questionnaire.

The numbers as listed in the Table 6.2 have figured that *the closest answers* between "agree" and "disagree" has belonged to "Derivated Rule 9" as given in the following by in the same turn, 46 to 43 frequencies and 43 to 40 percentages (furthermore, 19 "no idea" answers with about 17% percentage). Also, this rule has had *the lowest agreement ratio* among the participants' responses in the conducted survey.

Travel cost overhead is medium & Task coupling is medium & Process knowledge is low & Time pressure is high & Requirements stability is high → Maturity is medium

"The derivated rule 9" has showed that there have been five main influencing factors to affect process maturity of the software development project: travel cost of software developers in the project, task coupling in the development, process knowledge of the developers, time concept of the software development progress, and the software development project requirements. In addition, almost same number of the software experts who have taken the survey, both have accepted and at the same time, have denied this rule. This situation has been commented that the software development project process maturity could have been affected much more in the result of "the derivated rule 9" definition. In other words, it could be declared that 40% of the participants in the survey have thought that the process maturity level of the

software development project might have been low instead of medium while 43% of the software experts in the questionnaire have figured it has been well-defined.

The following figures have been clearly showed and illustrated the ratios belonging to the number of the derivated software risk parameter rules – clearly defined in the chapter 5 – based on the software experts' choices in the conducted survey; for the "agree" choice, Figure 6.5; for the "disagree" choice, Figure 6.6; for the "no idea" choice, Figure 6.7 have been drawn.

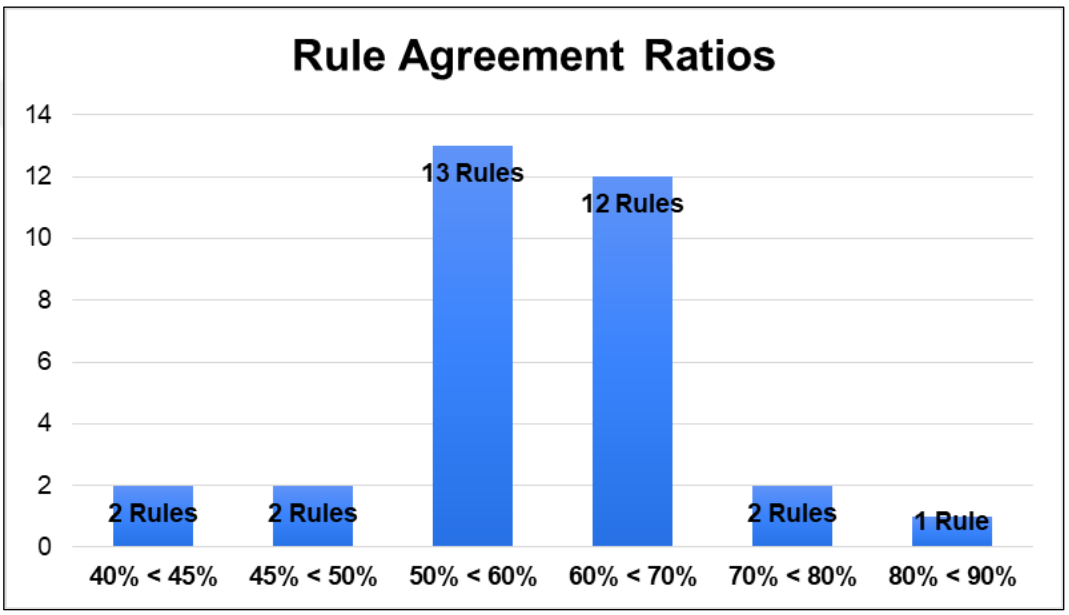

Figure 6.5 The number of rules based on the "agree" choice

According to the Figure 6.5, 28 derivated rules – almost all rules – have been accepted by more than half of the software experts, who have taken the survey, whereas just only 4 rules have been approved by less than half of them. In addition, 13 generated rules have been admitted in the percentages between 50% and 60%. Also, 60% to 70% (percentage) of the participants have confirmed 12 new original rules. Moreover, 2 derivated rules have been accepted by the ratio of 70% to 80% in total. Furthermore, the software developers, who have conducted the questionnaire, has accepted a software risks rule in more than 80% (percentage) – this rule has been confirmed by almost all respondents in the study.

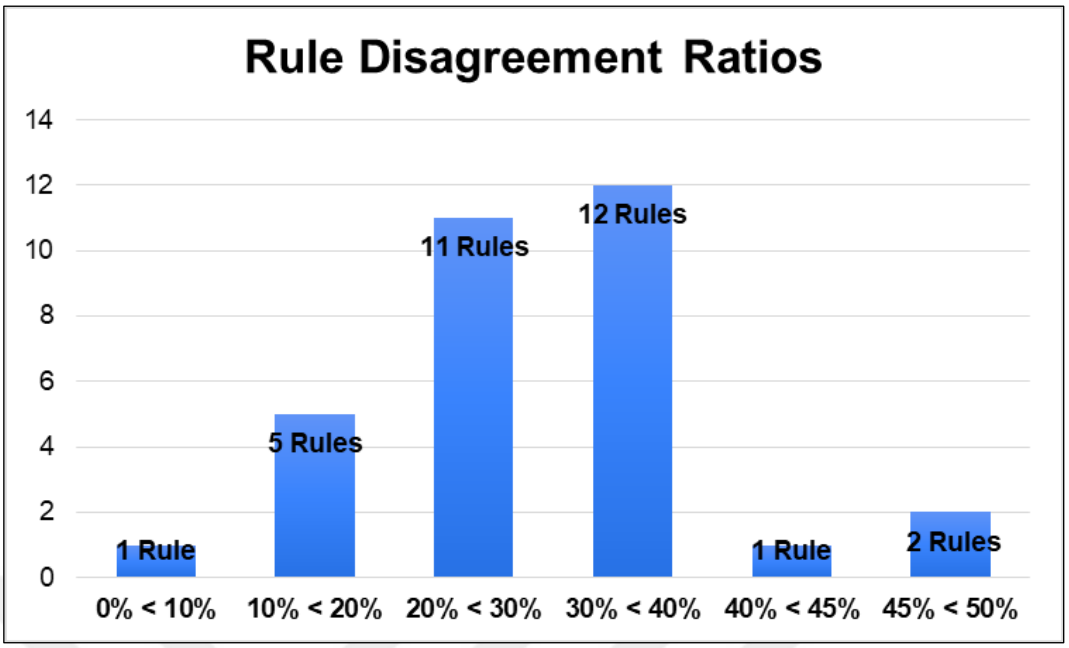

Figure 6.6 The number of rules based on the "disagree" choice

The findings in the Figure 6.6 have shown that all rules have had less than the percentage of 50 in the choice of "disagree" in the survey. Moreover, 29 rules have had less than the percentage of 40 in the selection of "disagree" in the questionnaire. Also, less than 30 percentage of the respondents, who have conducted the survey, have denied 17 new rules in the study. In addition, 5 new generated rules have been rejected in the percentages between 10% and 20%. Furthermore, a new derivated software risks rule has been declined by the ratio of less than 10% (percentage) – in other words, the rule has been accepted and approved by almost all of the software experts, who have taken the survey.

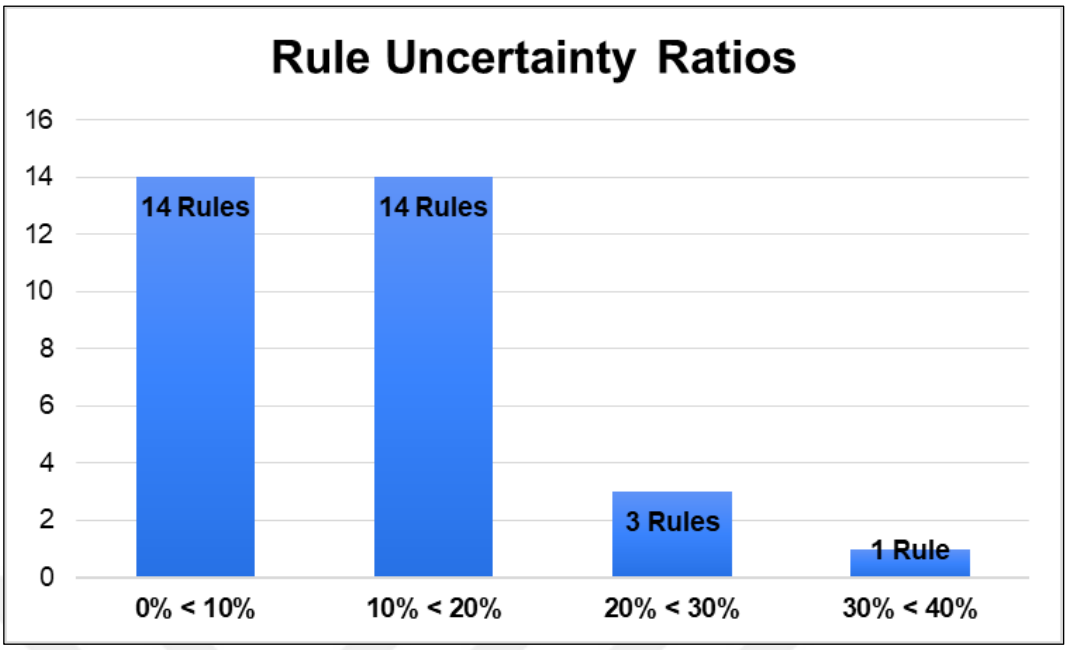

Figure 6.7 The number of rules based on the "no idea" choice

The Figure 6.7 has showed the number of the "no idea" answers based on the ratio of the software experts' choices, who have applied the survey in the study. Also, almost all rules (28 out of 32 rules) have had less than the percentage of 20 in the uncertainty concept. In addition, 14 generated rules have been selected as "no idea" in the percentages between 10% and 20%. Moreover, other 14 new derivated software risk parameters rules have been chosen as indefinite by less than 10 percent of the respondents, who have taken the questionnaire – means that more than 90 percent of the participants have defined these 14 rules as clear.

Furthermore, the derivated rule set, which has been listed in the Table 5.3 and explained in detail in the chapter 5, has been valid and accurate based on the ANFIS (Adaptive Neuro-Fuzzy Inference System) method on MATLAB by the low average testing error proof (Doufesh et al., 2016; Jassar et al., 2009; Kaynak et al., 2015; Mohammed et al., 2017; Neshat et al., 2011; Salehi et al., 2014) – 0.00235 as given in the Figure 5.8. Besides the point of this scientific value validation proof, Figure 6.8 in the following also have supported this validation of generated rule set created by ANFIS machine learning algorithm on MATLAB.

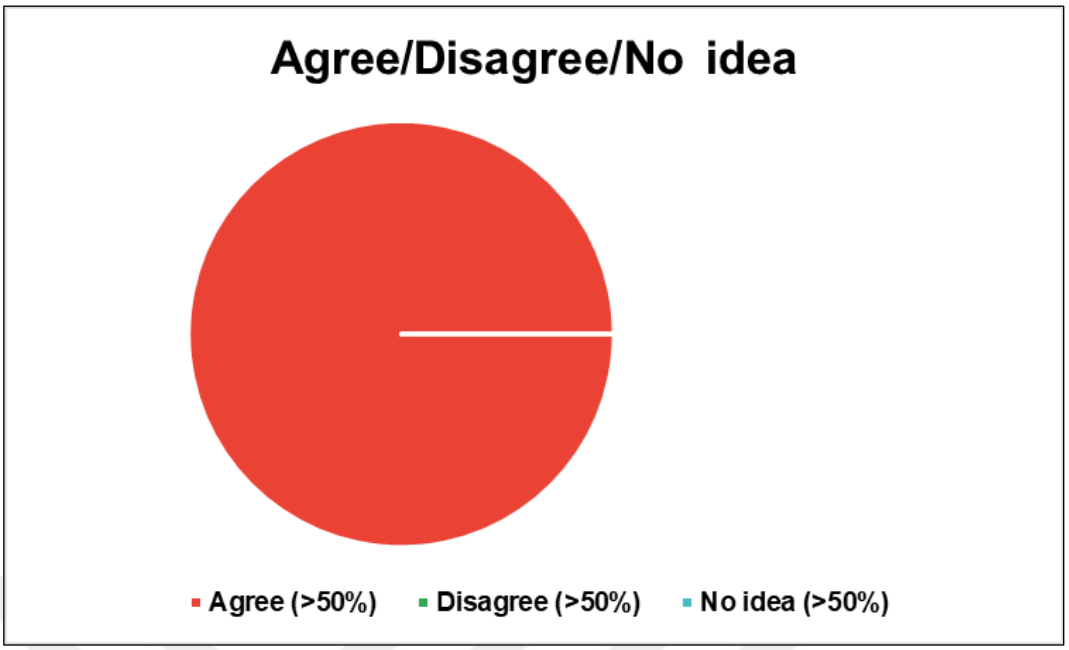

Figure 6.8 Choices that have had more than 50 percent

According to the findings in the Figure 6.8, there have been no rules which have got the answer with “Disagree”, and the choice with “No idea” in the ratio of more than 50 percent in the survey. Also, almost all rules (as in the Figure 6.5 – 28 out of 32) have been accepted by more than half of the software experts, who have taken the survey, in the area of software risk assessment and management. In other words, this software risk parameters rule set (Table 5.3), which has been generated by ANFIS – Adaptive Neuro-Fuzzy Inference System – on MATLAB has been operative for the real software projects according to more than half of the software developers, who have conducted the questionnaire. However, human beings are different from machines, and they usually couldn’t think straight logic. In addition, they have subjective judgement mentality in real life. Because of that, even though more than half the software experts (108 people in total) from 76 different companies (as listed in the Table 6.1 and as shown in the Figure 6.3 and Figure 6.4), some software experts (less than half) have had a negative opinion about several rules in this software risk parameters rule set. An illustration has been drawn about this situation and mentality as shown in the Figure 6.9 in the following.

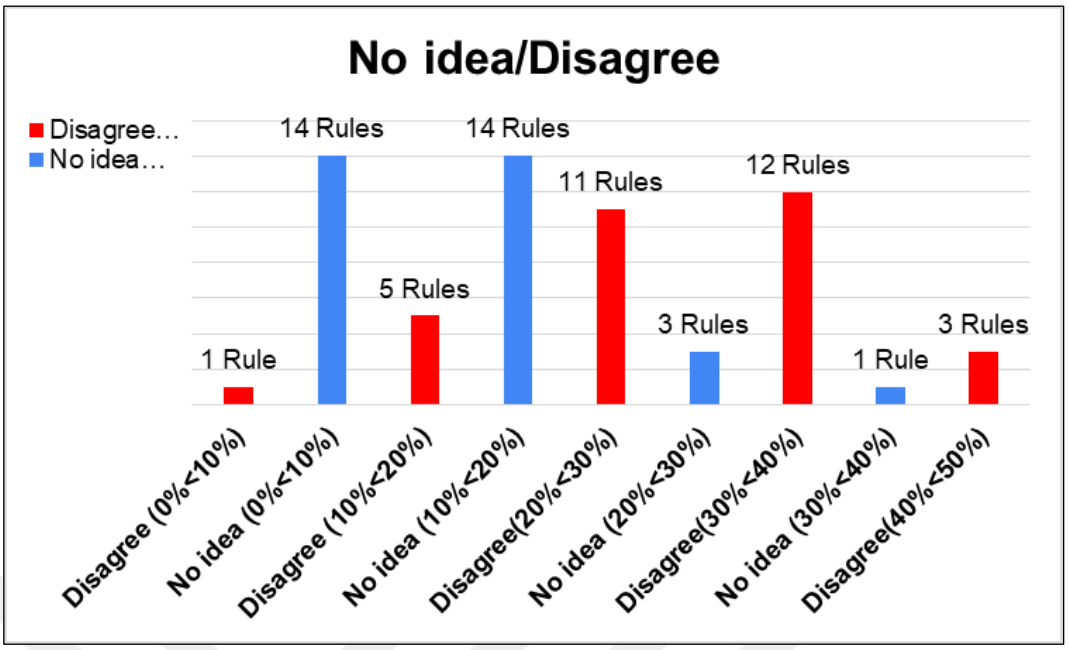

Figure 6.9 The number of rules based on the "disagree" and "no idea" choice

Figure 6.9 have been drawn in order to show the rule number (these rules have been derivated as explained in detail in chapter 5 and listed in the Table 5.3) based on the "disagree" and "no idea" answers of the software developers, who have conducted the survey. Moreover, this figure have been the combination of Figure 6.6 and Figure 6.7. In addition, based on the Figure 6.9, all rules in the derivated rule set, which has been formed and built with Adaptive Neuro-Fuzzy Inference System (ANFIS) on MATLAB, have had "disagreement" and "undefinite" answers with the ratio less than 50 percent – in other words, almost all rules have been admitted by software experts – think in fuzzy logic (not straight forward) as placed in ANFIS method – in the survey.

In the light of the results and findings shown and explained in this chapter (Chapter 6), it might be said that ANFIS machine learning algorithm integrated with fuzzy structured artificial intelligence – behave as human beings – could be used and applied effectively for building an original software risks parameter rule set (an integrated software risks framework) on MATLAB in software risk assessment and management methodology with high success rate and high subjective agreement.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK

In this doctoral dissertation, the background study and literature review about the software risk assessment and management have been done at first of the work. In addition, for this study, 50 science citation indexed articles published from the years 1978 to 2017 for project risks; 21 science citation indexed articles published from the years 1998 to 2017 for software project risks; and 17 science citation indexed articles published from the years 2001 to 2017 for software project risks based on developers' performance have been read and analysed. Also, based on these articles, 13 main papers – listed in the Table 4.1 in the chapter 4 – have been handled for risk parameters. Moreover, 50 software risk parameters (Table 4.2) under 6 main titles – Organizational Environment, Users, Requirements, Project Complexity, Team, and Planning and Control – have been determined for building software risks set. Furthermore, 18 original linguistic software risk rules – as given in the Table 4.4 – have been written based on 15 software risk factors which have been listed in the Table 4.3. With these rules, an application has been designed and developed in order to run Fuzzy Logic structure on MATLAB with Python programming language for the user interface. Moreover, this demo (explained in the section 4.2.2) has showed the fuzzy approach could work for software risk rules with software risk parameters on MATLAB.

For this doctoral dissertation, a real application has been designed and developed based on 36 software risk rules with 23 different software risk parameters – listed in the Table 5.1 in the chapter 5 – from an article published in IEEE in 2010 (Lamersdorf et al., 2010). In addition, this rule set has been set for the machine learning algorithm with integrated fuzzy artificial intelligence mechanism – Adaptive Neuro-Fuzzy Inference System (ANFIS). Moreover, the software risk rule set has been converted into numerical values (Table 5.2) for the ANFIS implementation, and have been loaded into MATLAB. Afterwards, based on ANFIS algorithm, 32 new software risk rules (Table 5.3) have been created from the rule set listed in the Table 5.1. Furthermore, both the rule set in the Table 5.1 and also the derivated original software risks rule set given in the Table 5.3 have been valid and accurate based on the low testing average

error in ANFIS configuration on MATLAB – for the rule set available on hand in the Table 5.1, 0.000018 given in the Figure 5.5; for the derivated software risks parameter rule set in the Table 5.3, 0.00235 shown in the Figure 5.8 (Doufesh et al., 2016; Jassar et al., 2009; Kaynak et al., 2015; Mohammed et al., 2017; Neshat et al., 2011; Salehi et al., 2014). In the light of this situation, ANFIS (run in fuzzy logic as human beings thinking and mentality) machine learning algorithm on MATLAB implementation has been resulted that machines could create several rules not to seen, recognized and told by the human beings in the software risk assessment and management with valid software risk parameter rules and accurate fuzzy models.

Additionally in this doctoral dissertation, a survey has been prepared in order to show subjective accuracy ratings of 32 automatically derivated new software risk rules listed in the Table 5.3 in the chapter 5 among software experts and developers from software companies in real world. In addition, this survey has been applied with the help of Google Forms on internet. Also, it has been has been applied by 108 different software experts from 76 different software companies (start-up and small companies, big and corporate companies, public enterprises, private institutions, national companies, international companies, universities, etc.) – explained in detail in the section 6.2 in the chapter 6. In the light of the survey results, almost all participants, who have taken the survey, have accepted the derivated rules (Table 5.3) in the ratio of more than 50 percent in total as seen in the Figure 6.5 and Figure 6.8. That means, the validation of these new and original developed software risk rules which have been generated by ANFIS method on MATLAB have been proven as well by the software experts' and developers' opinions and choses in the given survey besides the average testing error magnitude (Doufesh et al., 2016; Jassar et al., 2009; Kaynak et al., 2015; Mohammed et al., 2017; Neshat et al., 2011; Salehi et al., 2014).

For future work of this doctoral dissertation, several attachments may be done to increase originality of the derivated software risks parameters rule set listed in the Table 5.3 in the chapter 5. For instance, a case analysis of 50 software risk parameters listed in the Table 4.2 in the chapter 4 with several software companies about their software projects may be done. Thus, a new original software risk rule set has been

formed and built more free from literature comparing with the rule set listed in the Table 5.1. In the result of that, a new software risks rule database may be built, and the derivated rule set generated from this literature free software risks rule set by ANFIS (Adaptive Neuro-Fuzzy Inference System) algorithm on MATLAB may take place more original in the literature. Moreover, a survey about software risks and their linguistic rules may be applied face to face not by on internet to get more reliable data and feedback. Furthermore, genetic algorithms may be added into ANFIS method (hybrid method and implementation) in order to determine the order of importance for the outputs – software risk parameter rules. In addition, the study may be enhanced and improved to do dynamic operations for adding or deleting software risk parameters and for changing parameters' weights and for changing rules in the software risk assessment model and framework.

"You can't deal with the interaction which you don't quantify." This explanation which is professed to be said by Peter DRUCKER shows that the product hazard appraisal and the board under programming improvement measure must be estimated and assessed by substantial programming hazard rules dependent on the reasonable and target programming hazard boundaries. With the commitment of ANFIS (Adaptive Neuro-Fuzzy Inference System) AI calculation, a legitimate programming hazard boundaries rule set has been created, and its approval has been demonstrated by both low normal testing mistake of ANFIS design on MATLAB, and furthermore by programming specialists' conclusions and answers in the study for this standard set. Consequently, some dependable information have been indicated and decided so this product hazard evaluation and the board interaction could be designed by profiting with these reliable outcomes. As indicated by the consequences of the substantial programming hazard boundaries rule set, "labor", "time" and "cost" what are the principle assets of programming advancement cycle will be utilized all the more viably. And afterward, the advantages of the "Software Risk Assessment" under "Software Engineering" might be seen more substantial.

REFERENCES

Adler, M., & Dumas, B. (1984). Exposure to currency risk: Definition and measurement. *Financial Management, 13* (2), 41-50.

Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis, 22* (5), 717-727.

Andersen, B., & Fagerhaug, T. (2006). *Root cause analysis, simplified tools and techniques.* North America: ASQ Quality Press.

Arnuphaptrairong, T. (2011). Top ten lists of software project risks : Evidence from the literature survey. In *Proceedings of the International MultiConference of Engineers and Computer Scientists, 1*, 1-6.

Arora, N., & Saini, J. R. (2013). Time series model for bankruptcy prediction via adaptive neuro-fuzzy inference system. *International Journal of Hybrid Information Technology, 6* (2), 51-64.

Arora, N., & Saini, J. R. (2014). Bankruptcy Prediction of Financially Distressed Companies using Independent Component Analysis and Fuzzy Support Vector Machines. *International Journal of Research in Computer and Communication Technology, 3* (2), 924-931.

Asklund, U., & Bendix, L. (2002). A study of configuration management in open source software projects. *IEEE Proceedings - Software, 149* (1), 40-46.

Baggelaar, H. (2008). *Evaluating programmer performance visualizing the impact of programmers on project goals.* M.Sc Thesis, University of Amsterdam, Amsterdam.

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 Sciences,* 5442-5451.

Bannerman, P. L. (2015). A reassessment of risk management in software projects. *Handbook on Project Management and Scheduling, 2,* 1119-1134.

Becker, J., & Dai, R. W. (2011). Understanding IT project risks as disturbances to digital ecosystems. *International Conference on Management of Emergent Digital EcoSystems (MEDES'11),* 137-142.

Bedford, T., & Cooke, R. (2001). *Probabilistic risk analysis: Foundations and methods.* United Kingdom: Cambridge University Press.

Behbood, V., & Lu, J. (2011). Intelligent financial warning model using fuzzy neural network and casebased reasoning. *IEEE Symposium on Computational Intelligence for Financial Engineering and Economics, 1* (6), 11-15.

Birant, K. U., Işık, A. H., Batar, M., Akarsu, H. B., & Tektaş, A. B. (2019). Risk assessment and management method for distributed software development projects with "fuzzy approach". *International Journal of Computer Science and Software Engineering (IJCSSE), 8* (6), 133-139.

Biswas, G., Leelawong, K., Schwartz, D., & Vye, N. (2007). Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence, 19* (3), 363-392.

Boehm, B. W. (1989). *Software risk management.* New York: IEEE Press.

Boehm, B. W. (1991). Software risk management: principles and practices. *IEEE Software, 8* (1), 32-41.

Brasiliano, A. (2009). *Método Brasiliano avançado - Gestão e análise de risco corporativo.* Brazil: Sicurezza.

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,* 1-10.

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,* 1-11.

Carlsson, C., & Fuller, R. (2003). A fuzzy approach to real option valuation. *Fuzzy Sets and Systems, 139* (2), 297-312.

Carr, M. J., Konda, S. L., Monarch, I., Ulrich, F. C., & Walker, C. F. (1993). *Taxonomy-based risk identification.* United States: Software Engineering Institute.

Catherine, A. O., Sunday, A. O., & Kayode, A. B. (2013). Design of a neurofuzzy based model for active and collaborative online learning. *International Journal of Education and Research, 1* (10), 1-22.

Chelgani, S. C., Hart, B., Grady, W. C., & Hower, J. C. (2011). Study relationship between inorganic and organic coal analysis with gross calorific value by multiple regression and anfis. *International Journal of Coal Preparation and Utilization, 31* (1), 9-19.

Chen, M. Y. (2013). A hybrid ANFIS model for business failure prediction utilizing particle swarm optimization and subtractive clustering. *Information Sciences, 220,* 180-195.

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. *The DATA BASE for Advances in Information Systems, 41* (1), 8-35.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics, 58* (2), 345-363.

Coleman, J. L. (2002). *Risks and wrongs.* United Kingdom: Oxford University Press.

Copeland, B. J., & Proudfoot, D. (1996). On Alan Turing's anticipation of connectionism. *Synthese, 108,* 361-377.

De Bakker, K., Boonstra, A., & Wortmann, H. (2010). Does risk management contribute to IT project success? A meta-analysis of empirical evidence. *International Journal of Project Management, 28* (5), 493-503.

Dey, P. K., Kinch, J., & Ogunlana, S. O. (2007). Managing risk in software development projects: A case study. *Industrial Management & Data Systems, 107* (2), 284-303.

DoD, U. S. (2006). *Risk management guide for DoD acquisition.* USA: Department of Defense.

Dorofee, A. J., Walker, J. A., Alberts, C. J., Higuera, R. P., & Murphy, R. L. (1996). *Continuous risk management guidebook.* United States: Software Engineering Institute.

Doufesh, H., Ibrahim, F., Ismail, N. A., & Ahmad, W. A. W. (2016). Adaptive neuro-fuzzy inference system for predicting alpha band power of EEG during muslim prayer (SALAT). *Biomedical Engineering: Applications, Basis and Communications, 28* (6), 6-15.

Duarte, C. B., Faria, J. P., & Raza, M. (2012). PSP PAIR: Automated personal

software process performance analysis and improvement recommendation. *2012 Eighth International Conference on the Quality of Information and Communications Technology,* 131-136.

Efe, P., & Demirors, O. (2019). A change management model and its application in software development projects. *Computer Standards & Law, 66*.

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,* 945-954.

Evans, C. L. (2018). *Broad band: The untold story of the women who made the internet.* USA: Penguin Books.

Ezell, B. C., Farr, J. V., & Wiese, I. (2000). Infrastructure risk analysis model. *Journal of Infrastructure Systems, 6* (3), 114-117.

Fairley, R. (1994). Risk management for software projects. *IEEE Software, 11* (3), 57-67.

Fakhar, M. Z., Abbas, M., & Waris, M. (2013). Risk management system for ERP software project. *2013 Science and Information Conference,* 223-228.

Fan, C. F., & Yu, Y. C. (2004). BBN-based software project risk management. *Journal of Systems and Software, 73* (2), 193-203.

Feng, N., Li, M., & Gao, H. (2009). A software project risk analysis model based on evidential reasoning approach. *2009 WRI World Congress on Software Engineering,* 224-228.

Fu, Y., Li, M., & Chen, F. (2012). Impact propagation and risk assessment of

requirement changes for software development projects based on design structure matrix. *International Journal of Project Management, 30* (3), 363-373.

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,* 435-444.

Gerrard, P., & Thompson, N. (2002). *Risk-based E-business testing*. USA: Artech House.

Gilchrist, W. (1993). Modelling failure modes and effects analysis. *International Journal of Quality & Reliability Management, 10* (5), 16-23.

Giovanis, E. (2012). Study of Discrete Choice Models and Adaptive Neuro-Fuzzy Inference System in the Prediction of Economic Crisis Periods in USA. *Economic Analysis & Policy, 42* (1), 79-95.

Haimes, Y. Y. (1981). Hierarchical holographic modeling. *IEEE Transactions on Systems, Man, and Cybernetics, 11* (9), 606-617.

Hájek, P. (1998). *Metamathematics of fuzzy logic.* Dordrecht: Kluwer Academic Publishers.

Hall, E. M. (1998). *Managing risk: methods for software systems development.* United Kingdom: Pearson Education.

Hall, T., Wilson, D., Rainer, A., & Jagielska, D. (2007). The neglected technical skill? *2007 ACM SIGMIS CPR Conference on Computer Personnel Research: The Global Information Technology Workforce,* 196-202.

Han, W. M., & Huang, S. J. (2007). An empirical analysis of risk components and performance on software projects. *Journal of Systems and Software, 80* (1), 42-50.

Heldman, K. (2010). *Project manager's spotlight on risk management.* United States: John Wiley & Sons.

Hill, T., Marquez, L., O'Connor, M., & Remus, W. (1994). Artificial neural network models for forecasting and decision making. *International Journal of Forecasting, 10* (1), 5-15.

Hillson, D. (2002). The Risk Breakdown Structure (RBS) as an aid to effective risk management. *5th European Project Management conference*, 1-11.

Hoermann, S., Aust, M., Schermann, M., & Krcmar, H. (2012). Comparing risks in individual software development and standard software implementation projects: A delphi study. *45th Hawaii International Conference on System Sciences*, 4884-4893.

Hulebak, K. L., & Schlosser, W. (2002). Hazard analysis and critical control point (haccp) history and conceptual overview. *Risk Analysis, 22* (3), 547-552.

Inyang, U. G., & Joshua, E. E. (2013). Fuzzy clustering of students' data repository for at-risks students identification and monitoring. *Computer and Information Science, 6* (4), 37-50.

Iraji, M. S., Aboutalebi, M., Seyedaghaee, N. R., & Tosinia, A. (2012). Students classification with adaptive neuro-fuzzy. *International Journal of Modern Education and Computer Science, 4* (7), 42-49.

Jang, J. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics, 23* (3), 665-685.

Jang, J. S. R., Sun, C. T., & Mizutani, E. (1997). Neuro-fuzzy and soft computing - a computational approach to learning and machine intelligence. *IEEE Transactions on Automatic Control, 42* (10), 1482-1484.

Jassar, S., Liao, Z., & Zhao, L. (2009). Impact of data quality on predictive accuracy of ANFIS based soft sensor models. *IEEE International Conference on Soft Computing and Applications,* 1001-1006.

Jiang, J., & Klein, G. (2000). Software development risks to project effectiveness. *The Journal of Systems and Software, 52* (1), 3-10.

Jiang, J., Klein, G., & Discenza, R. (2001). Information systems success as impacted by risks and development strategies. *IEEE Transactions on Engineering Management, 48* (1), 46-55.

Johnson, J., & Mulder, H. (2016). CHAOS chronicles, focusing on failures and possible improvements in IT projects. *Journal of Systemics, Cybernetics and Informatics, 14* (5), 11-15.

Jurison, J. (1999). Software project management: The manager's view. *Communications of the Association for Information Systems, 2* (1), 17.

Kaynak, S., Evirgen, H., & Kaynak, B. (2015). Adaptive neuro-fuzzy inference system in predicting the success of student's in a particular course. *International Journal of Computer Theory and Engineering, 7* (1), 34-39.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A framework for identifying software project risks. *Communications of the ACM, 41* (11), 76-83.

Kelly, B., & Haddad, H. M. (2012). Metric techniques for maintenance programmers in a maintenance ticket environment. *Journal of Computing Sciences in Colleges, 28* (2), 170-178.

Kenarangui, R. (1991). Event-tree analysis by fuzzy probability. *IEEE Transactions on Reliability, 40* (1), 120-124.

Khan, S. (2009). An approach to facilitate software risk identification. *2009 2nd International Conference on Computer, Control and Communication,* 1-5.

Kontio, J. (2001). *Software engineering risk management: a method, improvement framework, and empirical evaluation.* PhD Thesis, Helsinki University of Technology, Espoo.

Kumar, C., & Yadav, D. K. (2015). A probabilistic software risk assessment and estimation model for software projects. *Procedia Computer Science, 54*, 353-361.

Lamersdorf, A., Munch, J., Torre, A. F., Sánchez, C. R., Heinz, M., & Rombach, D. (2010). A rule-based model for customized risk identification in distributed software development projects. *2010 5th IEEE International Conference on Global Software Engineering,* 209-218.

Lee, K., Joshi, K., & Kim, Y. (2008). Person-job fit as a moderator of the relationship between emotional intelligence and job performance. *2008 ACM SIGMIS CPR Conference on Computer Personnel Doctoral Consortium and Research,* 70-75.

Lezzoni, L. K. (1997). The risks of risk adjustment. *JAMA Journal of the American Medical Association, 278* (19), 1600-1607.

Liu, L., & Horowitz, E. (1989). A formal model for software project management. *IEEE Transactions on Software Engineering, 15* (10), 1280-1293.

Lyytinen, K., Mathiassen, L., & Ropponen, J. (1996). A framework for software risk management. *Journal of Information Technology, 11* (4), 275-285.

Maslow, A. H. (1954). The instinctoid nature of basic needs. *Journal of Personality, 22*, 326-347.

McManus, J. (2003). *Risk management in software development projects.* United

Kingdom: Butterworth-Heinemann.

Min, W., Jun, Z., & Wei, Z. (2017). The application of fuzzy comprehensive evaluation method in the software project risk assessment. *2017 International Conference on Management Engineering, Software Engineering and Service Sciences,* 76-79.

Mohammed, H., Hameed, I., & Seidu, R. (2017). Adaptive neuro-fuzzy inference system for predicting norovirus in drinking water supply. *2017 International Conference on Informatics, Health & Technology*, 1-6.

Neshat, M., Deli, A., Masoumi, A., & Sargolzae, M. (2011). A comparative study on anfis and fuzzy expert system models for concrete mix design. *International Journal of Computer Science Issues, 8* (3), 196-210.

Olaru, C., & Wehenkel, L. (2003). A complete fuzzy decision tree technique. *Fuzzy Sets and Systems, 138* (2), 221-254.

Pfleeger, S. L., Hatton, L., & Howell, C. C. (2001). *Solid software.* USA: Prentice Hall.

Powers, D. M. W., & Turk, C. C. R. (2012). *Machine learning of natural language.* Berlin: Springer Science & Business Media.

Pressman, R. S. (2000). What a tangled web we weave. *IEEE Software, 17* (1), 18-21.

Qinghua, P. (2009). A model of risk assessment of software project based on grey theory. *4th IEEE International Conference on Computer Science & Education,* 538-541.

Rakos, J. J. (1990). *Software project management for small to medium sized projects.* USA: Prentice Hall.

Ramler, R., Klammer, C., & Natschläger, T. (2010). The usual suspects: A case study on delivered defects per developer. *2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement,* 1-4.

Raz, T., Shenhar, A. J., & Dvir, D. (2002). Risk management, project success, and technological uncertainty. *R&D Management, 32* (2), 101-109.

Renn, O. (2004). Perception of risks. *Toxicology Letters, 149* (1), 405-413.

Robles, G., Amor, J.J., Gonzalez-Barahona J.M., & Herraiz, I. (2005). Evolution and growth in large libre software projects. *Eighth International Workshop on Principles of Software Evolution (IWPSE'05),* 165-174.

Rook, P. (1986). Controlling software projects. *Software Engineering Journal, 1* (1), 7-16.

Ross, T. J. (2017). *Fuzzy Logic with engineering applications.* United Kingdom: John Wiley & Sons Inc.

Salehi, K., Khazraee, S. M., Hoseini, F. S., & Mostafazadeh, F. K. (2014). 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, 5* (3), 290-293.

Salmeron, J. L., & Lopez, C. (2012). Forecasting risk impact on ERP maintenance with augmented fuzzy cognitive maps. *IEEE Transactions on Software Engineering, 38* (2), 439-452.

Sawyer, S., & Guinan, P. J. (1998). Software development: Processes and performance. *IBM Systems Journal, 37* (4), 552-569.

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*, 1317-1326.

Shapiro, F. (2000). Origin of the term software: Evidence from the JSTOR electronic journal archive. *IEEE Annals of the History of Computing. 22* (4), 69-71.

Shen, Q., & Chouchoulas, A. (2002). A rough-fuzzy approach for generating classification rules. *Pattern Recognition, 35* (11), 2425-2438.

Shoorehdeli, M. A., Teshnehlab, M., & Sedigh, A. K. (2006). A novel training algorithm in ANFIS structure. *2006 American Control Conference,* 6-16.

Silva, S. (2011). *Proposta de tratamento de fatores de riscos em desenvolvimento de software para uma organização no setor público.* PhD Thesis, Federal University of Permambuco, Recife.

Smith, M. (1989). The people risks. *Computer Law & Security Review, 4* (6), 2-6.

Sonchan, P., & Ramingwong, S. (2014). Top twenty risks in software projects: A content analysis and Delphi study. *11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON),* 1-6.

Stellman, A., & Greene, J. (2005). *Applied software project management.* USA: O'Reilly Media, Inc.

Stoneburner, G., Goguen, A., & Feringa, A. (2003). *Risk management guide for information technology systems and underlying technical models for information technology security.* USA: Diane Pub Co.

Sudhakar, G. P. (2012). A model of critical success factors for software projects. *Journal of Enterprise Information Management, 25* (6), 537-558.

Tanaka, H., Fan, L. T., Lai, F. S., & Toguchi, K. (1983). Fault-Tree analysis by fuzzy probability. *IEEE Transactions on Reliability, 32* (5), 453-457.

Tang, A. G., & Wang, R. L. (2010). Software project risk assessment model based on fuzzy theory. *2010 IEEE International Conference on Computer and Communication Technologies in Agriculture Engineering, 2,* 328-330.

Thelwall, M., & Kousha, K. (2016). Academic software downloads from google code: Useful usage indicators. *Information Research, 21* (1), 709.

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,* 1-4.

Trigo, T. R., Gusmão, C., & Lins, A. (2008). CBR risk - risk identification method using case based reasoning. *5th International Conference on Information Systems and Technology Management.*

Tukey, J. W. (1958). The teaching of concrete mathematics. *The American Mathematical Monthly, 65* (1), 1-9.

Van Loon, H. (2007). A management methodology to reduce risk and improve quality. *IT Professional, 9* (6), 30-35.

Walia, N., Singh, H., & Sharma, A. (2015). ANFIS: adaptive neuro-fuzzy inference system - a Survey. *International Journal of Computer Applications, 123* (13), 32-38.

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM, 47* (4), 68-73.

Wallace, L., Keil, M., & Rai, A. (2004a). Understanding software project risk: a cluster

analysis. *Information Management, 42* (1), 115-125.

Wallace, L., Keil, M., & Rai, A. (2004b). How software project risk affects project performance: an investigation of the dimensions of risk and an exploratory model. *Decision Sciences, 35* (2), 289-321.

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, 2,* 39-47.

Warkentin, M., Moore, R. S., Bekkering, E., & Johnston, A. C. (2009). Analysis of systems development project risks: An integrative framework. *The DATA BASE for Advances in Information Systems, 40* (2), 8-27.

Westermann, D. (2012). A generic methodology to derive domain-specific performance feedback for developers. 3*4th International Conference on Software Engineering,* 1527-1530.

Xiaosong, L., Shushi, L., Wengun, C., & Songjiang, F. (2009). The application of risk matrix to software project risk management. *2009 International Forum on Information Technology and Applications,* 480-483.

Yu, A. G. (2009). Software Crisis, What Software Crisis. *2009 International Conference on Management and Service Science,* 1-4.

Zhang, S., Wang, Y., & Xiao, J. (2008). Mining individual performance indicators in collaborative development using software repositories. *15th Asia-Pacific Software Engineering Conference,* 247-254.

Zanganeh, T., Rabiee, M., & Zarei, M. (2011). Applying adaptive neuro-fuzzy model for bankruptcy prediction. *International Journal of Computer Applications, 20* (3), 15-21.

APPENDICES

APPENDIX 1: The Survey

The survey has been written and prepared in English in the Google Forms, and also, it has been taken and conducted via internet. It has been formed as in the following.

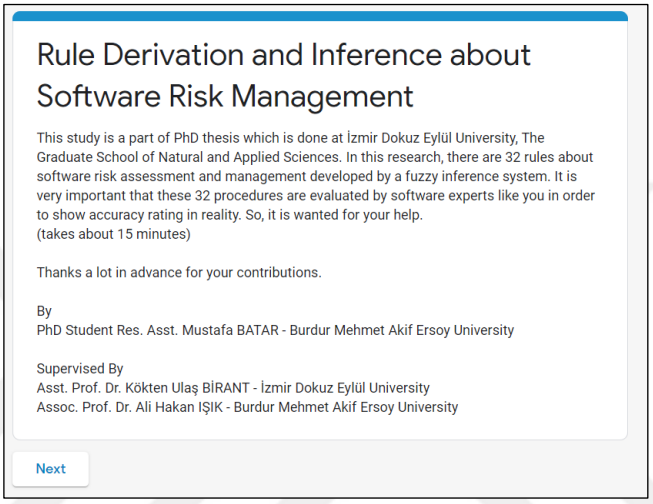

Figure A.1 Introduction of the survey

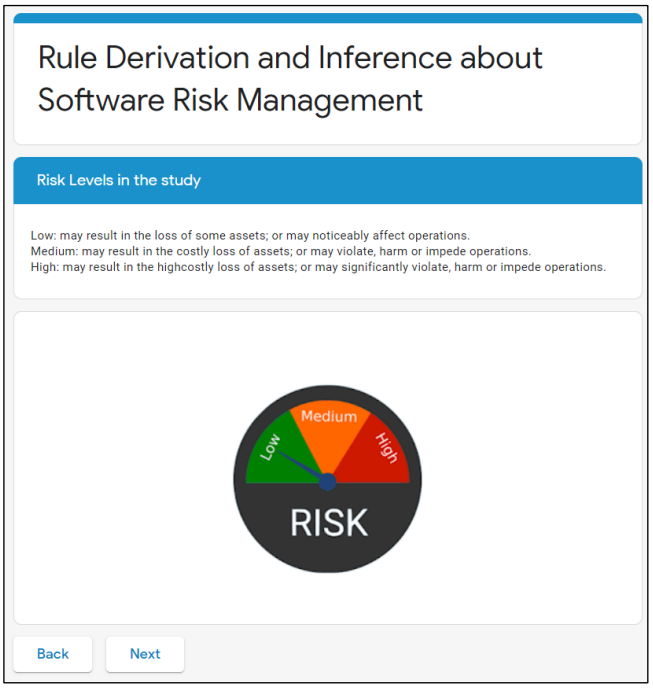

Figure A.2 Second page of the survey: Risk definition

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

Rule 6 - Cultural difference is medium & Quality problems is medium & Communication problems is low & Time zone difference is medium & Communication infrastructure is medium → phase=requirements is medium

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

Rule 19 - Lack of trust is high & Transparency is medium & Complexity is low & Coupling is low & Cultural differences is medium → Time pressure is medium

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

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

Agree

Disagree

No idea

Rule 30 - Formality is medium & Size is medium & Maturity is low & Cultural difference is low & Lack of trust is low → Process maturity is medium

Agree

Disagree

No idea

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

Agree

Disagree

No idea

Rule 32 - Productivity drop is high & Complexity is medium & Process maturity is low & Transparency is low & Personal relationships is medium → Coordination problem is high

Agree

Disagree

No idea

Figure A.3 Thanks page of the survey