A DECISION ANALYSIS APPROACH FOR SELECTING SOFTWARE DEFECT PREDICTION METHOD IN THE EARLY PHASES

ERKEN AŞAMALARDA YAZILIM HATA TAHMİN YÖNTEMİ SEÇİMİ İÇİN BİR KARAR ANALİZİ YAKLAŞIMI

RANA ÖZAKINCI

ASSOC. PROF. DR. AYÇA KOLUKISA TARHAN

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Doctor of Philosophy

in Computer Engineering.

2022

To my family

ABSTRACT

A DECISION ANALYSIS APPROACH FOR SELECTING SOFTWARE DEFECT PREDICTION METHOD IN THE EARLY PHASES

Rana ÖZAKINCI

Doctor of Philosophy, Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ayça KOLUKISA TARHAN

June 2022, 216 pages

Considering that software usage rates have increased, it is inevitable for end-users to prefer high-quality software products. Undoubtedly, one of the most important quality indicators of a software product is its defect rate. With the widespread use of methods and tools that support estimation tasks in software engineering, the interest in software defect prediction is increasing. Currently, most defect prediction models are built using the metrics from the coding phase. This situation leads to the inability to process the information belonging to the early stages of the software development life cycle such as requirements analysis or design, thus not being able to benefit from preventive actions such as cost reduction and effective resource planning in the early stages. Eventually, it becomes important for stakeholders to build the desired defect prediction model as early as possible and to use it throughout the software development life cycle. When the proliferation of methods of data science in software engineering is combined with the shortage of knowledge to use them in industry, an important need arises to guide

practitioners in selecting the best-fit methods by considering their specific needs. This thesis presents research aimed at addressing the method selection problem in software defect prediction during the early phases of the life cycle by using a formal decision analysis process. A two-phase decision analysis approach was proposed that is structured using a decision tree and multi-criteria decision analysis (MCDA) methodologies. In doing so, an extensive literature review was conducted to obtain a general view of the characteristics and usefulness of Early Software Defect Prediction (ESDP) models reported in scientific literature. As a result, the most preferred prediction methods, metrics, datasets, and performance evaluation methods, as well as the addressed SDLC phases were highlighted. Accordingly, the alternatives to be evaluated in the decision analysis and the criteria that may have an impact on the decision of method selection were systematically determined. To strengthen the knowledge, two different expert opinion surveys were conducted. Besides, to manage the operation of the decision analysis process, a questionnaire is proposed to reveal stakeholder needs and dataset characteristics. After, several case studies were performed to investigate the trustworthiness of the proposed approach with selected SDP methods using public datasets. The most convenient methods proposed by the decision analysis are Naïve Bayes (NB), Decision Tree (DT), and Fuzzy Logic-based methods for the case studies. It is concluded that the results of the decision analysis are consistent with both the results of the empirical evidence of the experiments conducted in the thesis and the results reported in the scientific literature. Overall, the presented approach could be useful in helping software practitioners decide which SDP method is advantageous by revealing their specific requirements for the software projects and associated defect data. While the results of this thesis provide guidance for future research on the context of ESDP, further studies on different software projects are necessary in order to expand knowledge prior to having decisions that are more reliable.

Keywords: Defect Prediction, Early Phases, Early Software Defect Prediction, Method Selection, Decision Analysis, Multi Criteria Decision Analysis, Fuzzy TOPSIS

ÖZET

ERKEN AŞAMALARDA YAZILIM HATA TAHMİN YÖNTEMİ SEÇİMİ İÇİN BİR KARAR ANALİZİ YAKLAŞIMI

Rana ÖZAKINCI

Doktora, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Doç. Dr. Ayça KOLUKISA TARHAN

Haziran 2022, 216 sayfa

Dünyada yazılım kullanım oranlarının günden güne arttığı göz önüne alındığında, son kullanıcıların kaliteli yazılım ürünlerini tercih etmek istemesi yadsınamaz bir gerçektir. Bir yazılım ürününün en önemli kalite göstergelerinden biri de hata oranıdır. Yazılım mühendisliğinde tahmin görevlerini destekleyen yöntem ve araçların yaygınlaşmasıyla birlikte yazılım hata tahminine olan ilginin arttığı bilinmektedir. Güncel durumda, çoğu hata tahmin modeli, kodlama aşamasından elde edilen metrikler kullanılarak oluşturulmaktadır. Bu durum, yazılım geliştirme yaşam döngüsünün gereksinim analizi veya tasarımı gibi erken aşamalarına ait bilgilerin işlenememesine, dolayısıyla erken aşamalarda maliyet düşürme ve etkin kaynak planlaması gibi önleyici faaliyetlerden yararlanılamamasına yol açmaktadır. Paydaşlar için, hata tahmin modelini mümkün olduğunca erken oluşturmaları ve yazılım geliştirme yaşam döngüsü boyunca kullanmaları önemli hale gelir. Yazılım mühendisliğinde veri bilimi yöntemlerinin çoğalması, fakat bunları sektörde kullanmak için bilgi ve uzmanlığın yeterli olmadığı göz önünde bulundurulduğunda, paydaşların proje özelindeki ihtiyaçlarını göz önünde bulundurarak en uygun hata tahmin yöntemini seçme konusunda rehberlik etmek için bir ihtiyacın ortaya çıktığı görülmüştür. Bu tez, bir karar analizi süreci kullanarak yaşam döngüsünün ilk aşamalarında yazılım hata tahmininde yöntem seçimi problemini ele

almayı amaçlayan bir araştırma sunmaktadır. Bu doğrultuda, karar ağacı ve çok kriterli karar analizi (İng. MCDA) metodolojileri kullanılarak yapılandırılmış iki aşamalı bir karar analizi yaklaşımı önerilmiştir. Öncelikli olarak, literatürde bildirilen Erken Aşama Yazılım Hata Tahmini (İng. ESDP) modellerinin özellikleri ve kullanışlılığı hakkında genel bir görüş elde etmek için kapsamlı bir literatür taraması yapılmıştır. Bu çalışma ile literatürde erken aşamada hata tahmini konusunda en çok tercih edilen tahmin yöntemleri, metrikler, veri setleri ve performans değerlendirme kriterleri analiz edilmiştir. Buna göre karar analizinde değerlendirilecek alternatifler ve yöntem seçimi kararına etki edebilecek kriterler sistematik olarak belirlenmiştir. Literatürde elde edilen bilgileri güçlendirmek için iki farklı uzman görüşü anketi yapılmıştır. Ayrıca, karar analizi sürecinin işleyişini yönetmek için paydaş ihtiyaçlarını ve veri seti özelliklerini ortaya çıkarmaya yarayan bir anket önerilmiştir. Daha sonra, karar analizi yaklaşımı tarafından önerilen tahmin yöntemlerinin doğruluğunu ve güvenilirliğini araştırmak için erişime açık veri kümeleri üzerinde birkaç vaka çalışması yapılmıştır. Karar analizi yaklaşımı tarafından önerilen en uygun yöntemler, üç farklı durum çalışması için sırasıyla Naive Bayes, Karar Ağacı ve Bulanık Mantık tabanlı yöntemlerdir. Karar analizi sonuçlarının hem tezde yapılan deneylerin ampirik kanıtlarının sonuçlarıyla hem de bilimsel literatürde raporlanmış sonuçlarla tutarlı olduğu gözlenmiştir. Genel olarak, sunulan karar analizi yaklaşımının, yazılım projeleri ve ilgili hata verileri için özel gereksinimleri ortaya çıkararak, yazılım uygulayıcılarına hangi hata tahmin yönteminin avantajlı olacağına dair ipucu vermesi açısından faydalı olacağı görülmüştür. Bu tezin sonuçları, erken aşamalarda yazılım hata tahmin kapsamında yapılacak gelecek araştırmalar için rehberlik sağlarken, karar analizi yaklaşımın sonuçlarının doğruluğunu arttırmak adına sektörden yazılım projeleri üzerinde daha fazla çalışma yapılması gerektiği düşünülmektedir.

Anahtar Kelimeler: Yazılım Hata Tahmini, Erken Aşama, Yöntem Seçimi, Karar Analizi, Çoklu Kriterli Karar Analizi, Bulanık TOPSIS

ACKNOWLEDGEMENTS

Looking back on my long journey, I would like to express my endless thanks to my advisor, Assoc. Prof. Dr. Ayça Kolukısa Tarhan, who has always been by my side as a great supporter. I want to thank her for her guidance, encouragement and feedback which has helped me tremendously in pursuing my PhD study.

I would like to express my gratitude to Assoc. Prof. Dr. Oumout Chouseinoglou and Assoc. Prof. Dr. Aysu Betin Can for their valuable feedback, contributions, and patience. I am also grateful to Prof. Dr. Pınar Karagöz and Assoc. Prof. Dr. Ebru Gökalp for their suggestions during my thesis defense presentation. I express my gratitude to the anonymous experts from both the academia and the industry, who contributed to the expert opinion studies and surveys.

I would like to thank TUBITAK BILGEM Software Technologies Research Institute (YTE) for supporting my academic studies. I also thank my colleagues who contributed to my work with their valuable suggestions and their participations when necessary. I am grateful to my dear friends for their endless support and motivation.

I would like to express my sincere thanks and love to my parents and sister, who have always supported me in my entire life and made the greatest contribution to my success.

I will be forever grateful to my dear husband, Mehmet, for helping me intellectually and emotionally from day one. This work would not have been possible without his patience and endless support. I have the deepest feelings for my lovely son, Özgün, who joined our family in the last years of my doctoral studies and gave me endless happiness.

Finally, I appreciate myself for continuing my devoted work with patience and pleasure during this long journey, seeing that I have taken a step towards knowing and understanding myself, and not losing my faith.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AUC	Area Under the Curve
AHM	Analytic Hierarchy Model
AHP	Analytic Hierarchy Process
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
ANP	Analytic Network Process
BBN	Bayesian Belief Network
BMMRE	Balanced Mean Magnitude of Relative Error
BN	Bayesian Network
CART	Classification and Regression Trees
CC	Closeness Coefficient
CMMI	Capability Maturity Model Integration
COCOMO	Constructive Cost Model
DA	Decision Analysis
DAG	Directed Acyclic Graph
DBMS	Database Management System
DCh	Data Characteristics
DQ	Data Quality
DT	Decision Tree
E	Expert
ELECTRE	Elimination and Choice Expressing the Reality
ERT	Experience of requirement team
ESDP	Early Software Defect Prediction
H	High
IEEE	Institute of Electrical and Electronics Engineers

FIS	Fuzzy Inference Systems
FNIS	Fuzzy Negative Ideal Solution
FNR	False Negative Rate
FPIS	Fuzzy Positive Ideal Solution
FPR	False Positive Rate
FRBC	Fuzzy Rule Based Classifier
KLOC	Thousands (Kilo) of Lines of Code
L	Low
LinR	Linear Regression
LOC	Lines of Code
LogR	Logistic Regression
M	Medium
MAE	Mean Absolute Error
MAPE	Mean Absolute Percent Error
MC	Model Construction
MCDA	Multi Criteria Decision Analysis
MCh	Method Characteristics
MDP	Metrics Data Program
ML	Machine Learning
MMRE	Mean Magnitude of Relative Error
NASA	The National Aeronautics and Space Administration
NB	Naïve Bayes
NRMSE	Normalized Root Mean Square Error
OO	Object Oriented
PC	Project Context
PD	Probability of Detection

PF	Probability of False Alarm
PhD	Doctor of Philosophy
PROMETHEE	Preference Ranking Organization METHod for Enrichment of Evaluations
QA	Quality Assurance
RFD	Requirement Fault Density
RIW	Review, Inspection and Walkthrough
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
RQ	Research Question
RS	Requirements Stability
SDL	Specification and Description Language
SDLC	Software Development Life Cycle
SDP	Software Defect Prediction
SLR	Systematic Literature Review
SM	Systematic Mapping
SVM	Support Vector Machines
TNR	True Negative Rate
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
TPR	True Positive Rate
UML	Unified Modeling Language
VH	Very High
VL	Very Low

1. INTRODUCTION

By nature, software systems are structures that are constantly growing and becoming increasingly complex. Research and development of techniques to facilitate and accelerate the successful completion of software projects have been ongoing since the 1970s. Ensuring software quality during and after software development is an indispensable task for those involved in software projects. Developing reliable software within limited time, budget and resources makes this task even more difficult. Still, project teams often spend at least 50% of development effort fixing defects, that could have been avoided or fixed at less cost [1]. In the complexity of the software development world, it is almost impossible to develop a software that is free of defects, but detecting existing defects in a timely manner and minimizing them are very important requirements for the product to be launched as reliable. It can be said that one of the most critical tasks of project management is to eliminate existing defects in the software, and even ensure that these errors do not occur, if possible.

Unfortunately, finding and fixing software defects are among the most expensive software development activities [2]. Often, detecting and fixing software defects after production are much costlier than detecting and fixing them early in the life cycle, such as requirements and design phases. According to Boehm, one of the first researchers to concretely exemplify this; if the cost of fixing a defect found at the requirement phase is expressed as 1 unit, the cost at the design phase is 3 - 6 units; 10 at the coding phase; increases to 15 - 70 units at the test phase; and 40 - 1000 units at the operation phase [3]. According to a NASA report that investigated cost escalation studies throughout the project life cycle in the literature [4], those ratios were determined as in Figure 1.1.

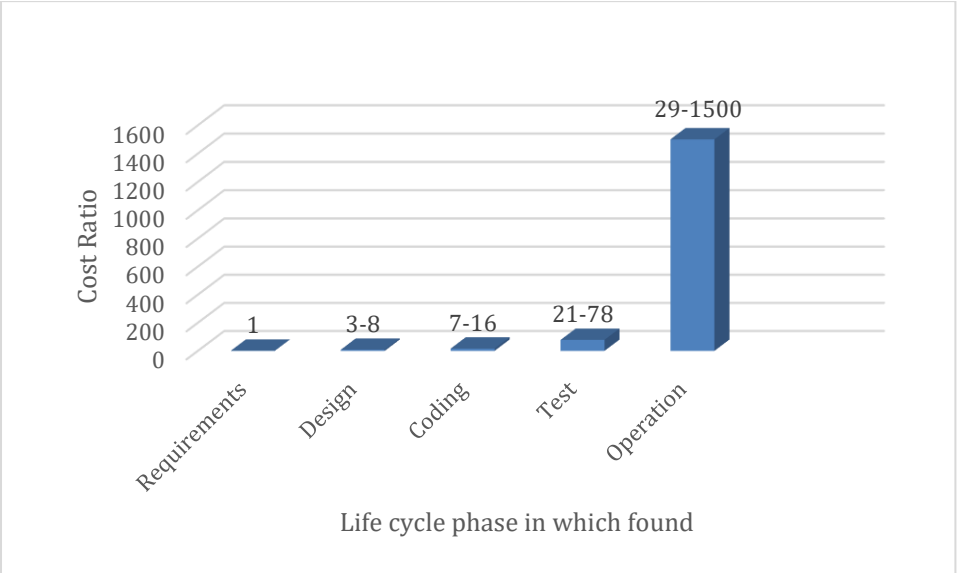


Figure 1.1. Relative Cost Ratio for Fixing Software Defects per Life Cycle Phase [3]

Obviously, as software evolves and grows, the cost of fixing existing or emerging defects increases dramatically. At the same time, it is crystal clear that the scope of these defects will also expand. Considering that defects that were not found on time and have moved on to later phases in the life cycle, especially during the coding phase, will spread to other modules of the project, much more changes and effort will be necessary to fix these defects. In addition, it is possible to say that the changes necessary to fix the common defects may also cause new ones in the software.

All these reasons show the importance of detecting and fixing defects as early as possible during the software life cycle, with the least cost and effort. Especially after the coding phase of the software, various test activities (unit testing, integration testing, automatic tests, etc.) can be carried out to detect defects related to the code. In addition, during the coding phase, code review activities carried out before the new developed code are merged to the version control system ensure that possible defects are noticed, and action can be taken. However, all these activities mentioned can be performed when the software moves to the coding phase, and there will be scenarios where the defects that emerged during the requirements analysis or design phases will be transferred to the code without being noticed.

At this point, a mechanism that systematically foresee the possible outcomes of the next phases of the software by making use of several existing metrics before the coding activities begin can be quite useful. As a matter of fact, predictive models are frequently used to evaluate development risks and improve quality throughout the life cycle of software development projects [5,6]. Such supportive models are the most important auxiliary mechanisms to predict problem areas early and make necessary corrections [7]. In order to form an idea about the quality of the software with software defect prediction throughout the software development life cycle (SDLC); it is intended for development, testing and management teams to anticipate defect-prone and/or defective parts of the software. Defect prediction models allow software developers to focus on defect-prone pieces of code, thus helping to reduce the potential for future defects [8]. Considering that software development companies can spend 50%-80% of their software development effort on testing practices [9], it is seen that research on defect prediction models is very critical in terms of cost savings in testing phases. Besides, it is reported that the analysis and prediction of software defects are also needed within the scope of project management [10,11]. In this context, it is recommended to use defect prediction models to evaluate project progress, plan project management activities, improve product quality and process management activities [12].

1.1. Software Defect Prediction (SDP) at Early Phases

Numerous defect prediction models have been presented in the literature over 40 years [13,14]. These studies mostly use various data processing methods and software metrics belonging to the late phases of the SDLC, such as testing or operational use. It is thought that the application of the prediction models during and after the coding phase of the software development will not be beneficial since it will be late to plan and control the cost-effectiveness activities [12].

On account of this, it can be appropriate to build and use software defect prediction (SDP) models earlier in software development life cycle, in terms of planning many corrective and preventive activities such as quality estimation, and effective resource, calendar and

cost planning [12]. Besides, it has been reported that the application of defect prediction models in the early phases of the SDLC, such as requirements analysis, design and/or early coding phase, will be more beneficial in many ways [15]. It plays a critical role in determining software quality, cost overrun, optimal development and testing strategy at an early stage. A useful approach for early evaluation in projects using Waterfall or V development model is to identify the number of defects in the requirements, design, or coding phases by verification and validation activities [16], and use this information to predict the number of defects in coding or testing phases [17]. In projects employing incremental or agile development, early evaluation includes identifying defects in early releases to predict defectiveness in later ones [18]. Cross-project defect prediction may also enable early evaluation if its underlying requirements regarding defect data across the projects are met [19]. In any case, foreseeing the defective parts of the software may provide preventive actions such as additional inspections and more comprehensive testing, therefore it helps to improve software process control and to ensuree higher software quality [12]. In addition, early SDP models will be able to help an effective decision-making process in the context of activities such as process improvement or trade-off analysis from the early stages of development [20,21].

Despite the aforementioned benefits, software defect prediction can be seen difficult to implement for a variety of reasons, such as context differences of software projects under development, software metrics that are needed to collect, behavioral dynamics of software team members, and different preferences of various software stakeholders. However, as data science is becoming widespread, there is a proliferation in methods and tools supporting prediction and estimation in software engineering, which makes selecting the best-fit methods important for early and effective use of such facilities. In addition, it is observed that the authors of SDP studies in literature are mostly academic, which means that the expertise to use and select prediction methods and supporting tools reside in academy rather than in industry. When the proliferation of methods of data science in software engineering is combined with the shortage of knowledge to use them in industry, an important need arises to guide practitioners in selecting and using the best-fit methods. Therefore, it might be a good solution to address method selection problem in software defect prediction by using a formal decision analysis process.

1.2. Goal and Research Questions

In this study, it is aimed to propose a decision analysis approach that can guide the determination of the most appropriate defect prediction method that can be used in software projects where defect prediction is desired from the early phases of the SDLC. To address the main purpose of the thesis study, the following research questions (RQs) were determined under five main headings.

RQ 1: What are the characteristics of early software defect prediction (ESDP) models?

- RQ1.1 Which types of datasets are used for performing the prediction? Identify the datasets that are used in the prediction models.
- RQ1.2 What are the development phases that originate metrics for the prediction models? Identify the phases that originate metrics as input to the prediction.
- RQ1.3 What are the entities that originate metrics for the prediction models? Characterize the software entities that are used in the models.
- RQ1.4 What are the attributes of each entity, which originate metrics for the prediction models? Categorize the attributes that are used in the models.
- RQ1.5 What are the software metrics that are used in the prediction models? Identify and categorize the software metrics related to each attribute of each entity used in the models.
- RQ1.6 What types of methods are used to build the prediction models? Identify and categorize the methods used in prediction models in the studies. Example methods include machine learning, fuzzy rule-based etc.
- RQ1.7 What are the contextual parameters reported in the prediction models? Gather the contextual information about the metric data included in the models for better revealing the factors that may affect the model construction.

RQ 2. Are models of ESDP successful and beneficial?

- RQ2.1 Which methods and measures are used for evaluating the performance of the models? Categorize the performance evaluation methods and metrics that are used for validating the models.
- RQ2.2 What are the performance values of the models based on the included SDLC phases that originate metrics for prediction? Gather the performance results of the studies with regard to SLDC phases in order to see the effects of the phase information to the prediction performance.
- RQ2.3 What are the benefits of early defect prediction as reported in the studies? Indicate the benefits or losses of using ESDP models if reported.

RQ 3. What is the current status of defect prediction applications in software companies in Turkey?

- RQ3.1. If software defect prediction is applied, how does the company operate it?
- RQ3.2. If the company is applying SDP, what are the advantages or disadvantages of applying it?
- RQ3.3. If the company is not applying SDP, what would be the benefits and/or challenges in applying SDP in your company?
- RQ3.4. Is there a need for guidance on software defect prediction from the early phases of SDLC?

RQ 4. How to select a method for early prediction of software defects?

- RQ4.1. What are the alternative methods for building ESDP models?
- RQ4.2. What are the criteria to consider when selecting a method for ESDP?
- RQ4.3. How should the most appropriate method be selected by evaluating the defined criteria?
- RQ4.4. How should we gather the characteristics of the project data and the needs of the users systematically?

RQ 5. How should we investigate the trustworthiness of the proposed SDP method selection approach through case studies?

- RQ5.1: Which SDP methods are primarily suggested by decision analysis approach?
- RQ5.2: Which SDP methods do perform better in execution?
- RQ5.3: Are there any difference between the results of RQ5.1 and RQ5.2?

1.3. Research Methods

Research methods describe the systematic processes that are carried out from the beginning to the end within the scope of the thesis studies and are necessary to reach the result. The research methods used in the thesis are explained below.

1.3.1. Literature Review

Systematic mapping (SM) studies are used to provide an overview of the research area [22]. Within the scope of systematic mapping, the relevant evidence is examined at a superficial level of detail, thus providing basic evidence that will contribute to possible systematic literature review studies and identifying areas that should be focused more in the field [23,24].

Systematic literature review (SLR) is a literature analysis method used for the purpose of determination, evaluation and interpretation of the available research on a specific topic. While individual studies contributing to the SLR are referred to as "primary studies"; systematic review itself is referred to as a “secondary study” [23,25]. SLR studies can be used to guide possible new studies by identifying gaps in the relevant field and presenting various suggestions [22,23,25].

1.3.2. Case Study

Case studies are empirical investigations of various contemporary phenomena in a real-life context [26]. The focus of case studies on making sense of context information is important in terms of evaluating the methods and tools used in software engineering in the industrial field [27].

1.3.3. Data Analysis

The data analysis method is used for both quantitative and qualitative research types. Within the scope of quantitative data analysis techniques, descriptive statistical analysis is generally performed. Mean value and standard deviation calculations and various visual graphics are frequently used to help understand the collected data [28].

1.3.4. Survey

Surveys are generally conducted with the participation of various distributed individuals, aiming to generalize from a sample to a population [29]. They often contain static questions that provide quantitative answers that are easy to analyze [30]. In addition, expert opinion surveys can be preferred for the evaluation of important factors and gathering the recommendations of the experts on the subject.

1.4. Contributions

The contributions made as a result of the studies conducted within the scope of the thesis can be summarized as follows:

- The first systematic mapping study in the literature that investigates process properties for early phase defect prediction was presented [31]. Studies using process-based metrics for reliability and defect prediction in the early phases of the SDLC are discussed. Thus, the current picture of the literature is systematically summarized, emphasizing the distinctive features of process knowledge in the field of ESDP.

- Studies included information about the early phases of the SDLC, such as requirements and design, into the defect prediction model were systematically investigated. The performance changes in the studies that structured the prediction model by using the early phase information with the coding phase information were examined and thus, a unique contribution was made to the literature [32].
- A total of 52 scientific publications published between 2000 and 2016 was examined in depth by systematic mapping and literature review method and analyzed over a total of 16 research questions [15]. The trend and demographic information of the primary studies, the maturity of the research situation, the characteristics of the structuring of the prediction models, the methods used, the metrics, datasets, the performances of the ESDP models and the benefits of using these models were reported.
- Multi-criteria decision analysis methods were used for the first time in the literature to determine the most appropriate early phase defect prediction method for the project context. For this purpose, the criteria to be considered in the selection and alternative SDP methods were determined according to the literature analysis, and a decision matrix that evaluates these methods and recommends the most appropriate one for the context was proposed [33].
- As a new contribution to the literature, a decision analysis approach has been proposed for the selection of an SDP method for early phases [34]. In order to enable software stakeholders to apply defect prediction from the beginning of the life cycle of the relevant software project, the proposed approach aims to systematically select the most appropriate defect prediction method in line with the needs of the stakeholders and the characteristics of the related software project data.
- A web application for the decision analysis has been developed using Angular, Java and Spring framework. The source code has been made available and shared on GitHub[1] to enable researchers or practitioners to perform the decision analysis using the determined criteria, weights, and the list of selected alternatives.

[1] https://github.com/rozakinci/phd_thesis_app

1.5. Overall Design of Thesis Study with Mappings to RQs and Chapters

In Figure 1.2, the overall design of the thesis study is demonstrated with the connection of the related RQs and consequent chapters.

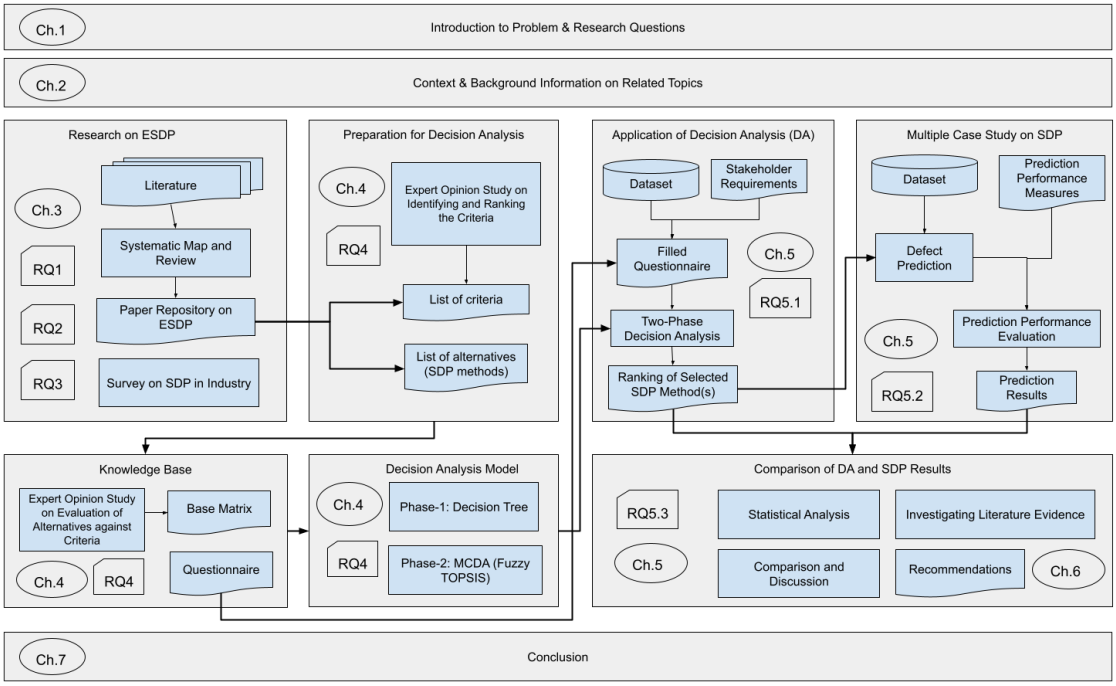


Figure 1.2. The design of the thesis with mapping to the RQs and chapters

1.6. Thesis Organization

Chapter 2 presents the background of this thesis by summarizing the general context of the software defect prediction. Chapter 3 examines the related work in the literature and determines the studies on the research area systematically and reports the analysis results within the scope of the research questions in detail. In addition, the survey conducted on the application of software defect prediction in companies in Turkey is also included in this chapter. Chapter 4 presents the necessary preparations for the selection of the early phase defect prediction method, how the emerging know-how as a result of the extensive work is systematically gathered and reported as a knowledge base, as well as the modeling of the decision analysis approach. In Chapter 5, the case studies that have been structured as an embedded multi-case design and experimental results related to the implementation and validation of the proposed decision analysis approach are described. Next, Chapter 6

summarizes a few critical points and recommendations that have emerged as a result of the thesis work. Finally, Chapter 7 summarizes the results obtained from the thesis and the contributions to the literature. In addition, the limitations of the thesis and plans for future studies are mentioned in this conclusion section.

2. BACKGROUND

2.1. What is “Defect”?

In the IEEE Standard Classification for Software Anomalies [35], a common dictionary has been created for the terms related to the errors that may be encountered throughout SDLC in the context of software engineering. According to the standard, the definition of defect is as follows:

- Imperfections or deficiencies that can be found in work products in the early phases of the SDLC, causing the work product to fail to meet requirements and needs to be fixed or replaced.

The definitions of other terms used in the same sense as the “defect” term are as follows:

- Error: Human action that can cause inaccurate results.
- Fault: Fundamental inaccuracies within the software program that can cause a malfunction.
- Failure: Deviation of program behavior from user expectations, failure to fulfill the expected function from the product under specified requirements and limits.
- Problem: Difficulty faced by the person while using a system, negative situation that needs to be solved.

Based on these definitions; malfunctions, disorders and anomalies that may be encountered in the early phases of the life cycle are discussed by using the term "defect" throughout the thesis [36].

2.2. Software Defect Prediction

Software defect prediction activities can be explained as using the models that are built via certain methods using different product, process, and/or resource-based metrics in order to prevent or minimize defects during software development life cycle. Its main purpose is to guide development, test and management teams to have an opinion on the software quality and therefore make decisions that provide to focus more deeply in

defective code, plan test activities in an effective way and make better use of resources [8].

2.2.1. Defect Prediction Approaches

In the field of data mining, the two most important types of prediction problems are defined as "classification" and "numerical prediction" [37]. Software defect prediction approaches are also divided into two as "classification as defective or non-defective" and "prediction of number of defects" based on the dependent variable. The most used defect prediction approaches can be grouped as follows according to their purpose of use:

- Classification[2]: Prediction of the category to which the data depends. The methods used for classification include: Expert judgement-based models (Fuzzy Inference Systems), Causal models (Bayesian Belief Network), Machine learning based models (Naïve Bayes, Artificial Neural Network, Decision Trees, Logistic Regression, Support Vector Machine).
- Numerical Prediction: Prediction of the number of defects. The methods used for numerical prediction include: Expert judgement-based models (Fuzzy Inference Systems), Machine learning based models (Artificial Neural Network, Decision Tree, Linear Regression, Support Vector Machine).

In addition, it is possible to categorize the SDP methods based on the approach to construct the model. In the context of ESDP, the most preferred approaches to construct the model can be said as machine learning (ML) based methods because of their ability to solve classification and prediction problems. Statistical methods are also preferred like ML based methods. In addition, it is possible to construct SDP models by considering the

[2] The term “classification” can be used to categorize a defect as belonging to certain classes, as in defect classification schemes, or to refer to a software defect prediction approach that involves classifying parts of software as defect-prone and defect-free. Throughout the thesis, the term "classification" is used for the defectiveness classification of a software part.

expert judgement-based approaches or causal methods. Therefore, we also present the below categorization of SDP methods:

- *Machine Learning based methods*: Supervised learning-based methods can be used in both classification and regression problems. Some implementations of this type include Artificial Neural Networks (ANN), Bayesian Networks (BN), Decision Trees (DT), Naïve Bayes (NB), and Support Vector Machines (SVM) [38].
- *Statistical methods*: These methods can also be preferred when applying SDP since they can be used in prediction models to be configured for both classification and regression [10]. Linear Regression (LinR) and Logistic Regression (LogR) methods can be categorized as statistical methods.
- *Expert judgement-based methods*: Fuzzy Inference System (FIS) based models can be constructed through a set of rules created according to expert judgment. The most important feature of the FIS methods is that they are independent from data and can handle imprecise data [39]. BN based models can also be built by expert judgement when there is sparse data and are known to be successful to address dependencies between attributes and handle uncertainty [21,40].

2.2.1.1. Statistical methods

Linear Regression (LinR)

It is one of the most known and best understood algorithms in statistics. When the class variable to be estimated and all attributes are numeric, the linear regression method is one of the simplest techniques to consider. A linear regression aims to find the line that best fits the relationship between the input variables (x) and the output variable (y). It can be defined as an equation ($y = C0 + C1 * x$) that detects and defines certain weights for input variables called coefficients (C) [41].

The purpose of the linear regression equation is to find the coefficient values when predicting the output (y) according to the input (x), namely C0 and C1. Some recommended good practices for linear regression are to exclude similar (related) variables from the dataset and, if possible, to remove noisy data. As a result, it is highly

preferred for numerical defect prediction in the field of SDP as it is a fast and simple technique.

Logistic Regression (LogR)

It is used to classify a categorical class variable based on the relationship between one or more numerical or categorical independent variables. It is similar to the linear regression method in that it aims to find the values of the coefficients that give weight to each input variable. Unlike LinR, a nonlinear function called logistic function is used to predict the output class. The logistic function has a structure similar to the letter “S” and converts any value into the range from 0 to 1 [37].

Thanks to the learning nature of the model, the predictions made by logistic regression can also determine the probability for the class to which the output belongs. This can provide a more meaningful result for the prediction problem. Logistic regression function, like LinR, performs better in the scenarios where attributes are related to the output and dependency between attributes does not exist. As a result, it is preferred for software defect classification problems since it is fast and effective.

2.2.1.2. Machine learning-based methods

Artificial Neural Network

The artificial neural network model is inspired by the human brain's ability to derive new information through learning. It consists of many small neuron-like elements called units and the directional and weighted relationships between these units. The layers are typically called the input layer, hidden layer, and output layer. There may be more than one hidden layer between the input and output layers. It is known to be more effective than other methods in modeling nonlinear functional relationships. It is generally used to predict the number of defects per class with object-oriented metrics [42]. However, artificial neural networks can be easily applied to very large datasets and can give results with higher accuracy than other methods [43]. They are suitable for problems where the number of feature-value pairs is high, the training set contains outliers / missing data, and the long training time is acceptable. The multiplicity of the number of connections, layers

and nodes determines the complexity of the system they can represent, the more nodes there are, the more complex (advanced) systems can be modeled. With these features, artificial neural networks solve problems that cannot be solved by classical algorithmic methods, similar to the system of the human brain [44].

Bayesian Classifiers

Bayesian classifiers are statistical classifiers based on Bayes theorem that aims to find the probability that a sample belongs to a class under given conditions. The most important feature is that they are incremental. That is, old knowledge can be used for observed data. Accordingly, the calculated probability increases or decreases incrementally [45]. Bayes rule states that "Based on the arguments we observe, what is the probability that the output belongs to class C?" and answers the question. Suppose Y is the class variable and X is the collection of independent classes. In this case, the formulation of the question "Given X, what is the probability that the result is of class C?" is given in Equation 2.1 [41]:

$$\Pr[Y = C \mid X] = \frac{\Pr[Y]\Pr[X|Y = C]}{Pr[X]} \qquad \text{(Eq 2.1)}$$

Naive Bayes

Naive Bayes, one of the Bayesian classifiers, has the advantage of handling various and independent features, missing values and noisy data. It also achieves results very quickly. The most obvious disadvantage of Naive Bayes is that it assumes that classes are conditionally independent. This assumption may cause a loss of accuracy [46].

Bayesian Networks (BNs)

Bayesian networks are represented by directed acyclic graphs, where each node defines a separate variable. Relations between these variables can be shown with Bayesian networks (such as the order of transition from one node to another). Bayesian networks generally consist of two parts [47]:

- Directed acyclic graph (DAG): The nodes in the graph can be defined as model variables and the connections between the nodes represent the causal effects among the variables.

- Conditional probability distributions (CPT): Unconditional probability distribution is applied for nodes with no ancestors. For nodes with ancestors, conditional probability distributions are made depending on the status of their ancestors.

Bayesian networks has many advantages. It has the ability to handle missing data, where each variable is assigned a preliminary probability, thus, if no input is provided for a variable, the default value of the probability is used in the computations. The BN models are generally easy to interpret, as the causal relationships between the variables are clearly visible in the graph. It can combine different types of data (e.g., quantitative and qualitative) where they can be used as inputs in model designs. Inputs and outputs do not have to be defined statically; a variable is an input if the user can observe it; if no observation can be made about the variable, it becomes an output.

Decision Trees

The structure of a decision tree is simple. The starting node in the tree is the root node. Each internal node represents the decision point that contains questions or criteria to be answered. The branches that connect nodes reflect the flow from question to answer. Lastly, leaf nodes give a result or result-set, which applies to all nodes that reach the leaf [38]. Decision tree algorithms have many implementations. The most common ones are ID3, C4.5, CART (Classification and Regression Trees). Classification trees are suitable for classifying the defectiveness of software components. Regression trees, on the other hand, can predict the number of defects [48]. Decision trees can use multidimensional data. The learning and classification process of the decision trees is often fast. Besides, they yield high performance prediction results generally. However, their performance can be affected from the nature of the data [37,38,41].

Support Vector Machines

It uses a non-linear mapping to convert the original training data to a higher dimension. Within this new dimension, the linear searches for the best parsing hyperplane (i.e., a "decision boundary") separates the threads of one class from another. With a suitable

nonlinear mapping in a sufficiently high dimension, data from the two classes can always be separated by a hyperplane, which can be found with the help of support vectors and margins [38]. SVM can be applied on both linear and nonlinear data. The learning phase can be slow; however, it has a high accuracy rate generally thanks to its ability to model complicated and nonlinear decision boundaries. They are prone to over-learning compared to other methods.

Genetic Algorithms

Genetic algorithms produce a set of solutions instead of producing a single solution to problems. Many points are evaluated at the same time in the search space, and as a result, the probability of reaching a holistic solution increase. It has been stated that it is suitable for use in scenarios where assumptions are excluded and the model focuses only on defect data [49]. The reasons for this are that genetic algorithms do not make any assumptions about data distribution, are not a parametric method, and do not form the model in a specific structure [49].

Ensemble Learning

It is a machine learning approach that is generally used for improving the prediction accuracy of classifiers. More than one classifier is trained to solve the same problem and these classifiers are combined to obtain stronger generalization ability [37]. As it will be explained in the following sections, ensemble learning methods are not included within the scope of the thesis, since it is desired to compare machine learning methods with their simplest forms.

2.2.1.3. Expert judgement-based methods

Fuzzy Inference Systems (FIS)

The fuzzy classification technique describes the dataset with approximate (partial membership) values without having precise and defined boundaries. For a software segment to be classified as defective, it must be defined with a membership value between 0 and 1. Using the data classified by the model based on fuzzy inferences, the “module-ordering model” predicts whether that module is defect-prone [50,51]. The most

important advantages of fuzzy logic-based methods can be listed as follows [52]. Data independence is the most important advantage of the FIS method. FIS models perform the modeling of the desired environment with the help of experts on research field, not by learning from data. Since it does not need historical data, it can be used from the beginning of the software project, providing faster results and usage repeatedly for the same research field. FIS models are said to be more suitable for defect prediction than data-driven methods. Models created can also be used for other software projects regardless of the domain, as they are data independent. Verbal, qualitative and non-numerical data are also well suited to use in fuzzy inference models.

The steps to be followed while building fuzzy models can be listed as follows:

1. Determination of membership functions of inputs and outputs

   Membership Functions (5 Scales) for linear scale:

   - VL (0; 0; 0.25),
   - L (0; 0.25; 0.50),
   - M (0.25; 0.50; 0.75),
   - H (0.50; 0.75; 1.00),
   - VH (0.75; 1.00; 1.00)

   Membership Functions for logarithmic scale (3 Scales):

   - L (0; 0; 0.37),
   - M (0; 0.37; 1),
   - H (0.37; 1; 1)

2. Determination of fuzzy logic rules: Various rules are determined by the field expert according to fuzzy sets and verbal variables. For a successful model design, all verbal variables in the fuzzy rule set and combinations of all verbal values of these verbal variables should be included. The number of rules is calculated by multiplying the number of verbal values of each verbal variable with each other. For example, the

number of rules required for an FIS consisting of 3 verbal variables and where each variable has 4 verbal values is 4 * 4 * 4 = 64.

3. Fuzzy inference: The fuzzy inference process can be explained as follows, in order:
    - Fuzzification of the determined inputs using membership functions
    - Performing the execution of fuzzy logic rules
    - Generating the fuzzy outputs of rules

4. Defuzzification step: After producing the fuzzy outputs, the defuzzification step is applied, where the fuzzy output is converted to crisp output. Although the fuzzy output helps to interpret the crisp values given as input, it does not tell the final decision, so the fuzzy output needs to be converted to crisp output. This conversion is called defuzzification. There are several types of models that vary in the technique they use for the crisp output generation step. The most used types are Mamdani, Sugeno and Tsukamoto.

2.2.2. Software Metrics

Software metrics enable us to understand and evaluate many aspects of software, thus to plan and track critical aspects throughout the project life cycle. The healthier we can perform the software measurement process, the more accurately we can control the software quality.

- Measurement: It is the process of assigning a value to an attribute. It can be a figure, size or quantity obtained as a result of the measurement process [53]. Measurement is also defined as the process of assigning numbers or symbols to the properties of real-world entities, according to strictly defined rules [54].
- Metric: Indicates the level at which a product, system, component or process possesses a certain attribute [50].

According to Fenton and Bieman [54], it is important to define the entities and attributes of the measurements as the first rule of thumb for performing software measurement activity. Based on Fenton and Bieman's classification, entities within the scope of software measurement activities are divided into three:

- Process: Refers to activities related to the software.
- Product: Outputs or documents obtained from a process activity.
- Resource: Refers to the entities required to perform the process activities.

Product metrics allow to measure structural and physical properties such as size (source code, requirement specification document size, size of design documents, etc.), complexity, length, dependency, and interactivity. The metrics defined in the Chidamber & Kemerer metric set [55] are the most widely used design and coding phase metrics for SDP in object-oriented software [56]. Process metrics measure the efficiency and effectiveness of software development processes, the duration of process activities, the effort spent, and the number of errors seen throughout the process. Since defects can be encountered from the earliest stages of software development processes, process metrics will be useful in SDP [57]. Resource metrics enable to measure the characteristics of the personnel (developer, designer, test staff, etc.) working in software development projects, such as experience, motivation, the characteristics of resources such as software and hardware needed in the project, and the structure of the working environment [54].

For each metric class (process, product, resource) it is divided into internal and external characteristics:

- Internal properties: can be measured by the product, process or resource itself.
- External properties: can be measured by how the product, process or resource relates to its environment, i.e., taking into account its behavior.

2.2.3. Public Datasets

2.2.3.1. PROMISE Repository – NASA Dataset

PROMISE data repository contains open datasets published to support the creation of prediction and/or decision support models in the field of software engineering on various topics (defect prediction, cost estimation, effort estimation, subsequent release monitoring etc.). It is aimed that the relevant prediction models can be applied by different researchers in the field or experts in the industry. The most used dataset in the software defect prediction field in this data repository has been published under MDP (Metric Data Program), a metric program created by NASA. In this context, there is data on 12 projects published. The PROMISE repository is currently not accessible [58], but a backup for the data is available fortunately and stored in GitHub [59]. The most used ones are given in Table 2.1.

Table 2.1. The characteristics of the projects from public NASA dataset

Project Name	Programming Language	Total Sample Number	Samples Marked as Defective	Defectiveness Rate (%)	Number of Attributes	Dataset Size
CM1	C	327	42	12.8	38	Small
JM1	C	7,720	1,612	20.9	22	Large
KC1	C++	1,162	294	25.3	22	Large
KC3	Java	194	36	18.6	40	Small
MC1	C++	1,952	36	1.8	39	Large
MC2	C	124	44	35.5	40	Small
MW1	C	250	25	10.0	38	Small
PC1	C	679	55	8.1	38	Medium
PC2	C	722	16	2.2	37	Medium
PC3	C	1,053	130	12.3	38	Large
PC4	C	1,270	176	13.9	38	Large
PC5	C++	1,694	458	27.0	39	Large

2.2.3.2. NASA-93 Dataset

It is an open dataset containing data from 93 projects prepared by NASA for use in the COCOMO model in the 90s, and later defect number data was added [60]. The attributes were demonstrated in Table 2.2, with their related software entity categorization. There are a total of 25 attributes in the version with defect data, which consists of:

- 15 standard COCOMO-I discrete attributes in the range from "Very Low" to "Extra High"
- 7 attributes describe the features of the project
- one of them describes the number of lines of code
- one of them is the actual effort in person months
- the dependent attribute is the number of defects

Further detailed descriptions can be found in the COCOMO II model manual [61].

2.2.3.3.Fenton Dataset

Fenton et al. proposed a causal defect prediction model using several quantitative and qualitative process factors [20,21]. The design of the model and the specified qualitative factors were first described in [20]. After that, they extended this work to describe the prediction model in more detail and validate it [21]. The most critical output of this study is the open dataset they provide to the literature[3]. Their main motivation for presenting their raw data is the possibility of enabling different SDP methods to be implemented by other researchers, and that the results are useful for software project managers to use practically.

[3] Throughout the thesis, the Fenton dataset is referred from their extended work [21].

Table 2.2. Attributes of NASA-93 dataset

Entity	**Attribute**	**Abbreviation**	**Type**
Product	Precedentedness	prec {h}	Nominal
Product	Development Flexibility	flex {h}	Nominal
Process	Architecture and Risk Resolution	resl {h}	Nominal
Resource	Team Cohesion	team {vh}	Nominal
Process	Process Maturity	pmat {l,n,h}	Nominal
Product	Required software reliability	rely {l,n,h,vh}	Nominal
Product	Database size	data {l,n,h,vh}	Nominal
Product	Product Complexity	cplx {l,n,h,vh,xh}	Nominal
Product	Developed for Reusability	ruse {n}	Nominal
Product	Documentation match to life-cycle needs	docu {n}	Nominal
Product	Execution Time Constraint	time {n,h,vh,xh}	Nominal
Product	Main Storage Constraint	stor {n,h,vh,xh}	Nominal
Product	Platform Volatility	pvol {l,n,h}	Nominal
Resource	Analysts capability	acap {n,h,vh}	Nominal
Resource	Programmers capability	pcap {n,h,vh}	Nominal
Resource	Personnel continuity	pcon {n}	Nominal
Resource	Application experience	apex {l,n,h,vh}	Nominal
Resource	Platform experience	plex {vl,l,n,h}	Nominal
Resource	Language and Tool Experience	ltex {vl,l,n,h}	Nominal
Resource	Use of Software Tools	tool {n,h}	Nominal
Resource	Multisite development	site {n}	Nominal
Resource	Required Development Schedule	sced {n,l,h}	Nominal
Product	Equivalent physical 1000 lines of source code	kloc	Numeric
Process	Development effort in months	effort	Numeric
Process	Number of defects	defects	Numeric

The dataset contains data on 31 software projects developed in the consumer electronics industry. The scope of the projects is the development of embedded software in consumer electronics products, and it is aimed to develop several functions provided by a product in each project. The developed software are not independent systems, and they are developed as subsystems of other software in the electronic product. Waterfall approach is followed as the SDLC. In the software engineering part of the life cycle, requirements documentation review, design, design review, coding and unit testing activities are carried

out. Later, the software is put into independent testing at many stages, from integration testing to system testing. Requirements analysis and independent testing processes are usually performed in a different location than the coding.

Data are collected through questionnaires conducted with project managers, quality managers and/or expert project personnel of the relevant projects. Qualitative data from surveys have 5 scales and can take the following values in order: Very High, High, Medium, Low, Very Low. There are areas such as explanations and detailed sub-questions regarding the questions. For example, if there are 10 sub-questions for a question, if all sub-questions are answered yes, the score of the question will be VH, if 7-9 of them are yes, the score will be H, and so on. For example, for gathering the answers on factor "S1 - Relevant Experience of Spec and Doc Staff", the main question and additional questions were defined as follows:

Question: How would you evaluate the experience and skill level of your team members who took part in the requirement phase of this project?

- Sub-question1: Did the requirements team have sufficient experience?
- Sub-question2: Did the requirements team have sufficient domain expertise?

Sample Answers:

- Very High: Software engineers with more than 3 years of requirements management experience and extensive domain knowledge.
- High: Software engineers with more than 3 years of requirements management experience but limited domain knowledge.
- Intermediate: Software engineers with 1 to 3 years of experience in requirements management.
- Low: Software engineers with 1 to 3 years of experience but no experience in requirements management.
- Very Low: Software engineers with less than 1 year of experience and no previous field experience.

The identified factor groups and related factor names were demonstrated in Figure 2.1.

Factor group	**Factor ID and Name**
Specification and documentation process	S1 Relevant Experience of Spec and Doc Staff
	S2 Quality of Documentation Inspected
	S3 Regularity of Spec and Doc Reviews
	S4 Standard Procedures Followed
	S5 Review process effectiveness
	S6 Spec Defects Discovered in Review
	S7 Requirements Stability
New functionality	F1 Complexity of New Functionality
	F2 Scale of New Functionality Implemented
	F3 Total Number of Inputs and Outputs
Design and development process	D1 Relevant Development Staff Experience
	D2 Programmer Capability
	D3 Defined Processes Followed
	D4 Development Staff Motivation
Testing and rework Factor	T1 Testing Process Well Defined
	T2 Testing Staff Experience - unit
	T3 Testing Staff Experience - integrated
	T4 Quality of Documented Test Cases
Project management	P1 Development Staff Training Quality
	P2 Requirements Management
	P3 Project Planning
	P4 Scale of Distributed Communication
	P5 Stakeholder Involvement
	P6 Customer Involvement
	P7 Vendor Management
	P8 Internal Communication/Interaction
	P9 Process Maturity
Quantitative Data	E Total Effort
	K KLOC
	L Language
	TD Total Defects

Figure 2.1. Factors in Fenton Dataset [21]

2.2.4. Performance Evaluation Measures

In order to choose an approach for the performance evaluation of defect prediction models, first of all, the type of the predicted dependent variable should be considered. In this context, it is possible to divide the models into two [8]:

- Categorical Models: use categorical variables (defective or non-defective) as dependent variable. Models created with classification methods fall into this group.
- Continuous Models: use numerical variables (number of defects) as dependent variables. Models created with numerical prediction methods fall into this group.

2.2.4.1. Measures Used in Performance Evaluation of Categorical Models

In categorical models, the evaluation of the prediction performance of the model is basically made by confusion matrix analysis given in Figure 2.2 [62]. This matrix uses various calculations where the model considers actual class labels to measure how it classifies different categories. In other words, the class label predicted by the model is compared with the class label to which the dependent variable actually belongs.

- True Positive (TP): The class label ("defective") was predicted correctly.
- False Positive (FP): The class label ("non-defective") was guessed incorrectly ("defective"). Also known as Type I Error.
- False Negative (FN): The class label ("defective") was guessed incorrectly ("non-defective"). Also known as Type II Error.
- True Negative (TN): The class label ("non-defective") was predicted correctly.

		Predicted Class	
		Positive (P)	**Negative (N)**
Actual Class	**Positive (P)**	TP	FN
	Negative (N)	FP	TN

Figure 2.2. The confusion matrix

Based on this matrix, many performance evaluation measures can be derived [63] as listed below. The sysnonims and formulations of these measures are presented in Figure 2.3.

Measure	Synonyms	Formulation
True positive rate (TPR)	Recall Probability of detection / pd Sensitivity	$\frac{TP}{TP + FN}$
False positive rate (FPR)	Probability of false alarm / pf Type-I Error	$\frac{FP}{FP + TN}$
True negative rate (TNR)	Specificity	$\frac{TN}{TN + FP}$
False Negative rate (FNR)	Type-II Error	$\frac{FN}{TP + FN}$
Precision		$\frac{TP}{TP + FP}$
f-measure		$\frac{2 \times Recall \times Precision}{Recall + Precision}$
Accuracy		$\frac{TP + TN}{TP + TN + FP + FN}$
Misclassification rate	Error-rate	$1 - Accuracy$
Balance		$1 - \frac{\sqrt{(PF^2 + (1 - PD)^2}}{\sqrt{2}}$

Figure 2.3. Performance evaluation measures

- True positive rate (TPR): It is synonymous with Recall, probability of detection (pd) and Sensitivity. It refers to the rate at which the class that is actually labeled as “defective” is predicted as “defective” in the prediction result.
- False positive rate (FPR): It is synonymous with probability of false alarm (pf) and Type-I Error. It refers to the rate at which the class labeled as “defect-free” is predicted as “defective” in the prediction result.
- True negative rate (TNR): It has the same meaning as Specificity. It refers to the rate at which the class labeled as “defect-free” is also predicted as “defect-free” in the prediction result.

- False negative rate (FNR): It has the same meaning as Type-II Error. It refers to the rate at which the class that is actually labeled as “defective” is predicted as “defect-free” as a result of the prediction.
- Precision: refers to the rate at which “defective” predictions are made correctly.
- f-measure: It is expressed as the harmonic mean of the precision and recall values.
- Accuracy: The ratio of correctly classified units.
- Misclassification rate: It has the same meaning as Error-rate. It is the proportion of incorrectly classified units.
- Balance: It expresses the distance to the most perfect point, defined as PD=1 and PF=0, in terms of PD and PF calculated as a result of the estimation.

ROC Curve and AUC Value

ROC Curve (Receiver Operating Characteristic curve) is a method applied to interpret classification performance graphically. As shown in Figure 2.4, the ROC curve graph has two dimensions: PD (true positive rate) on the y-axis and PF (false positive rate) on the x-axis. The most successful classifiers have high PD and low PF.

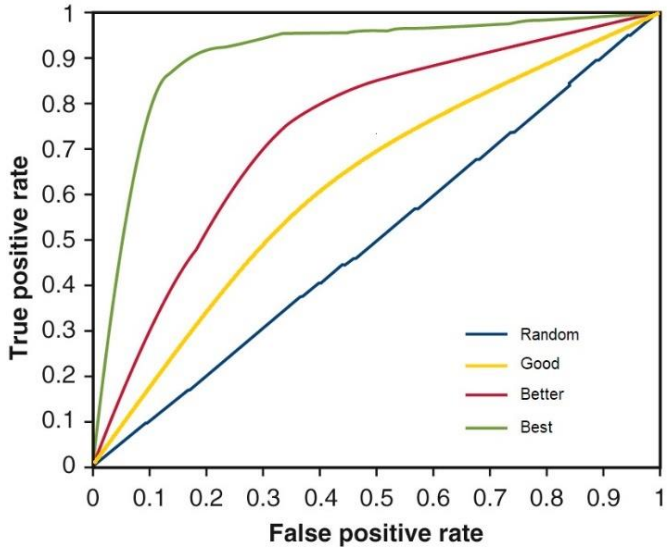


Figure 2.4. ROC curve

AUC (Area Under the Curve) refers to the area under the ROC curve. When PD is equal to PF, the area under the ROC line is an isosceles triangle with sides of length 1; thus the AUC value is 0.5. If the AUC value is calculated over 0.5 in the performance evaluation of a model, it can be said that the model gives acceptable prediction results, and the results get better as it gets closer to 1.

2.2.4.2. Measures Used in Performance Evaluation of Continuous Models

Co-efficient of determination (R^2): It is a statistical measure of goodness-of-fit, which measures how good the predicted regression equation is. It has the range of values between 0 and 1, where higher R^2 represents more confidence in the equation. Suppose we have existing values yi and predicted values y'i (for i = 1, 2, 3, ... , n; n ∈ N), where $\overline{y}$I is a mean value of y′i,

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(yi - y'i)^2}{\sum_{i=1}^{n}(yi - \bar{y}'i)^2} \quad \text{(Eq 2.2)}$$

Root mean square error (RMSE): Relative squared error takes the total squared error and normalizes it by dividing by the total squared error of the predictor. Then taking the square root of the relative squared error, reduced error being predicted is calculated.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(yi - y'i)^2} \quad \text{(Eq 2.3)}$$

Normalized root mean square error (NRMSE): It shows the ratio between RMSE and existing values. The NRMSE value can be used to compare single model performance.

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}} \quad \text{(Eq 2.4)}$$

Mean Magnitude of Relative Error (MMRE): It is the arithmetic mean of absolute relative error. The lower it is, the better the prediction.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|yi - y'i|}{yi} \quad \text{(Eq 2.5)}$$

Balanced Mean Magnitude of Relative Error (BMMRE): It is a balanced version of the MMRE that deals more with underestimation than overestimation.

$$\text{BMMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|yi - y'i|}{\min(yi, y'i)} \quad \text{(Eq 2.6)}$$

2.2.5. SDP During Early Phases

Most SDP models are generated using metrics from the coding and testing phases of the SDLC. However, when it comes to those phases, it may be too late to plan corrective and preventive actions effectively. As a solution to this problem, it can be appropriate to build and use SDP models in the early phases of the SDLC, which can be defined as requirement analysis or design phases, in terms of many activities such as quality estimation, effective resource, calendar and cost planning in the software life cycle [12,64].

In the earlier phases of SDLC, project teams do not have any metrics related to source code or testing, or reported defect data from the product environment that could be used to predict future defects of the software. Therefore, the data and metrics that can be used early in the SDLC can be summarized as follows:

- Sub-product data that can be collected from early-phase sub-products (such as requirement specification document and design documents).
- Process-based data that can be collected from early-stage processes (requirements analysis, design, early stages of coding).
- Resource-based data on the experience of the software development team and the availability of other resources.
- Qualitative data based on expert opinions that can be obtained in the early stages from the opinions of experts who can evaluate the software according to the software context parameters.
- Historical project data similar in context to the related software.

2.3. Decision Analysis

In complex situations that require in-depth knowledge of the subject to be decided, a decision analysis process should be performed using systematic methods among the alternatives. The definition of decision making is expressed as choosing the most appropriate one among the alternatives to be considered in terms of goals, objectives, values and criteria [65]. According to Fulop [66], a general decision-making process can be divided into the following steps:

1. Define the problem,
2. Determine requirements,
3. Establish goals,
4. Identify alternatives,
5. Define criteria,
6. Select a decision-making tool,
7. Evaluate alternatives against criteria,
8. Validate solutions against problem statement.

Especially for the decision-making problems involving high risk and uncertain scenarios, it is a possible approach to first use a decision tree to see the potential results, and then apply the multi criteria decision anaylsis (MCDA) on these potential results to reach the final result over the total preference score [67]. These two analysis methods used in the decision analysis approach within the scope of the thesis are summarized below.

2.3.1. Decision Tree

In decision-making systems, decision tree is one of the best-known techniques. They allow to make decisions through a “top-down, divide-and-conquer” approach to the problem by addressing a set of decisions available in the tree nodes.

In decision analysis context, there are a couple of advantages of decision trees [68]. A rule set emerges as a result of structuring decision trees, thus providing clarity and

conciseness for decision makers by making it easier to explain the decisions taken, which can be presented in an interpretable format. Not all decision attributes may be helpful in the same way for different decision-making contexts. For those types of problems, decision trees ensure that the suitability of different attributes depends on the results of the previous tests, thus they have a high context sensitivity. Besides, they can successfully handle both continuous and discrete attributes. They can be combined with other decision techniques. No domain knowledge is required for the construction of decision trees, so it is suitable for knowledge discovery.

2.3.2. MCDA

Multi-criteria decision analysis is a set of formal approaches to address complex decision problems in a scientific and analytical framework, aimed at assessing multiple criteria for a decision maker to reach the most appropriate solution [69]. There are different MCDA methods in the literature, each with its own characteristics and categorized in many different ways [70]. The best known methods can be listed as AHP (Analytic Hierarchical Process) [71], ELECTRE (Elimination and Choice Expressing the Reality) [72], TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) [73] and PROMETHEE (The Preference Ranking Organization METHod for Enrichment of Evaluations) [74].

Fuzzy set theory can be applied to address uncertainty issues that may arise in a few situations where the criteria are vague or decision makers are unsure how to evaluate the relevant criteria [75]. Fuzzy TOPSIS introduced by Chen and Hwang [76] by extending the TOPSIS method using linguistic variables represented by triangular fuzzy numbers. Later, studies that utilizes fuzzy logic theory with TOPSIS method continued in the literature [77–79]. The basic logic of the Fuzzy TOPSIS method is that the selected alternative should have the shortest distance to the Fuzzy Positive Ideal Solution (FPIS) that maximizes the benefit criteria and minimizes cost criteria, and the farthest distance to Fuzzy Negative Ideal Solution (FNIS) that maximizes the cost criteria and minimizes the benefit criteria [78,79]. The general steps of Fuzzy TOPSIS method can be summarized as follows [78,80]:

1. Determine the appropriate linguistic variables for ranking alternatives with respect to each criterion.
2. Assign weights to the criteria and ratings to the alternatives.
3. Calculate the aggregated weight of alternatives with respect to each criterion.
4. Compute the normalized fuzzy decision matrix.
5. Compute the weighted normalized fuzzy decision matrix.
6. Calculate the Fuzzy Positive Ideal Solution (FPIS) and Fuzzy Negative Ideal Solution (FNIS).
7. Determine the distance of each alternative from FPIS and FNIS.
8. Calculate the closeness coefficient (CCi) for each alternative.
9. Rank the alternatives.

The most important advantage of Fuzzy TOPSIS method is that when the decision makers evaluate the alternatives, they benefit from using a natural language to describe their subjective judgement in a quantitative manner [80].

3. RELATED WORK

3.1. Secondary Studies on SDP

Numerous software defect prediction papers have been published in the literature. Therefore, there are many literature review and analysis studies about these papers. These secondary studies have surveyed the literature according to several aspects of the defect prediction models, such as methods, metrics and performance evaluation methods. We have analyzed these studies in software defect prediction literature by grouping them based on their research method (Systematic Literature Review (SLR), Systematic Mapping (SM), and Literature Review). We should note that research methods of the secondary studies were classified based on the guidelines provided by Petersen et al. [22] and Kitchenham and Charters [23]. If any guidelines were not followed in secondary studies, we classified them as literature review.

3.1.1. Systematic Literature Review Studies

Çatal and Diri [10] reviewed software defect prediction papers by examining their types of metrics, methods and datasets. The results show that the usage of the public datasets and machine learning approaches increased significantly after 2005 when PROMISE repository was created.

Hall et al. [8] investigated the performance values of SDP models in their systematic review study in 2012, included 208 experimental studies published between 2000 and 2010, and examined a subset of 36 out of 208 studies. The main objective was to evaluate the context information, input variables and modeling techniques and their effects to the performance of the models. The main findings showed that models based on simple approaches such as Naïve Bayes or Logistic Regression performed well. Besides, the combination of different input variables, and usage of feature selection techniques resulted in better performance.

Radjenovic et al. [81] reviewed software metrics and their usability in SDP over 106 studies. They reported that object-oriented (OO) metrics were used nearly twice as often

compared to traditional source code or process metrics. They also stated that OO and process-based metrics are more successful than size and complexity metrics in predicting defects.

Malhotra [82] analyzed the performance of the machine learning techniques for SDP models through 64 studies in 2015, and summarized the characteristics based on metrics reduction techniques, metrics, datasets and performance measures. It was concluded that the machine learning techniques had acceptable defect prediction capability and could be used by software practitioners and researchers.

Wahono [83], identified and analyzed the research trends, datasets, methods and frameworks used in SDP studies published between 2000 and 2013. The results showed that about 77% of the studies were focused on classification methods, and 65% of the research studies used public datasets.

3.1.2. Systematic Mapping Studies

Murillo-Morera et al. [84] investigated the software metrics, prediction techniques based on data mining or machine learning and their performance over 70 studies. They found the frequently used combination of metrics and methods as follows: Halstead, McCabe and LOC metrics with Random Forest, Naive Bayes, Logistic Regression and Decision Tree methods.

Özakinci and Tarhan [31], presented initial results from a systematic mapping of 41 early software defect prediction studies published between 2000 and 2015, and reviewed 18 papers in detail and in a narrower scope, to elicit the process attributes and metrics used in the models. It was observed that 44% of the early defect prediction studies build the prediction model by using process-based data, such as effort of the review activities, or requirement stability metrics.

Özakinci and Tarhan [15], systematically mapped and reviewed 52 primary studies published between 2000 and 2016. They provided a general view about the characteristics, performances, and usefulness of ESDP models by elaborating on the prediction methods, software metrics, performance evaluation approaches used in the studies, as well as the reported benefits of using ESDP models. This study differs from the existing works in that it is the first study that focuses on the literature about early software defect prediction in a systematic and comprehensive manner.

3.1.3. Other Literature Reviews

Catal [85] investigated 90 software defect prediction papers published between 1990 and 2009. This review provided a guide for researchers to investigate the studies on software metrics, methods, datasets, and performance evaluation metrics.

Jureczko and Madeyski [86] presented a review and investigated process-based metrics in SDP. They focused on the most important results, recent advances and summary regarding the use of these metrics in prediction models. They reported that employing process metrics in the defect prediction could lead to better results than working only with the product metrics.

Singh et al. [87] investigated various prediction methods used in the area over 20 studies. According to the results, researchers have mainly used multivariate regression analysis, genetic algorithms, neural networks, Bayesian network techniques for SDP. It is stated that NASA datasets are the most common data source and widely used in the area.

3.2. Studies Focus on SDP Frameworks

Several studies that propose different frameworks in the field of SDP research are discussed below.

Wahyudin et al. [88] proposed an SDP framework to provide guidance on how defect prediction should be organized in a particular project and organizational context. The

framework includes a three-stage defect prediction model. First, the requirements are defined to align the expectations of the software stakeholders with what can be achieved in practice. Second, the model is constructed based on the identified variables and the selected defect prediction method. In the final stage, the prediction model is applied to the actual software project data and the accuracy of the model is tested. An initial empirical evaluation of the framework was conducted based on the findings of the 12 studies in the literature, although no experiments were conducted for the implementation of the framework.

Song et al. [14] proposed a framework that includes schema evaluation and defect prediction components. The first component examines prediction performances by applying learning schemes on historical datasets, and the second component constructs a prediction model that uses the high-performance schema and applies it to the actual dataset. The performances of the experiments performed on the simulation data and NASA dataset were compared according to the AUC values, the framework was reported to be efficient but different schemes may be required for different data types.

Meta-learning is also used in the literature for algorithm selection and recommendation as an alternative approach, which aims to learn the behavior of the classifiers and determines the dataset features that contribute to better performance. According to the results of the experiments performed on the PROMISE datasets for the “meta-learning framework” [89], it was reported that algorithms with better defect prediction performance were recommended successfully. The findings of this study are important for the literature, as its authors reported that researchers should focus on improving algorithm recommendation rather than trying to build more robust SDP models for different contexts. In addition, Porto et al. [90] proposed a meta-learning approach to automatically select and recommend the most suitable cross project defect prediction method. They evaluated their meta-learning solution on 15 open-source software projects. According to the results, the proposed solution can learn from previous experiences and recommend suitable methods dynamically, however, there was a minor loss in the prediction performance compared to the base methods.

Another approach that has attracted a lot of attention in recent years is the transfer learning method [91]. When the target domain has a limited amount of data, transfer learning uses the source domain information for model learning in the target domain. Therefore, it is considered a useful approach for cross-company software defect prediction, and in cases when different distributions of the training and testing datasets exist [92,93].

Rathore and Kumar [94] presented a recommendation system that facilitates the selection of the appropriate technique(s) to build an SDP model, addressing the various characteristics of the defect data as well as the appropriateness of both machine learning-based and statistical techniques. In this context, they made a review of the literature to reveal the features that should be evaluated, after that, they created various decision rules according to the evaluation of these features and presented a decision tree-based recommendation system. The system was evaluated with several case studies, and it is reported that it provided useful hints in choosing SDP techniques.

3.3. SDP Studies Using MCDA

In the field of SDP, there are a couple of MCDA studies in the literature. Balogun et al. [95] evaluated the performance of various machine learning approaches by using Analytic Network Process (ANP). Peng et al. [96] focused on comparing the performance of several ensemble methods through the application of (Analytic Hierarchy Process) AHP, where Wu [97] presented an Analytic Hierarchy Model (AHM) to select the best algorithm for high-efficiency clustering in SDP. In addition, Kou et al. [98] applied feature selection and classifier evaluation in the context of SDP by using different MCDA methods such as ELECTRE, PROMETHEE and TOPSIS.

All of the studies above focus on the comparison of various machine learning based classification methods with performance measurements using data from NASA Metrics Data Program (MDP) published in PROMISE repository. Overall, these studies report a positive effect of applying MCDA methodologies in assessing the predictive performance of different classifiers. In addition, it is important to know that the experimental results using different performance measures over different project data on NASA MDP may be

different from each other. Therefore, these studies are very valuable to evaluate the performance of different classification methods to be used in other software projects with context information similar to a project in NASA MDP.

3.4. Defect Prediction in Early Phases – State of the Art and Benefits of ESDP

A systematic mapping and systematic literature review study [15] was conducted as a basis for this thesis. To ensure transparency, we have published the entire repository of the primary studies and results of the study online at [99]. We identified the primary studies with the prefix 'S' as an abbreviation for the ‘Source’ term. The mapping table for the source IDs of the primary studies and the corresponding reference is given in Appendix-1.

While constructing the review process, the guidelines and protocols proposed by both Petersen et al. [22] and Kitchenham et al. [23] were followed. It is important to note that Petersen [100] and Idri et al. [101] also adopted the same methodology for conducting systematic mapping and review study. The protocol of our systematic study is shown in Figure 3.1.

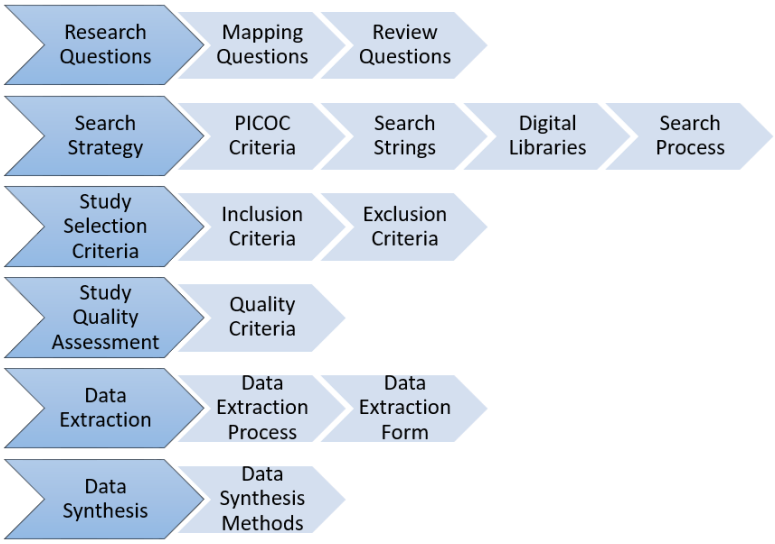


Figure 3.1. Research protocol for systematic mapping and literature review

The objective for this study was to obtain a general view of the characteristics and usefulness of ESDP models reported in scientific literature. The authors searched for the

studies reported between 2000 and 2016. A total of 52 studies were reviewed and analyzed with regard to the trend and demographics, maturity of state-of-research, in-depth characteristics (datasets used, SDLC phases, software metrics, prediction methods, contextual information), prediction performance evaluation and benefits of ESDP models. A more detailed classification scheme of the SLR is given in Table 3.1.

Table 3.1. Classification scheme

Research Question	**Property**	**Possible Values**	**(M)ultiple/ (S)ingle**
RQ1.1	Dataset Type	Public, Private	M
RQ1.2	SDLC Phase	Requirement, Design, Coding, Testing	M
RQ1.3	Software Entity	Product, Process, Resource	M
RQ1.4	Attributes Associated with Product Entity	Size, Structure	M
	Attributes Associated with Process Entity	Effort, Stability, Process Maturity, Number of Defects, Adequacy, Time	M
	Attributes Associated with Resource Entity	Project, Human	M
RQ1.5	Software Metrics	Full list is given in Table 3.3.	M
RQ1.6	Prediction Method	Bayesian Network, Fuzzy Logic, Machine Learning, Statistical	M
RQ1.7	Contextual Parameters	Commercial, Criticality, Development Methodology, Domain, Programming Language, Quality Expectancy, Size, System Type	M
RQ2.1	Performance Evaluation Methods	Categorical, Continuous	S
	Performance Evaluation Measures	ROC, AUC, PD (Recall), PF, Precision, Accuracy, F-measure, error measures, goodness-of-fit, ranking results, accuracy, difference between expected and observed	M
RQ2.2	Prediction Performance Values	Performance values based on mostly reported measures such as AUC or MMRE	M
RQ2.3	Benefits	Full list is given in Table 3.5.	M

As seen in Table 3.1, the first column represents the research questions that are relevant to each property in the classification scheme listed in the second column. The set of all possible values for each property is given in the third column. The fourth column indicates if a property can have multiple values. For example, a study may have used more than one prediction method; therefore, multiple possible values regarding prediction method category will be marked in this case. The explanation for each property and related possible values are given below.

- "Dataset type" refers to the access the data used in the study is whether public or private. Neither dataset, defect data nor source code is available for "private" datasets. It is therefore not definite if the study is reproducible. It is worth to note that if the study did not mention the availability of dataset, it was categorized as private. On the other hand, in "Public" datasets, the metrics and the defect data are publicly available (e.g., PROMISE Data Repository), therefore, the study using public datasets is considered reproducible.

- "SDLC phase" states the software development life cycle stage that originates the metrics for the prediction model. In other words, this property explains the phase in which the inputs needed for the prediction model are obtained. The phases were categorized as Requirement, Design, Coding, and Testing. Together with the phase information, it would be beneficial to report the software development method used in the studies; however, only a few papers [S1, S5, S13, S37] clearly expressed the development method used.

- According to [54], as the first rule for performing software measurement activity, it is crucial to identify the entities and attributes of the measure. Therefore, based on definitions of Fenton and Bieman's classifying software measures [54] and measurable product and process attributes of Florak et al. [102], we include "Software Entity" and "Attributes Associated with Each Entity" to describe the type of the entities and their related attributes, respectively. Some of these attributes are highly relevant with software metrics used as inputs to the ESDP models. During the review of the papers included in this systematic review, those attributes and metrics were progressively added to the classification scheme.

- "Prediction Method" expresses the specific method used in the study regarding the building of the prediction model. Examples of prediction methods include

machine learning, fuzzy logic based, Bayesian Network based, statistical based etc.

- "Contextual Parameters" are required to obtain more detail about the datasets used in prediction studies. We adopted some of the contextual characteristics from [19] and [103]. Examples of contextual parameters include domain, programming language, and size of the software, development methodology used in the project life cycle etc.
- "Performance Evaluation Methods and Measures" are necessary for assessing the success of the prediction model. According to the classification of Hall et al. [8], defect prediction studies may report their results via categorical or continuous dependent variables.
- "Benefits" were categorized with regard to the mostly reported qualitative benefits in the primary studies. They were gathered through the iterative cycles of the full-text reading and categorized with regard to similar phrases which primary studies reported as a benefit or advantage.

3.4.1. RQ 1: What are the characteristics of ESDP models?

3.4.1.1. RQ1.1 Which types of datasets are used for performing the prediction?

The distribution of the dataset types given in Figure 3.2. Public datasets (50%) were preferred since they are open to access. Public datasets includes: 1) NASA Metrics Data Program (MDP) which is located in PROMISE repository [58], 2) qualitative and quantitative data about 31 projects that were published in [20], and 3) raw data published in [S16]. Private datasets were also used (with 48%) in ESDP studies, which belonged to industrial companies or individuals. One study did not use any type of dataset as it is not a case study, it only proposes the defect prediction model [S17]. Moreover, in order to see the change of interest to public or private dataset types, the cumulative distribution over years is presented in Figure 3.3. It was obtained from the number of dataset types used in the studies by summing them over the years.

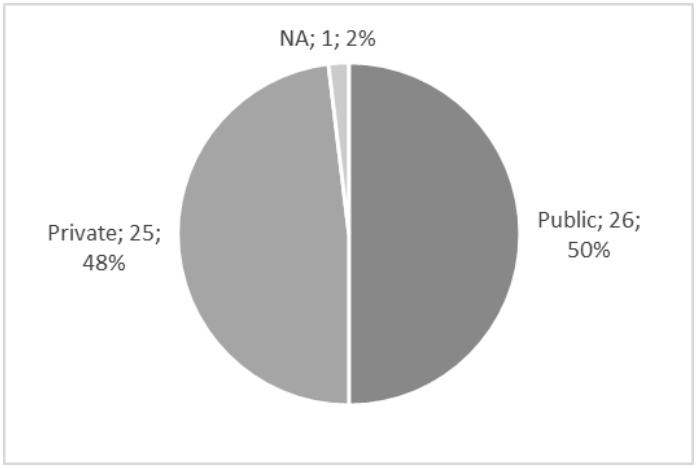


Figure 3.2. Distribution of dataset types

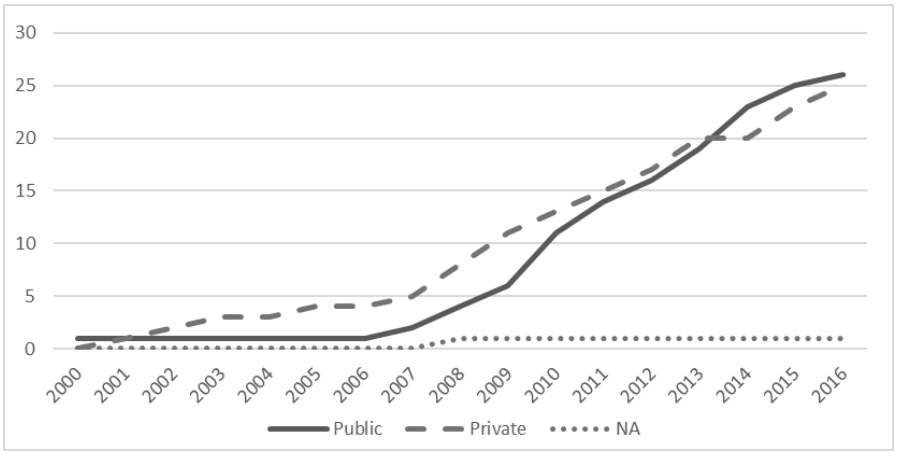


Figure 3.3. Cumulative number of dataset types per year

3.4.1.2. RQ1.2 What are the development phases that originate metrics for the prediction models?

The individual numbers of SDLC phases included in prediction models are provided in Figure 3.4. While three studies used only requirement phase-based data, eleven studies preferred only design phase-based data. Six studies focused on requirement, design and coding phase-based data together; and, six studies included only design and coding phase-based data for early defect prediction.

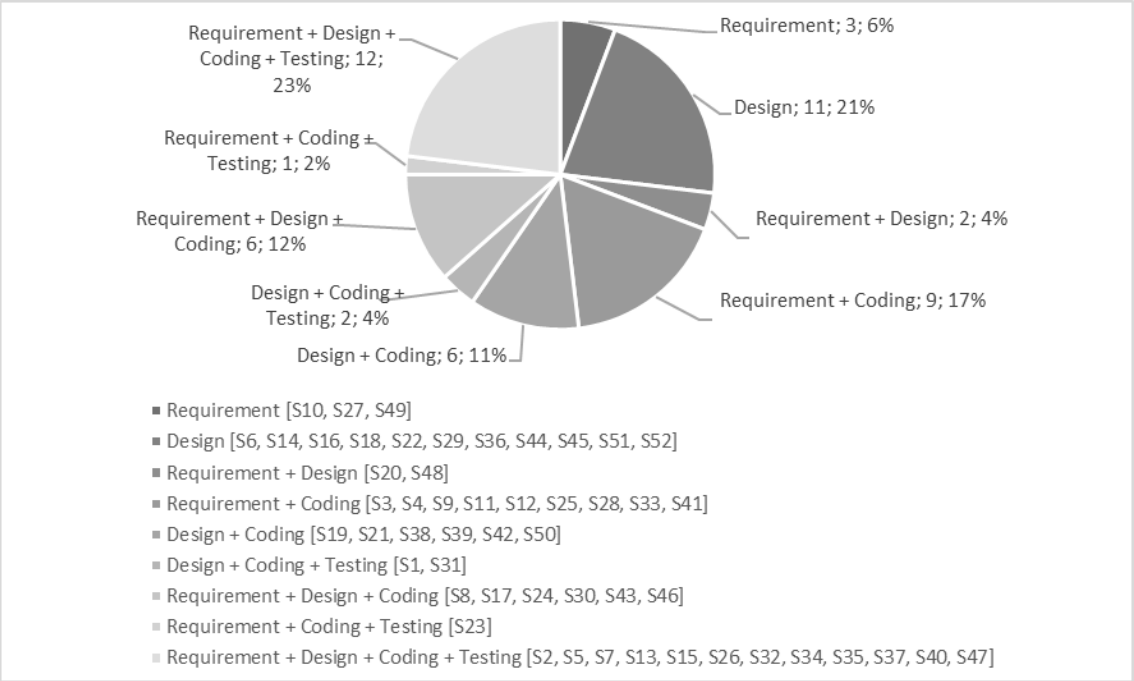


Figure 3.4. Individual distribution of SDLC phases

The cumulative percentages of the SDLC phases associated with early prediction studies can be seen in Figure 3.5. Overall, 33 studies covered requirement phase-based data for the early prediction. Besides, 39 studies included design phase-based data in the prediction methods. Design phase-based data was mostly preferred (32%) while constructing early prediction models. In addition, it is important to note that there is a high adoption of requirement phase-based data (27%) in order to provide earlier prediction results. Since studies that used requirement and design phase-based data mostly covered coding phase-based data too; its percentage was about 29%.

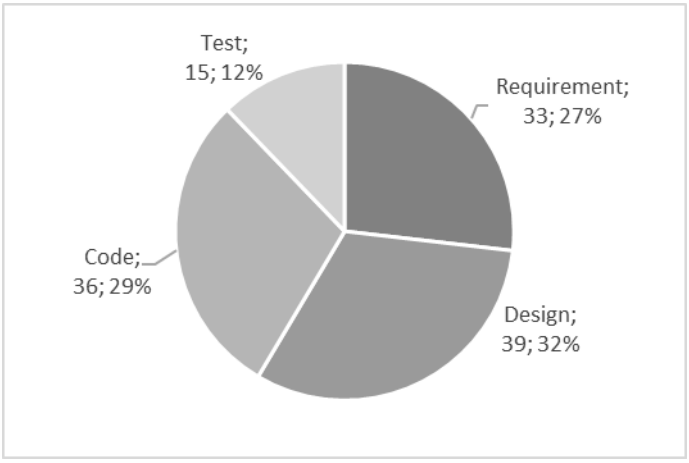


Figure 3.5. Cumulative distribution of the SDLC phases

3.4.1.3. RQ1.3 What are the entities that originate metrics for the prediction models?

The software entities subject to prediction studies were elicited from the software metrics used in the studies. Twenty-seven studies used only product entity-based data, and three studies used metrics of process entity. Six studies used both product and process entity-based data to gather metrics, where only two studies used metrics from process and resource entities together. Fourteen studies used metrics that were related to all entities. The individual distribution of the entities among all studies is shown in Figure 3.6.

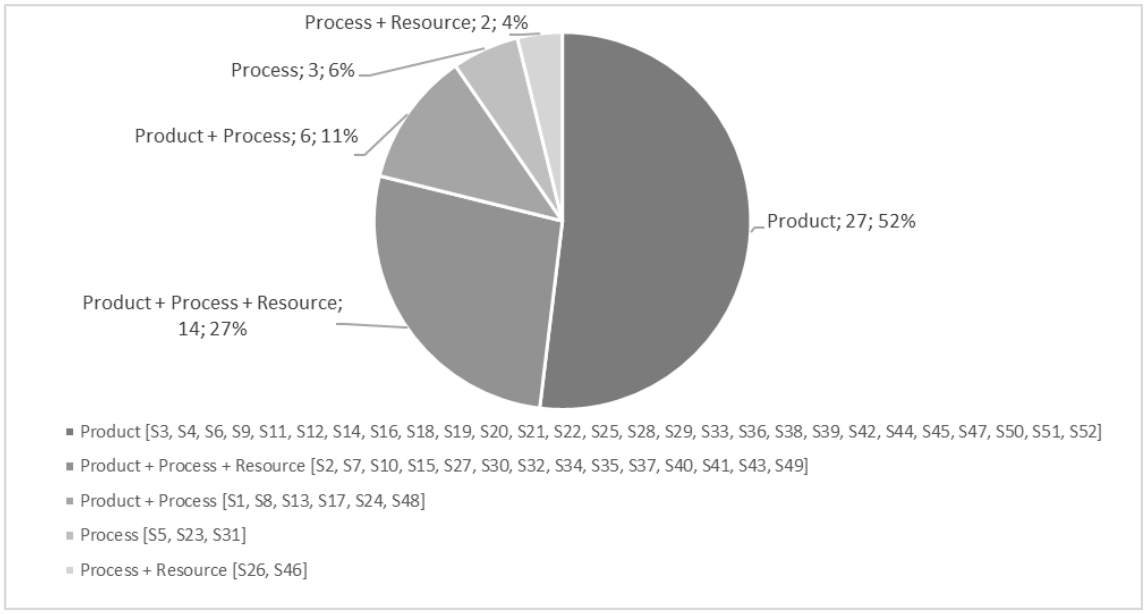


Figure 3.6. Individual distribution of software entities

Overall, 47 studies (53% of total) covered product entity related metrics to collect data for early defect prediction. Twenty-five studies (29%) included process entity-based data and 16 studies (18%) covered resource related data. The cumulative distribution of the software entities used in studies can be seen in Figure 3.7. It can be seen that product was the most common to measure since it is more concrete and there is a room for further studies that address process and resource entities in building ESDP models.

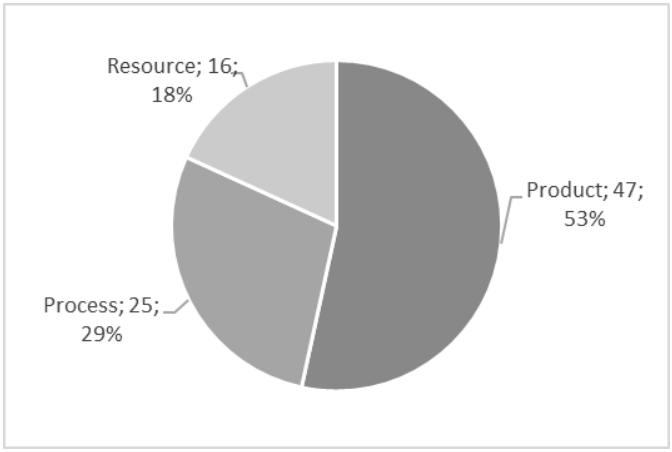


Figure 3.7. Cumulative distribution of software entities

3.4.1.4. RQ1.4 What are the attributes of each entity, which originate metrics for the prediction models?

Software attributes associated with each software entity were classified based on [54,102] as shown in Table 3.2. Accordingly, product structure, size, process effort and human resource characteristics were the most included attributes in the prediction models.

3.4.1.5. RQ1.5 What are the software metrics that are used in the prediction models?

Software metrics associated with each software attribute have been classified based on [54,102] as shown in Table 3.3. According to the table, lines of code (LOC) or number of use cases, McCabe's and Halstead's complexity metrics, requirements stability and staff experience were the most used metrics in ESDP models.

Table 3.2. Software attributes and referencing studies

Software entity	**Software attribute**	**Explanation of the attribute**	**References**	**# of Refs**
Product	Size	Identifies the magnitude of the work products such as LOC or number of use cases.	S1, S2, S50, S33, S29, S37, S15, S21, S28, S49, S32, S11, S12, S3, S9, S4, S47, S35, S27, S13, S42, S34, S20, S10, S25, S16, S19, S36, S41	29
	Structure	Covers the flow of the work products such as Complexity, Length, Coupling, Cohesion, Modularity or Reuse.	S2, S51, S52, S50, S33, S29, S21, S17, S28, S8, S18, S38, S32, S11, S12, S3, S9, S4, S47, S14, S40, S35, S42, S24, S34, S43, S20, S44, S30, S6, S39, S22, S7, S48, S25, S16, S36, S19, S45	39
Process	Effort	Covers the measures related to the effort of a process activity.	S1, S2, S5, S37, S23, S31, S26, S8, S32, S40, S35, S13, S46, S24, S34, S43, S30, S7, S10, S48	20
	Time	Covers the measures related to the time for a process activity.	S15, S31, S41	3
	Stability	States the changefulness of a process artifact.	S2, S37, S17, S8, S49, S32, S35, S27, S24, S34, S43, S30, S7, S10, S48	15
	Process Maturity	States the maturity of the organization about the process activities.	S2, S37, S8, S32, S40, S35, S24, S34, S30, S7	10
	Number of Defects	Specifies the number of defects found during a process activity.	S1, S37, S15, S17, S8, S49, S35, S27, S13, S24, S30, S7, S10, S48	14
	Adequacy	Represents the quality or completeness of a process artifact.	S2, S37, S49, S40, S35, S34, S43, S7, S41	9
Resource	Project characteristics	Covers the magnitude or quality of the input elements for software production, such as number of stakeholders, development language.	S37, S15, S26, S49, S35, S46, S41	7
	Human characteristics	Covers the personnel or team's quality for the activities, such as experience, motivation.	S2, S37, S15, S26, S49, S32, S40, S35, S27, S46, S34, S43, S30, S7, S10, S41	16

Table 3.3. Software metrics and referencing studies

Software entity	**Software attribute**	**Software metrics**	**References**	**# of Refs**
Product	Size	LOC or number of use cases	S2, S37, S15, S21, S49, S32, S47, S35, S27, S13, S42, S34, S10, S19, S41, S16, S36,	17
		Size of artifact	S1, S13	2
		Size metrics from NASA projects (Halstead size metrics)	S50, S33, S21, S28, S11, S12, S3, S9, S4, S25	10
		Requirement metrics: action, conditional, continuance, imperative, incomplete, option, risk level, source, weak phrase	S33, S28, S11, S12, S3, S9, S4, S20, S25	9
	Structure	McCabe Metrics (Complexity etc.) Halstead Metrics (total number of operators, operands etc.)	S52, S21, S17, S47, S42, S44, S30, S39, S22, S7, S48, S19	12
		Object-oriented Metrics (Complexity, Length, Coupling, Cohesion, Modularity, Reuse) Design metrics from UML [55]	S51, S50, S29, S18, S14, S6, S16, S36, S19	9
		Data flow complexity, cyclomatic complexity	S8, S24, S43	3
		Requirements complexity, Complexity of new functionality	S2, S37, S32, S35, S34, S48	6
		Program dependencies	S38	1
		Design metrics: edge count, node count, branch count, decision count, multiple condition count and condition count, densities, complexities	S20	1
		Architectural design metrics to quantify SDL (Specification and Description Language) blocks	S45	1
Process	Effort	Design, review or development effort measured in person hour	S1, S5, S23, S37, S31, S40, S35, S13, S43	9
		Creation effort, review effort	S26, S46	2
		Design review effectiveness	S30, S7	2
		Review, inspection and walkthrough (RIW)	S2, S8, S32, S24, S34, S30, S7, S10, S48	9
	Time	Total months of the project duration	S15, S31, S41	3

	Stability	Requirements stability (RS), requirement change request	S2, S37, S17, S8, S32, S35, S27, S24, S34, S43, S30, S7, S10, S48	14
	Process Maturity	Capability Maturity Model Integration (CMMI) Level	S2, S37, S8, S32, S40, S35, S24, S34, S30, S7,	10
	Number of Defects	Number of defects from review	S1, S37, S15, S35, S13	5
		Requirement fault density, design defect density, fault days number, code defect density	S15, S17, S8, S27, S24, S10, S48,	7
	Adequacy	Analysis, design, review quality	S37, S40, S35, S43	4
		Quality of documented test cases	S35, S7, S41	3
		Defined process followed	S2, S32, S34, S35, S37	5
Resource	Project characteristics	Number of stakeholders/members	S15, S49, S41	3
		Development language	S37, S15	2
		Configuration management	S37, S35, S41	3
		Project planning	S37, S35	2
		Scale of distributed communication	S37, S35	2
		Vendor management	S37, S35	2
		DBMS type, development solution, industry area	S15, S41	2
		Techno complexity	S26, S49, S46	3
		Urgency	S46	1
		Novelty to developer	S49	1
	Human characteristics	Staff experience	S2, S37, S32, S40, S35, S27, S34, S43, S7, S10	10
		Staff motivation	S37, S35	2
		Programmer capability	S37, S35, S30, S7	4
		Staff training quality	S37, S35	2
		Internal communication/interaction	S37, S35	2
		Productivity	S15	1
		Practitioners level	S26, S46	2
		Stakeholder involvement	S2, S32, S34	3
		People dependence	S41	1

3.4.1.6. RQ1.6 What types of methods are used to build the prediction models?

Figure 3.8 shows the distribution of the prediction methods used for early defect prediction in the studies. It can be seen that machine learning-based methods were the most frequently used (with 39%). Machine learning methods included support vector machines, artificial neural networks, genetic algorithms, K-means clustering, decision

trees and so on. Fuzzy logic-based methods (28%) were widely preferred since fuzzy logic is appropriate for handling qualitative data gathered from early phases. In addition, Bayesian network-based methods were preferred (with 13%) thanks to its capability to measure abstract data, which exists in early phases. Statistical methods, which are mostly based on regression, were used for early prediction with the percentage of 20%.

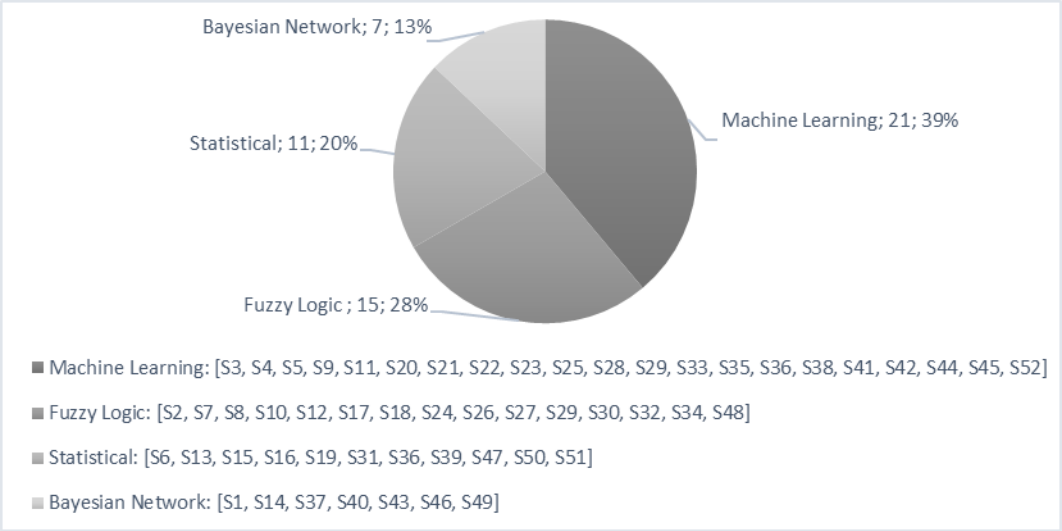


Figure 3.8. Distribution of prediction methods

3.4.1.7. RQ1.7 What are the contextual parameters reported in the prediction models?

The contextual parameters were gathered according to some references, such as [19] and [103]. It was investigated whether the studies reported the contextual parameters of the dataset explicitly or not. However, it was also possible for a study to address the contextual parameters in an implicit way. For example, if a study used NASA MDP data from PROMISE repository for early defect prediction, its contextual parameters can be inferred since the dataset is public to access. Besides, the contextual parameters about the NASA MDP dataset are known through the studies that reported this information explicitly, such as [S21, S44]. Overall, 14 studies [S3, S4, S9, S11, S12, S18, S20, S21, S22, S25, S28, S33, S42, S44] used NASA MDP dataset. In addition, some explicit contextual parameters were reported for public dataset published by Fenton et al. [S37], where 10 studies [S2, S7, S10, S27, S30, S32, S34, S35, S37, S43] used this dataset.

Lastly, a public raw data was published in [S16] and [S36] also used this dataset in their study.

Reported contextual parameters of these public datasets are given in Table 3.4, which include business domain, product size (as KLOC), programming language, development methodology, and effort.

Table 3.4. Context parameters of the public datasets

Public Dataset	**# of Studies Use the Dataset**	**Business Domain**	**Size (KLOC)**	**Programming Language**	**Development Methodology**	**Development Effort**
NASA MDP [58]	14	X	X	X		
Fenton dataset [20]	10	X	X	X	X	X
Data published in Cartwright and Shepperd [S16]	2	X	X	X	X	

Aside from these public datasets, the contextual parameters reported in 18 studies out of 25 studies that used private dataset were extracted. Figure 3.9 shows those parameters and the distribution of numbers among the studies. It is seen from the figure that the most reported contextual parameter (with 25%) was domain information of the projects. Also, technical information of the software product was given by reporting programming language (19%), size of the product (16%), and the type of the system (14%). In addition to that, it was mentioned whether the software was commercial or not (14%). Some other information about the quality requirements or processes was reported, such as criticality or quality expectancy from the system, and development methodologies adopted during the life cycle of the software. Unfortunately, 10 studies (out of 52) did not address any information regarding the context of the data used. It is a disadvantage that studies reporting the context were relatively few, which makes it difficult to repeat the study and compare the model performances based on contextual similarity.

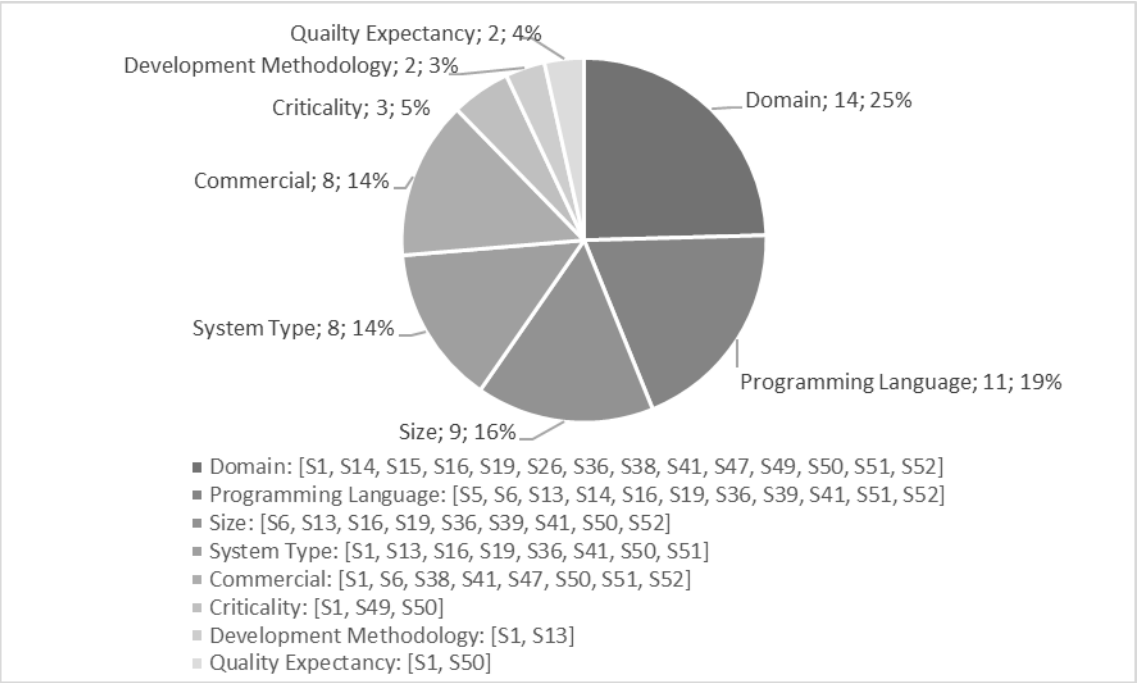


Figure 3.9. Categories of contextual parameters reported in 18 primary studies

3.4.1.8. Observations on review by characteristics of models

- Increased interest in public datasets is critical in terms of questioning the reproducibility of the studies. It is good to see that public datasets have gained interest through the years.
- SDLC phase information is important on ESDP studies, since we define "early" studies as the ones that built the prediction models before coding phase has started, i.e. in requirement or design stages. Approximately 60% of the primary studies focus on requirement or design phases to construct their prediction models, which indicates the importance of these phases in ESDP.
- It was observed that metric data based on product entity is mostly preferred in building ESDP models in the studies, while metric data based on process and resource entities follow that category.
- Most interested attributes are product size and structure, process effort, and human resource characteristics.
- Most commonly used metrics can be listed as follows: metrics that measure the length of the software product (i.e. LOC or number of use cases), complexity

related metrics (i.e. McCabe or Halstead metrics), effort for review activities, stability of requirements, maturity of the organization (i.e. CMMI level), and experience of the staff.

- On the side of prediction methods used in the models, machine learning and fuzzy logic methods are the most frequently chosen ones. It is worth to note that, fuzzy rule-based models are relatively suitable to model the vague, incomplete, or qualitative data gathered from the early phases. That is why fuzzy logic-based approaches are preferred frequently in building ESDP models.
- It can be said that contextual parameters have importance in the early phases of software development, since qualitative data is commonly used to construct the prediction models. Context information may undertake the task of guiding and can be helpful to build simple and effective models.

3.4.2. RQ 2. Are models of ESDP successful and beneficial?

3.4.2.1. RQ2.1 Which methods and measures are used for evaluating the performance of the models?

Performance evaluation methods of the prediction results varied according to the dependent variable of the model, which in general were defectiveness and number of defects, corresponding to categorical and continuous performance evaluation, respectively. The distributions related to performance evaluation methods were given in Figure 3.10. It can be seen that more than half of the studies used continuous performance evaluation methods, while nearly one-quarter of them used categorical methods. Unfortunately, nine studies (17%) did not evaluate the performance of the prediction models.

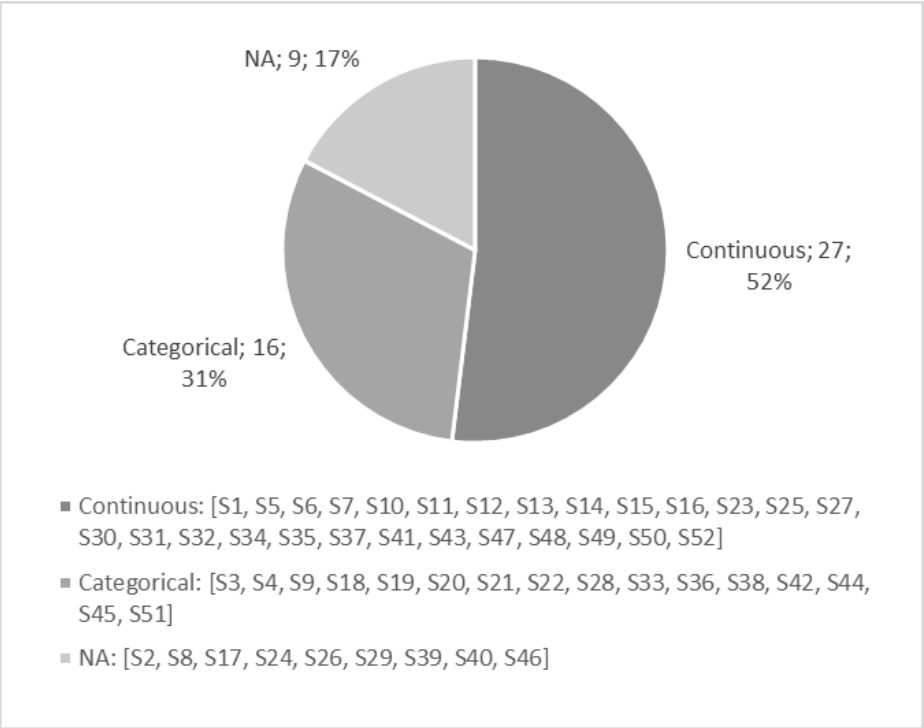


Figure 3.10. Distribution of the prediction performance methods

As mentioned above, categorical studies focused to predict whether the specific part of the software was defect-prone or not. Papers reported the prediction performance using ROC (Receiver Operating Characteristic), AUC (Area Under Curve), Probability of Detection (PD, Recall), Probability of False Alarms (PF), Precision, Accuracy, and F-measure. Continuous studies, which predicted the number of the defects, reported the prediction performance using various measures. Most of the measures reported by continuous studies were related to error measures, goodness-of-fit, ranking results, accuracy, or difference between expected and observed results. The distributions related to performance evaluation measures for categorical and continuous models were given in Figure 3.11 and Figure 3.12, respectively.

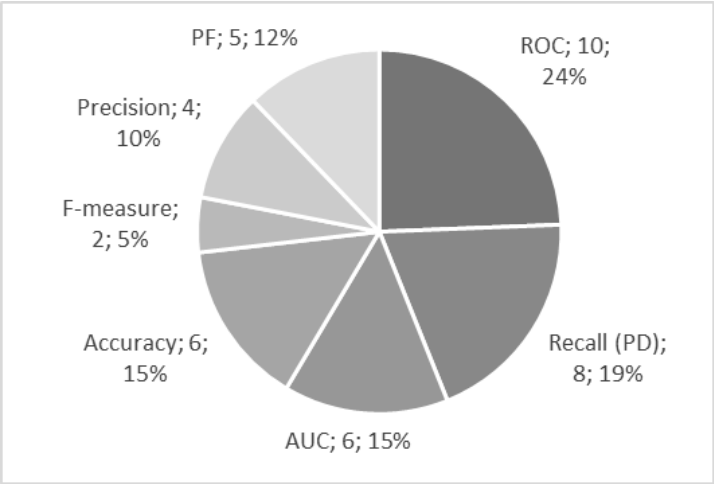


Figure 3.11. Performance evaluation measures in categorical models

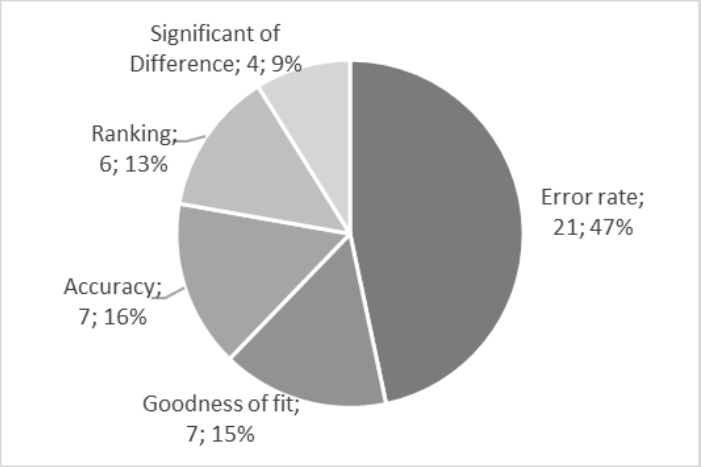


Figure 3.12. Performance evaluation measures in continuous models

3.4.2.2. RQ2.2 What are the performance values of the models based on the included SDLC phases that originate metrics for prediction?

Performance data of the prediction was extracted for every individual ESDP model given in the papers. We collected the performance values for each model presented in the related paper and synthesized the values with regard to phase information of the model. Note that we used the notation "*<phase>* (n = *<number of models>)"* in the tables reported in this section, to be able to provide the number of models presented in the papers with regard to the phase information of the constructed model. It is important to say that there is a one-to-many relationship between a primary study and the number of models it presents,

and ‘n’ values belong to the sum of the individual models presented in each study with regard to a specific phase.

Most of the categorical studies reported AUC or Precision, Recall, and F-measure, therefore we analyzed the results through these measures. Also, we provided f-measure where it was not reported by the paper directly, as it can be calculated from precision and recall. In order to interpret performance evaluation results, we used box-plots that are beneficial to show the differences between populations visually as they do not make assumptions about the distribution of the data [8]. Therefore, we provided the categorical performance results with regard to phase information by using two different box-plot graphics, in order to observe its likely effects on prediction performance. Figure 3.13 shows the results based on AUC values; while Figure 3.14 shows the results based on precision, recall, and f-measure values that were provided. It is very important to see that models based on requirement and design phase metrics were very successful based on both AUC and f-measure values, which were pretty close to 1.0.

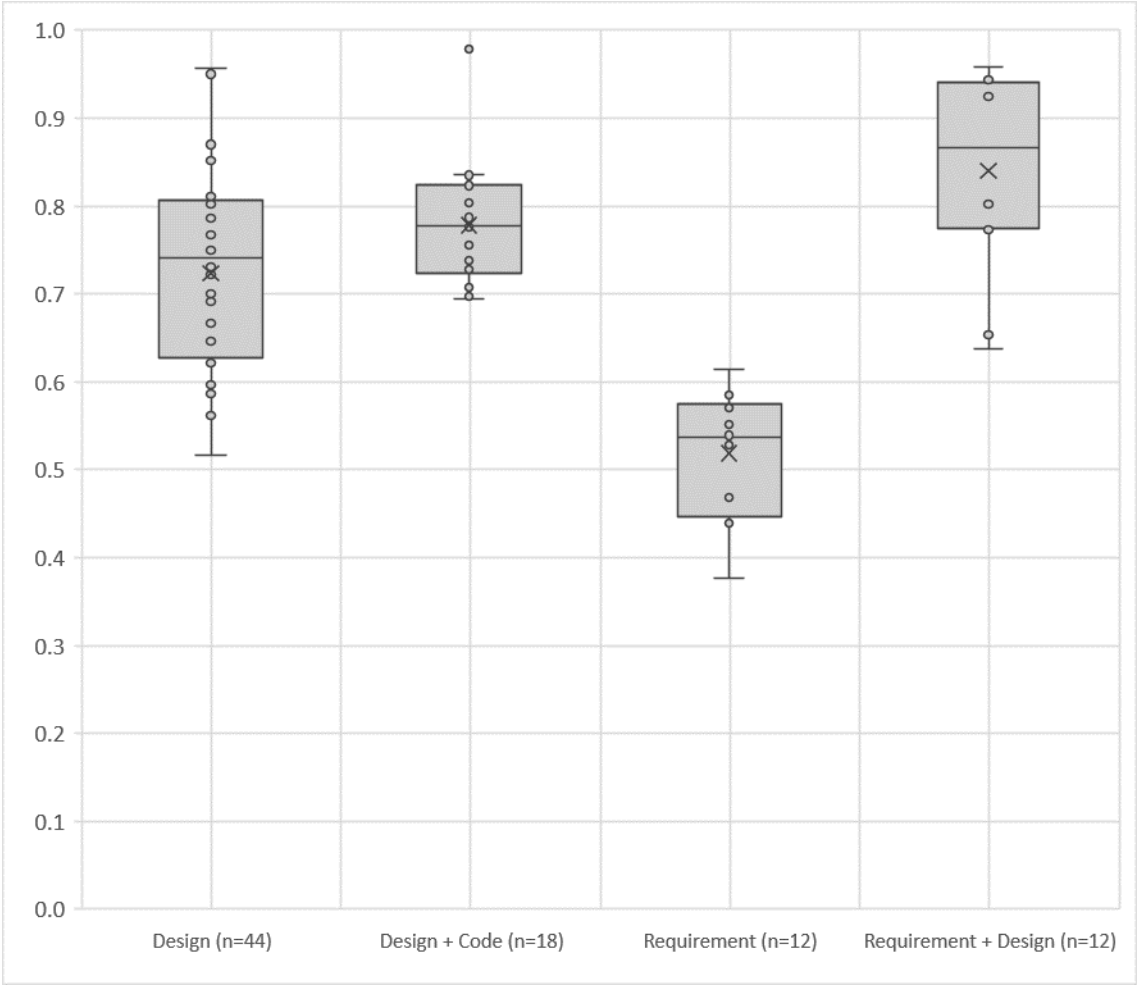


Figure 3.13. Performance results (AUC) regarding phase in categorical studies

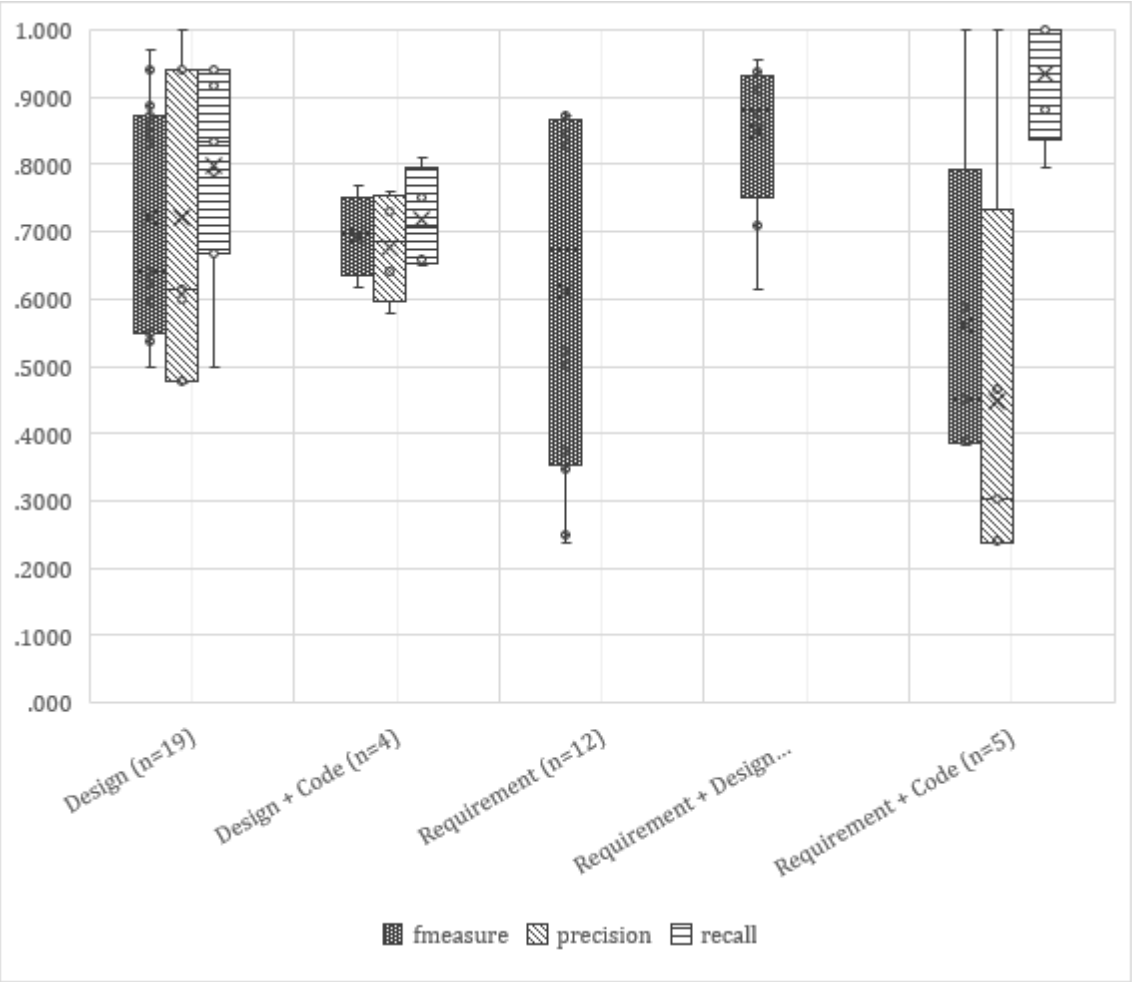


Figure 3.14. Performance results (f-measure, precision and recall) regarding phase in categorical studies

For the continuous studies, the prediction performance results were reported in a variety of measures, which makes it difficult to convert the results into a common measure. Mostly preferred performance measures reported in continuous studies were based on error measures, which are Mean Magnitude of Relative Error (MMRE), Root Mean Square Error (RMSE), Balanced Mean Magnitude of Relative Error (BMMRE), and Mean Absolute Error (MAE). MMRE results with regard to phase information were provided in Figure 3.15, which were reported in 10 studies [S7, S10, S15, S27, S30, S34, S37, S43, S48, and S49]. Except an outlier value reported in [S37], which belonged to a Bayesian network-based model built with data from all phases, it can be seen that most MMRE results were smaller than 0.5. In addition, it is very important to see that three models including only requirement phase-based data [S10, S15, S49] resulted in an MMRE value of approximately 0.28, which was smaller in comparison to the error value

of the models based on requirement and coding phase data in [S27]. Also, models based on requirement and design phase-based data in [S48] and design phase-based data in [S15] reported good performance values, which were MMRE = 0.098 and MMRE = 0.2, respectively. Besides, it is important to note that these models were based on different kinds of prediction methods (i.e. Bayesian networks, fuzzy rule-based and statistical techniques), which might have had an effect on the performance of the prediction apart from the phase information. Still, despite the differences in prediction methods, ESDP models demonstrated desired (high) performance.

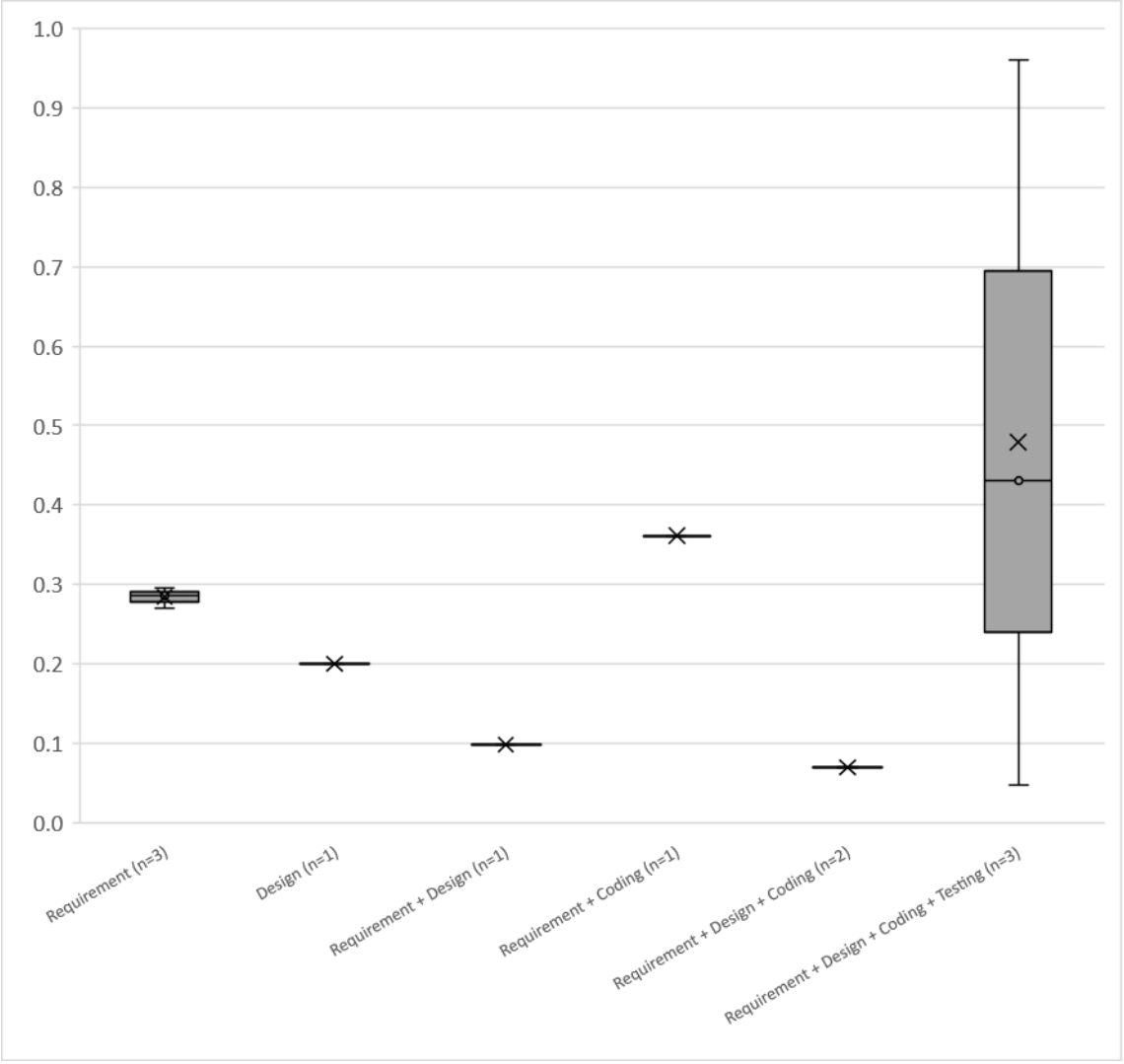
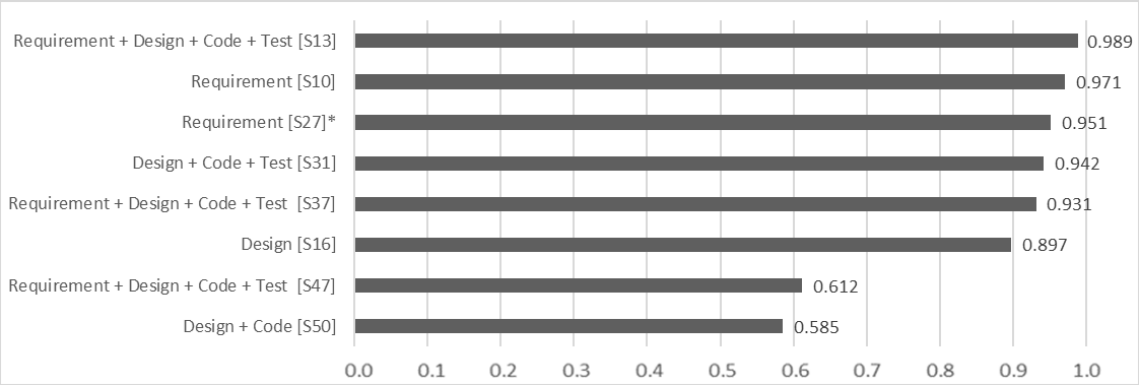


Figure 3.15. Performance results (MMRE) regarding phase in continuous studies

Moreover, R^2 values were also preferred among continuous studies. We provide those results with regard to the phase information in Figure 3.16. It can be seen that the most successful model [S13] was built with integrating data from the requirement, design, coding, and testing phases together (with R^2 = 0.989). Two studies [S10] and [S27] presented an ESDP model based on data only from the requirement phase with the performance values very close to 1.0, which were R^2 = 0.971 and R^2 = 0.951, respectively. These two distinctive studies demonstrate the power of requirements stage in the performance of ESDP models.



** Retrieved from [S10]*

Figure 3.16. Goodness-of-fit (R^2) values reported in continuous studies

3.4.2.3. RQ2.3 What are the benefits of early defect prediction as reported in the studies?

Only few of the studies, i.e. [S37] and [S49], both using Bayesian Network models, reported comprehensive benefits of the ESDP. In [S37], it was indicated that an obvious benefit of a Bayesian Network was its capability to organize a range of decision analysis and risk assessment modeling, which were conceivably important for software project managers. In addition, decision support capability was explained with example scenarios, in which the model parameters were changing regarding to the values of others, especially when the resource constraints made some of them impossible to increase. In [S49], the usability of the model was evaluated by using data (e.g. size of artifacts, number of defects) collected for five historical projects. Knowledge of seven domain experts was gathered by using questionnaires in order to build the prediction model, which required 112 min per expert. The results indicated that the model was useful for quality assurance

(QA) planning by identifying high-risk projects. Moreover, this also applied for QA controlling by providing better prediction for the number of defects than models using only measurement data. Consequently, it was stated that the proposed hybrid prediction model would be used in the software requirements phase of the company to support QA activities.

Aside from these two studies, most of the other studies concluded with a couple of general findings, which represented the benefits of early models verbally. We have categorized those benefits with regard to the mostly reported benefits in the primary studies. Table 3.5 presents the benefits of early software defect prediction and highlight the main focuses that the ESDP models can be used advantageously. It is worth noting that; for better clarification of this RQ, we performed "reciprocal translational analysis" reported in Dixon-Woods et al. [104]. This technique is helpful in order to analyze and synthesize the qualitative data and translate the main benefits reported across primary studies to the headings to identify the similarities between them.

Table 3.5. Reported benefits of early software defect prediction

Benefit ID	**Benefits Focus**	**Reported Benefits**	**Primary Studies**	**# of Studies**
B1	Useful for software practitioners in requirement phase	ESDP models can be beneficial to software engineers, managers and researchers for defect prediction in the requirement phase of software development.	[S10, S49]	2
B2	Useful for software practitioners in design phase	Experiments resulted in the fact that design metrics can be used accurately as software defect indicator in early phases of software development.	[S16, S19, S22, S29, S36, S44, S51, S52]	8
B3	Supports making best design decisions with the help of design phase metrics	Design phase-based metrics are good predictors of software defects, thus they support for selecting the suitable design among the available different design choices by avoiding defect-prone areas of the software.	[S6, S14, S38]	3
B4	Improved and effective resource planning	ESDP provides a basis for effective resource planning and utilization by allocating the necessary resources (human, computer of infrastructure) optimally.	[S2, S3, S4, S5, S7, S8, S9, S11, S15, S18, S20, S23, S24, S25, S28, S30, S32, S43, S46, S48]	20

B5	Improved testing focus and effective testing effort planning	ESDP models can be used for prioritizing software testing activities effectively with a specific focus on defective parts of the software in a comprehensive way, hence enable developers, testers or verification experts to concentrate their time and resources on the problematic areas.	[S5, S9, S10, S11, S12, S13, S18, S19, S23, S25, S28, S31, S33, S35, S38, S41, S43, S46, S47]	19
B6	Developing cost effective software and providing cost reduction	Identifying defective parts of the software early in the SDLC may lead to reduce cost by better planning and management of the project. Early identification of cost overruns and making corrective actions enable the software teams for developing cost effective software.	[S2, S7, S8, S9, S10, S18, S24, S30, S32, S42, S45]	11
B7	Useful in optimizing software schedule	Early prediction of defects supports software managers through improved scheduling and early identification of schedule mismatch.	[S9, S10, S30, S32, S35]	5
B8	Helpful for developing more reliable software	Predicting defects early in the SDLC can be used to achieve high software reliability by making effective strategies for improving the reliability of the whole system and deciding the necessary amount of corrective actions is achieved or not in order to achieve target software reliability.	[S2, S6, S7, S8, S12, S14, S17, S24, S32, S35, S47]	11
B9	Effective project planning and management	Early life cycle prediction can play an important role in project management by supporting software quality engineering through highlighting the quality needs earlier. Involving early phase risk mitigation and planning frequent review activities may also provide better software project management.	[S5, S15, S23, S31, S33, S35, S40, S51]	8
B10	Effective decision-support	ESDP provides effective decision-support and enables to make correct decisions regarding rework, testing and release planning. Software developer may easily detect the defective artifacts and may make correct decisions accordingly.	[S7, S20, S23, S30, S37, S40]	6
B11	Trade-off analysis	ESDP models provide to make effective trade-off analysis during early phases of software development.	[S20, S37]	2
B12	Improved software process control	Early prediction is used to improve software process control by early identification of software development process issues, therefore will be helpful for taking corrective actions through process improvement.	[S12, S30, S35]	3

3.4.2.4. Observations on review by performance of models

Regarding performance evaluation methods, most studies choose to predict the number of defects that exist in the software (i.e. continuous studies); hence they prefer to report performances based on measures related to error-rate.

We extracted performance values of continues studies with regard to MMRE and R^2 values. It is very important to see that studies include only requirement phase-based data, only design phase-based data, and requirement/design phase-based data together reported good performance values, in terms of MMRE values smaller than 0.28. We can also say that two studies [S10] and [S27] presented models based on data only from the requirement phase with $R^2 = 0.971$ and $R^2 = 0.951$, respectively, which may indicate the power of requirement phase-based data for ESDP.

When we look at the phase-based performance values of the categorical models, we see that model types established from the early-stage knowledge are successful. One of the most important finding of this systematic review is that models based on requirement and design phase metrics are very successful based on both AUC and f-measure values, which are pretty close to 1.0.

The main benefits of the ESDP as reported in the studies can be listed under several topics:

- It can be beneficial to software project managers by supporting early planning and management of project with higher quality in requirement or design phases of software development.
- It may provide a basis for effective resource planning by allocating the necessary resources optimally.
- It can be useful for planning of testing activities effectively, reducing the testing effort, and focusing the defective parts of the software in a comprehensive way as defect-prone areas will be already known.

- It may be used as a decision analysis mechanism during early phases of software development by supporting design decisions and helping the developers to select the suitable design choice by avoiding defect-prone areas of the software.
- The cost of the software development could be optimized and even may be reduced through early defect predictors.
- Early software defect prediction helps software managers on planning schedule effectively.
- High software reliability may be achieved and guaranteed early in the SDLC, by identifying the defective parts earlier.
- Predicting defects early in the software life cycle may improve software process control with early identification of the issues in software development processes.

Consequently, early phase data can help to build more accurate models when combined with metric data from the coding phase, and provide more benefits than software defect predictors based only on metric data from coding and testing stages.

3.5. Software Defect Prediction in Turkey – A Survey Study from Industry (RQ3)

A survey study was conducted to take a picture of the applications on SDP in software companies in Turkey. Mainly, we wanted to get the opinions of people working in different companies in the sector, and gather the needs and expectations of the industry. The relevant survey can be accessed via the Google forms[4].

3.5.1. Survey Design

The questionnaire is structured in three parts. In the first part, the title information of the participant's company and some general information specific to the company are asked for statistical evaluation. In this context, there are questions such as quality certificates and activities carried out within the scope of quality management to determine the quality management approach of the company. Finally, it is asked whether software defect

[4] tinyurl.com/yc7ah7xt

prediction is applied in the company. The second part of the questionnaire is structured according to the answer to this question.

If it is stated that software defect prediction is applied in the company; to understand in detail how the defect prediction process works, the following questions are asked:

- How do you operate software defect prediction?
- For what purpose do you apply software defect prediction?
- At what phases of the software development life cycle do you predict defects?
- Which metrics do you use for software defect prediction?
- What approach(es) and/or tool(s) do you use to build the software defect prediction model?
- What do you think are the benefits or advantages of software defect prediction applications in your company?
- What do you think are the difficulties or disadvantages of software defect prediction applications in your company?

If it is stated that there is no software defect prediction in the company; the following questions are asked to generate recommendations to motivate the useful application of defect prediction in software companies:

- Why do you think software defect prediction is not applied in your company?
- What do you think would be the benefits if software defect prediction was being applied in your company?
- What kind of difficulties would you think if software defect prediction was being applied in your company?

The final part of the questionnaire asks the following questions to understand the need for guidance for software defect prediction from the early phases of SDLC:

- Do you think it would be helpful if there was a guide on how to operate the software defect prediction process from the early phases of life cycle?
- Is guidance needed for choosing the defect prediction method?
- Is guidance needed to identify the inputs and outputs of the defect prediction model?
- Is guidance needed for the creation of the defect prediction model?
- Is guidance needed on how to predict defects?
- Is guidance needed on how to evaluate defect prediction performance?
- What do you think, in addition to the above issues, could be included in a guideline for software defect prediction from the early phases of software development?

3.5.2. Results

A total of 35 people participated in the survey. The data provided by the participants are shared in Appendix-2. The most important results grouped by the research questions can be listed as follows.

RQ 3.1. If software defect prediction is applied, how does the company operate it?

- 28.6% of the participants stated that software defect prediction was applied in their companies.
- It was seen that 60% of the participants applied SDP to predict the number of defects, 50% for the prediction of defective components, and 50% for determining the severity of the defects.
- It is seen that defect prediction is mostly applied in the requirement analysis phase of the software development life cycle (60%). This result is critical for addressing early-phase information while predicting the defects. In addition, it is seen that defect prediction is applied with a rate of 50% during the design phase. It is understood that the coding and testing phases are preferred by 50% and 40%, respectively.

- It is seen that process metrics are used with a rate of 90% in companies where defect prediction is made. Also, 80% of the participants stated that they used product metrics and 60% stated that they used resource metrics.
- While it is seen that statistical methods / tools are mostly preferred as an approach to creating a prediction model (80%), it is seen that approaches based on expert opinion are used at a rate of 40% and machine learning approaches at a rate of 20%.

RQ 3.2. If the company is applying SDP, what are the advantages or disadvantages of applying it?

- The benefits / advantages reported by those who stated that defect prediction was applied in their companies can be expressed as: predicting possible risks in projects, its contribution on time and quality management, and controlling the number of defects that will appear in future versions.
- The difficulties / disadvantages of defect prediction were stated as: the lack of qualified human resources to apply prediction, the different dynamics of the projects and the inability to be used by the teams, while the possibility of incorrect prediction of the defects that may occur was reported as its disadvantage.

RQ 3.3. If the company is not applying SDP, what would be the benefits and/or challenges in applying SDP in your company?

- While 37.1% of the participants stated that no prediction was made, 34.3% of them stated that they did not know whether SDP was applied or not.
- In companies that do not apply SDP, time, budget and cost constraints come to the fore, while the lack of experienced personnel and the lack of know-how on SDP are among the reasons for not using SDP models.
- It was stated that if they would apply SDP in their companies, there would be an increase in efficiency and quality in the planning of development and testing processes, resource and time management could be made more efficiently, the

developed software could be produced with higher quality, thus increasing customer satisfaction, awareness, and reusability.

- It was stated that in companies that do not apply SDP, if estimation were made, it would be the most important difficulty to collect the necessary data for applying SDP models, and there might be difficulties in building SDP models correctly. Besides, it is thought that SDP would bring an extra cost and workload.

RQ 3.4. Is there a need for guidance on software defect prediction from the early phases of SDLC?

- 89% of the participants stated that a guide would be helpful in choosing the SDP method and determining the inputs and outputs of the model.
- 86% of the participants stated that there should be guidance on the building of the model, how to apply the prediction and how to evaluate its performance.

In addition to these results, the survey contributors stated that issues such as which model will be selected in which type of projects and/or sectors among different models, usage and example scenarios of those models, and the benefits of the defect prediction process to the companies can be included in the defect prediction guide.

In line with the information obtained from the literature review and the survey results, it was seen that a decision analysis method is required for the selection of the defect prediction method in the field of ESDP. In this direction, in the studies described in the next section, details are given for the steps of preparation, design and implementation of a decision analysis method that will provide a basis for the selection of the defect prediction method suitable for the early phases.

4. DECISION ANALYSIS APPROACH

Up to this section, we have explored the feasibility of early phase defect prediction by addressing the most important aspects of SDP models. Thus, it was deemed appropriate to adopt a broad and comprehensive decision analysis approach to answer the crucial question of this thesis: *"RQ4. How to select a method for early prediction of software defects?"*

In this section, the steps taken in order to systematically synthesize the information obtained as a result of the extensive literature review and to use it in the modeling of the decision analysis approach are explained by matching the related processes with the detailed RQs.

4.1. Design of Decision Analysis Approach

The design of the decision analysis approach can be seen in Figure 4.1. It consists of four components: the preparation for decision analysis approach, generating the knowledge base, modeling of the decision analysis approach and the application of the approach.

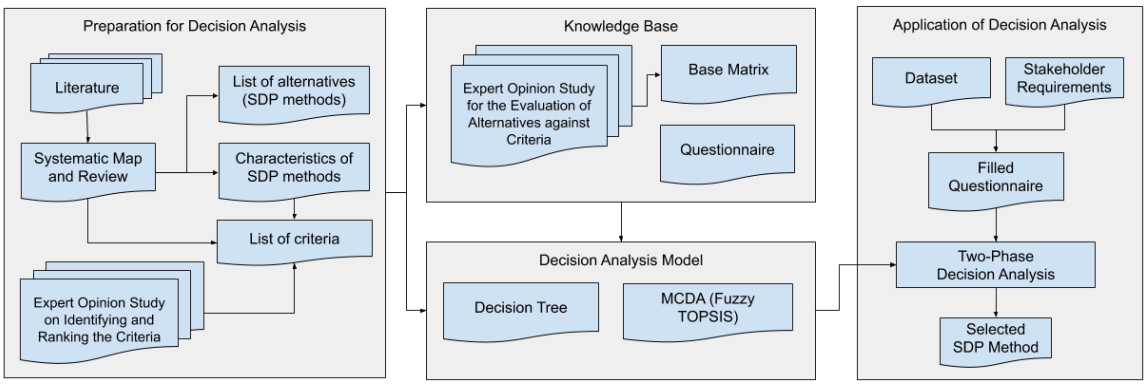


Figure 4.1. Design of the decision analysis approach

In the preparation stage for decision analysis, the literature was examined in detail as explained in Chapter 3.4, to reveal the current state of the early software defect prediction area. With this in mind, a list of alternatives to be compared during the decision analysis process was identified. After, several important characteristics that will be considered for

the selection of the alternative SDP methods, namely the criteria, were outlined. In doing so, an expert opinion study was prepared in order to gather opinions about the proposed criteria and to finalize them. The overall preparation process of decision analysis approach is given in Chapter 4.2-4.3.

The knowledge base contains all the data in a format that was derived from the previous stage. At this stage, a base matrix is defined, which contains the values that the criteria can take for each alternative. A second expert opinion study is conducted in order to finalize the base matrix, as well as to evaluate the alternatives against criteria. Chapter 4.4 covers the detailed steps executed to generate and develop the knowledge base.

In Chapter 4.5, a questionnaire is presented to collect the preferences of the stakeholders to guide the selection in line with the criteria and alternatives.

For modeling the decision analysis approach, all the information gathered in the knowledge base were synthesized. In this manner, a two-phase decision analysis approach that combines decision tree and MCDA methodologies is presented to form the decision analysis process for SDP method selection in the early SDLC phases. The decision analysis process is explained in detail in Chapter 4.6.

For the application of the decision analysis, the characteristics of the example dataset and the stakeholders' requirements are elicited through the proposed questionnaire. This allows the stakeholders to select the values of various attributes regarding their needs in the context of their software project and related defect dataset. In Chapter 5, the application of the decision analysis approach through several case studies were demonstrated.

4.2. What are the alternative methods for building ESDP models? (RQ4.1)

Based on our systematic literature review study on ESDP [15] and by considering other systematic reviews on SDP [8,10,82,83,105], several prediction methods were identified

to be considered as alternatives. In Table 4.1, these alternative methods and their basic characteristics were listed. The references to the primary studies were also provided in the rightmost column, which were helpful in retrieving the characteristics of the methods.

Table 4.1. Characteristics of software defect prediction methods

Method	**Approach to construct the SDP model**	**Purpose of use**	**Type of output**	**Dataset size**	**Primary Studies in [15]** [c]
ANN	Data Dependent	Classification, Regression	Categorical, Numerical	Medium / Large / Very Large	S5, S23, S25, S29, S35, S36
BBN	Can Address Both	Classification, Regression[a]	Categorical, Numerical[a]	No data required[b], Small / Medium / Large	S1, S14, S37, S40, S43, S46, S49
DT	Data Dependent	Classification, Regression[a]	Categorical, Numerical[a]	Large	S9, S33, S44, S52
FIS	Based on Human Judgement	Classification, Regression[a]	Categorical, Numerical[a]	No data required[b]	S7, S10, S12, S18, S27, S30, S32, S34, S48
LinR	Data Dependent	Regression	Numerical	Small / Medium / Large	S16, S47, S50
LogR	Data Dependent	Classification	Categorical	Small / Medium / Large	S19, S36, S51
NB	Data Dependent	Classification	Categorical	Small / Medium	S20, S21, S22, S42, S44
SVM	Data Dependent	Classification, Regression[a]	Categorical, Numerical[a]	Medium / Large	S38, S45

a. May depend on the implementation of the algorithm
b. Can be constructed independent from data
c. Full references of primary studies can be obtained

4.3. What are the criteria to consider when selecting a method for ESDP? (RQ4.2)

4.3.1. Initially Defined Criteria

The criteria that should be considered in the context of ESDP for the evaluation of the identified alternatives were determined and grouped under five main headings. The relevant criteria were defined roughly before the preparation stage of the decision analysis, which were first published as a conference paper [33], then matured and updated with various feedbacks received from the experts (e.g. in conference peer-reviews or expert opinion study described in the next sub-section).

To put it concretely, basic characteristics of the prediction methods have been considered for the determination and grouping of criteria, as well as the information required to build an SDP model in the early phases, such as data characteristics, data quality and the context information of the project. These criteria have also been mentioned in literature in various ways [8,37,38,41]. The grouping for the criteria is given as follows:

- *Model Construction (MC)*: The main purpose and model constructing approach are discussed under this group.
- *Data Characteristics (DCh)*: There are several characteristics which are crucial to address the constraints of the data that will be used for building the SDP model.
- *Data Quality (DQ)*: The quality characteristics of the data to be used to construct the SDP model are discussed under this group.
- *Method Characteristics (MCh)*: The characteristics of the methods to be used to construct the SDP model are discussed under this group.
- *Project Context (PC)*: The factors related to the context information of the project subject to SDP are discussed under this group.

Next, the definitions of the criteria under each grouping are given below.

Model Construction

- *Main purpose of use*: The purpose of an SDP model can be predicting the number of defects or classifying the software as defective / defect-free (i.e. prediction versus classification) [106]. This information is said to be distinguishable for both the construction of the model and for the performance evaluation of the resulting model [8].
- *Approach to construct the model*: To construct the SDP model, we can use machine learning based methods that learn from historical data and make predictions on new data, or we can prepare a model that is independent from data with the help of expert judgement [106]. It is necessary to evaluate the modeling technique since different techniques may produce different results under varying conditions [8].

Data Characteristics

- *Dataset size*: Dataset size is the size of the dataset that will be used for training the model. Small (number of examples (n) ≤ 500), Medium (500 < n < 1000), Large (1000 ≤ n < 10000), Very Large (n ≥ 10000) [81,94,107].
- *Type of input / output data*: Type of data can be categorical or numerical [54].

Data Quality

- *Causality*: Causality is the degree that attributes are dependent when the value of one attribute influences the other [21].
- *Uncertainty*: Uncertainty is the degree to which data is inaccurate, imprecise, untrusted or unknown [108].
- *Missing data*: Missing data is the values that are empty or left blank in the dataset [109].
- *Outlier*: Outlier is the degree to which the data do not meet with the general behavior of the dataset [110].

Method Characteristics

- *Interpretability*: Interpretability is the degree of which the user can understand the cause of any result (output) [37,111].
- *Complexity*: Complexity is the degree to which the method is complicated or complex in design [37].
- *Performance*: Performance is the degree of which the method performs well in general [112].
- *Speed*: Speed is the degree of costs associated with generating and using the method [37].
- *Maintainability*: Maintainability is the degree of which the method is easy to manage in time [41].

Project Context

- *Size of the artifact*: Size metric of the artifact subject to SDP can be used as a coefficient (normalizer) if the case is predicting the number of defects [21]. It is important to note that, the size of the artifact is defined as an indicator of the project rather than an indicator of the dataset.
- *Development methodology*: Development methodology is the approach used throughout the project's life cycle [15].
- *Development phase*: Development phase information can be considered as the phase information (requirements analysis, design, coding etc.) when the SDP model is constructed [21].
- *Domain*: Domain information is about the business domain of the project [15].

4.3.2. Expert Opinion Study on Identifying and Ranking the Criteria

To select the most suitable method for early software defect prediction, an expert opinion survey was prepared with a purpose of investigating the main factors (criteria) that were considered important for evaluating alternative SDP methods and weighting the determined criteria.

The survey was prepared in Google Forms and it consisted of four sections[5]. In the first section, there was an introduction part to inform the experts about the research conditions, with the terms of agreement. In the second section, the participants were asked about some personal information to be processed for descriptive statistics anonymously. In the third section, each criterion was presented under the related criteria group given in the previous section. The experts were expected to evaluate each criterion based on a scale that consist of six values: “Not Important”, “Very Low”, “Low”, “Medium”, “High”, and “Very High”. In addition, the experts were expected to select which of the relevant criteria might be important in the context of the early phases. In the last section of the survey, experts could submit a new criterion proposal and rate its importance within a scale of

[5] https://tinyurl.com/2e6tvcd5

"Very Low" to "Very High". The results of the expert opinion survey were given in Appendix-3.

The expert opinion survey was sent to twenty identified experts in the field via e-mail. At the end of the defined period, eight experts participated in the study. The descriptive information about the participant profiles is given in Table 4.2.

Table 4.2. The profile of the experts

Expert	Organization Type	Title	Level of knowledge in SDP (out of 5)	Experience on SDP (in years)	h-index	# papers in SDP	Years in Industry
E1	Government	Software Quality Manager	3	3 - 5 years			15
E2	University	Assistant Professor	5	6 - 10 years	24	21	
E3	University	Professor	5	> 20 years	35	34	
E4	University	Associate Professor	5	11 - 20 years	25	19	
E5	Government	Senior Software Engineer (PhD)	5	6 - 10 years			13
E6	Private Company	Senior Software Engineer (PhD)	5	6 - 10 years			12
E7	University	Associate Professor	4	3 - 5 years	16	10	
E8	University	Assistant Professor	4	6 - 10 years	16	20	

Figure 4.2 shows the responses of the experts for all the criteria questions. Each response reflects the opinion of an expert about the importance degree of the related criteria in the context of software defect prediction. Verbal scales are defined as VH, H, M, L, VL, and NI that correspond to “Very High”, “High”, “Medium”, “Low”, “Very Low”, and “Not Important”, respectively.

Criteria	E#1	E#2	E#3	E#4	E#5	E#6	E#7	E#8
MC - Main purpose of use	VH	VH	VH	VH	VH	VH	VH	H
MC - Approach to construct the model	H	H	M	H	VH	VH	H	VH
DCh - Type of the input	H	VH	H	M	L	L	H	VH
DCh - Type of the output	H	VH	H	H	L	L	VH	VH
DCh - Dataset size	VH	VH	H	H	H	VH	H	VH
DQ - Causality	H	VH	M	L	H	M	M	M
DQ - Uncertainty	VH	VH	H	L	H	VH	H	VH
DQ - Missing values	M	VH	H	L	H	H	H	M
DQ - Outlier	M	VH	H	H	H	M	M	H
MCh - Interpretability	H	VH	H	M	H	H	L	VH
MCh - Complexity	VH	H	H	L	M	M	M	VH
MCh - Performance	H	VH	H	VH	VH	H	H	VH
MCh - Speed	M	H	H	L	L	L	L	VH
MCh - Maintainability	VH	VH	H	L	H	M	H	VH
PC - Size of the artifact	L	M	M	H	H	VH	L	VH
PC - Development phase information	H	L	M	L	H	H	H	H
PC - Development methodology	VH	L	M	L	H	M	M	M
PC - Domain information	M	H	M	L	NI	M	L	M

Figure 4.2. Responses of the experts (E) regarding the criteria

As mentioned before, the expert opinions were gathered about which of the relevant criteria may be important in the context of ESDP. Based on the answers, we determined that it would be more appropriate to address the criteria that were selected for ESDP context. According to the frequency values of each criterion shown in Table 4.3, “Domain information” criterion was eliminated since it has not been selected.

Table 4.3. Frequency values of each criterion

Criteria	**# of selection for ESDP**	**# of responses**	**Frequency**
Main purpose of use	8	8	1
Dataset size	7	8	0.875
Approach to construct the model	6	8	0.75
Performance	6	8	0.75
Uncertainty	5	8	0.625
Interpretability	5	8	0.625
Maintainability	5	8	0.625
Development phase information	5	8	0.625
Type of the output	4	8	0.5
Type of the input	4	8	0.5
Missing values	4	8	0.5
Outlier	4	8	0.5
Complexity	4	8	0.5
Causality	3	8	0.375
Size of the artifact	2	8	0.25
Development methodology	2	8	0.25
Speed	2	8	0.25
Domain information	0	8	0

In addition, the elimination of the five initially defined criteria was decided in subsequent iterations. Below, the excluded criteria with their elimination reasons is presented.

- “Type of the output”: This criterion was found to have values that are directly parallel to the values of the “Main purpose of use” criterion. Therefore, this criterion was eliminated to avoid addressing the same information.
- “Type of the input”: It was found to be a neutral element, that is, it was not a criterion affecting the decision analysis during the evaluation of the alternatives. In other words, it was pruned from the decision tree according to the results of the second expert opinion study, because it had the same value for all alternatives.
- “Development phase information”: This criterion would not be evaluated in a meaningful way in the decision-making process, since it was not clear enough which phase information could be handled by any of the alternatives. Besides, its

existence is good for any SDP method but does not affect selection, as it can be used only as an input element when building phase-based SDP models.

- "Size of the artifact": It was found to be useful as an input element (rather than a criterion) when building an SDP model in the early phases.
- "Development methodology": It is decided to address in future studies for two reasons: i) it may not be possible to know the development methodology in open datasets that are subject to the most empirical studies, ii) the projects included in the NASA dataset, which was also used in our case study, were developed long before the advent of agile methodologies. For this reason, it is assumed the development method as plan-driven in the relevant dataset and decided to consider the update of the knowledge base to address this criterion as a future work.

4.3.3. Ranking and Weighting the Criteria

The first two criteria groups (i.e. Model Construction and Data Characteristics) were used in the decision tree analysis, where the last two criteria groups (i.e. Data Quality and Method Characteristics) were used in the Fuzzy TOPSIS analysis, as described in more detail in the following sections. The main reason for this distinction is the data type considered when evaluating the alternatives for the first two criteria groups is nominal, while the data type considered for the last two criteria groups is interval.

Ranking the criteria under MC and DCh

A data transformation was performed to rank the criteria in the first two groups. Following transformation is used to convert verbally collected responses to the numerical weights within a scale from 1 to 5:

- Very High (VH) corresponds to "5"
- High (H) corresponds to "4"
- Medium (M) corresponds to "3"
- Low (L) corresponds to "2"
- Very Low (VL) corresponds to "1"

To determine the criteria group order, the sub-criteria weights in the group were aggregated and a calculation was made by considering their average weights. After weighting a criteria group, each criterion was ranked in the group based on their median/mean values. Based on these values, criteria were reordered within the criteria group. In Table 4.4, the criteria group and criterion ranking with their resulted values (mean and median) determined from the results of this initial expert opinion study are demonstrated. These criteria would later be used in Phase - 1 of the decision analysis process, which is explained in Chapter 4.6.1.

Table 4.4. Numerical values of the expert opinions and mean / median values

Criteria Group	Criteria	E#1	E#2	E#3	E#4	E#5	E#6	E#7	E#8	Group Mean	Group Median	Criteria Mean	Criteria Median
MC	Main purpose of use	5	5	5	5	5	5	5	4	4.56	4.50	4.88	5.00
	Approach to construct the model	4	4	3	4	5	5	4	5			4.25	4.00
DCh	Dataset size	5	5	4	4	4	5	4	5	4.50	4.50	4.50	4.50

Weighting the criteria under DQ and MCh

As we used linguistic variables to gather the opinions of the experts, we needed to transform their values into the fuzzy numbers [113]. In fuzzy set theory, conversion scales are applied to transform the linguistic terms into fuzzy numbers. As for the last two criteria groups, aggregated fuzzy importance weights were calculated with regard to expert opinion study results given in Figure 4.2, since they would be used for the Fuzzy TOPSIS evaluation, which is explained in Chapter 4.6.2.

4.4. How should the most appropriate method be selected by evaluating the defined criteria? (RQ4.3)

In order to develop the decision analysis approach, it is important to reveal the knowledge base that would provide the necessary inputs to the decision analysis process. In doing so, first the results were collected from literature review [15].

To execute the decision analysis process, alternative SDP methods should be evaluated on the basis of the final set of criteria. The information in the literature was gathered and synthesized, which is necessary for the evaluation of alternatives against criteria. Even though our knowledge could allow us to make the evaluation on the basis of criteria for alternatives, there would be a possible validity threat of reflecting only the views of the authors. To eliminate this threat, the opinions of experts in the field were reflected to the evaluation, so that a more reliable and robust decision analysis process could be operated. At this point, the knowledge base was not only an input for the preparation of the second expert opinion study, but also the outputs of the expert opinion study enabled us to update and finalize the knowledge base. When the second expert opinion study was conducted, its results revealed the "base (decision) matrix", which should provide the basis for decision analysis.

4.4.1. Expert Opinion Study for the Evaluation of Alternatives against Criteria

This expert opinion survey was prepared to be helpful for creating the base matrix and selecting the most appropriate method for early software defect prediction. Therefore, the aim was to gather the expert opinions for evaluating the alternatives regarding each criterion in the context of ESDP. The experts were asked to complete the survey by rating the related criteria on each SDP method.

This survey was also prepared in Google Forms and it consisted of eight sections[6]. As in the previous survey, the first section was an introduction form to inform the experts about the research conditions, with the terms of agreement. In the second section, the participants were asked about some personal information as well as with their degree of knowledge and experience years both in SDP and in building / using of the prediction methods. In addition, the experts were expected to choose their expertise level on the alternative methods, since this information would be used as a basis for scoring the alternatives. In the following sections, the criteria groups were presented with included criteria and their explanations to be evaluated by the experts. In the last section of the survey, experts could note any additional comments that should be considered when

[6] https://tinyurl.com/mryyx5hx

selecting the SDP method in the early phases. The results of the expert opinion study on the evaluation of alternatives against criteria were given in Appendix-4.

The expert opinion survey was sent to fifteen identified experts in the field via e-mail. At the end of the defined period, four experts participated in the study. The descriptive information related to the participant profile were as follows: Two university staff with the title of assistant professor, one private sector staff with the title of senior software engineer (PhD), and one government staff with the title of senior software engineer (PhD). Two of the participants reported the degree of knowledge as 4 out of 5 while the other two reported it as 5. Two participants reported their years of expertise in the field of SDP as 3-5 years, and the other two participants as 6-10 years. Three participants reported the degree of knowledge in prediction methods as 4 out of 5, while the other participant reported it as 5. Lastly, one participant stated the years of experience on building / using of the prediction models as 3 - 5 years, where two participants stated it as 6 - 10 years and the other one stated it as 11 - 20 years.

The values reflected to the base matrix from the expert opinion study results are summarized in Table 4.5 for the first two criteria groups ("model construction" and "data characteristics"), and in Table 4.6 for the last two criteria groups ("data quality" and "method characteristics").

4.4.2. Base Matrix

In base matrix, possible values of the criteria were extracted for each alternative. For the first two criteria groups ("model construction" and "data characteristics"), base matrix was easier to fill in, since the values that the criteria could take on the basis of alternatives were more precise. However, for the other criteria groups ("data quality" and "method characteristics") it was not that easy to distinguish the values of the criteria and evaluate them clearly, since they contained uncertainty for decision makers. For example, to assess the "Interpretability" criterion, it was harder to answer the question "Do you think the following methods are interpretable?" than "To what extent do you think the following methods are interpretable?". The main reason for this was the first question is a "yes / no"

question, while the second question contains a scale that supports a linguistic rating. For this reason, the results of the second expert opinion study is used to finalize the base matrix, especially for those vague criteria groups. Table 4.5 demonstrates the final base matrix generated based on both the literature review and the expert opinions for the first two criteria groups. This table was then used as a basis for "Phase - 1: Decision Tree Analysis", as we explain in Chapter 4.6.1.

Table 4.5. Base matrix for the decision tree analysis

Method	**Main Purpose of use**	**Approach to construct the model**	**Dataset Size**
ANN	Classification, Prediction	Data Dependent	Medium / Large / Very Large
BBN	Classification, Prediction[a]	Can Address Both	No data[b], Small / Medium / Large
DT	Classification, Prediction[a]	Data Dependent	Large
FIS	Classification, Prediction[a]	Human Judgement	No data[b]
LinR	Prediction	Data Dependent	Small / Medium / Large
LogR	Classification	Data Dependent	Small / Medium / Large
NB	Classification	Data Dependent	Small / Medium
SVM	Classification, Prediction[a]	Data Dependent	Small / Medium

a. May depend on the implementation of the algorithm
b. Can be constructed independent from data

Table 4.6 demonstrates the final base matrix generated based on the expert opinions for the last two criteria groups. Here, the formula used for weighting the criteria (Eq. 4.1) was used similarly to evaluate the alternatives against criteria based on the responses of four experts participated in the second expert study. In other words, the aggregated fuzzy values of the alternatives with regard to the criteria were reflected in this base matrix. This table was then used as a basis for "Phase - 2: MCDA (Fuzzy TOPSIS)", as explained in Chapter 4.6.2.

Table 4.6. Base matrix (continued) for the Fuzzy TOPSIS evaluation

	ANN	BBN	DT	FIS	LinR	LogR	NB	SVM
Causality	1	3	1	1	1	1	1	1
	4	7.67	3.67	5.5	2.33	3	2.33	3
	7	9	7	9	5	7	7	7
Uncertainty	1	3	1	1	1	1	5	1
	5	7.67	4.33	4	3.67	3.67	7.67	4.33
	9	9	7	7	7	7	9	7
Missing Data	3	3	1	1	1	1	5	1
	5.5	7	5	5	4.33	4.33	7.67	5
	9	9	9	9	9	9	9	9
Outlier	5	3	1	1	1	1	1	1
	7	5.67	4.33	4.33	4.33	5	5	4.33
	9	9	9	7	7	9	9	9
Interpretability	1	3	5	3	5	5	3	1
	1	6.33	8.5	7.67	7.67	7.67	7	3
	3	9	9	9	9	9	9	7
Complexity	5	3	1	1	1	1	1	1
	8	6.33	2.33	3.67	1	1	2.5	5.5
	9	9	5	9	3	3	7	9
Performance	3	3	5	1	3	5	5	3
	7.5	6.33	7	4.5	6.33	7	7	5.5
	9	9	9	9	9	9	9	9
Maintainability	1	3	1	1	3	3	3	1
	4	6.33	3.67	5	6.33	6.33	6	3
	9	9	9	9	9	9	9	7
Speed	1	3	5	1	3	3	5	3
	3.5	5.67	8.33	4.5	7.67	7.67	8.5	5.67
	9	9	9	9	9	9	9	9

4.5. How should we gather the characteristics of the project data and the needs of the users systematically? (RQ4.4)

It should be re-emphasized that the choice of the method to be used to build the SDP model depends mostly on the data to be used, as well as the needs of the stakeholders to apply the model in the early phases [94,114]. For this purpose, a questionnaire was proposed to reveal the values of the criteria in a comprehensible manner in terms of being the basis of decision analysis. This questionnaire needs to be filled before the application of the decision analysis, each time, prior to the ESDP effort.

Questionnaire to address the needs of stakeholders in a systematic manner is given in Table 4.7 along with the criteria that each question is related to. The questions should be answered by “yes” or “no”. In order to facilitate the answering process, te selection type is also specified for each criteria group. If the criteria will be evaluated by answering multiple questions, the user is expected to answer more than one question.

Table 4.7. Questionnaire for evaluation of SDP methods in the early phases

Criteria group	**Criteria**	**Selection type**	**Questions**
Model Construction	Approach to construct the model	Select multiple	Do you want your method be dependent on data?
			Do you want to address human judgement?
	Main purpose of use	Select one	Do you want to perform classification?
			Do you want to make a numeric prediction?
Data Characteristics	Dataset size	Select one	Do you have a large sized dataset to train an SDP model?
			Do you have a medium sized dataset to train an SDP model?
			Do you have a small sized dataset to train an SDP model?
Data quality	Causality	Select multiple	Is there any dependency between data attributes? If yes, do you want to address these dependencies?
	Uncertainty		Is there any uncertainty in the data? If yes, do you want to address the uncertainty?
	Missing data		Is there any missing point in the data? If yes, do you want to handle the missing data?
	Outlier		Is there any outlier in the data? If yes, do you want to handle these outliers?
Method Characteristics	Interpretability	Select multiple	Is it important that SDP method has high interpretability?
	Complexity		Is it important that SDP method has low complexity?
	Performance		Is it important that SDP method has high performance?
	Maintainability		Is it important that SDP method has high maintainability?
	Speed		Is it important that SDP method has high speed?

4.6. Modeling the Decision Analysis Approach

4.6.1. Phase - 1: Decision Tree Analysis

According to the ranking of identified criteria groups presented in Table 4.4 and the base matrix given in Table 4.5, the first phase of the decision analysis process was designed based on the decision tree concept. The first two criteria groups ("Model Construction" and "Data Characteristics") were included in the decision tree construction. In fact, the tree was structured based on the rules extracted from the base matrix. Proposed decision tree is intended to recommend a subset of alternative methods in line with the needs and requirements of the practitioner. In Figure 4.3, proposed decision tree for the first phase of the decision analysis process was presented.

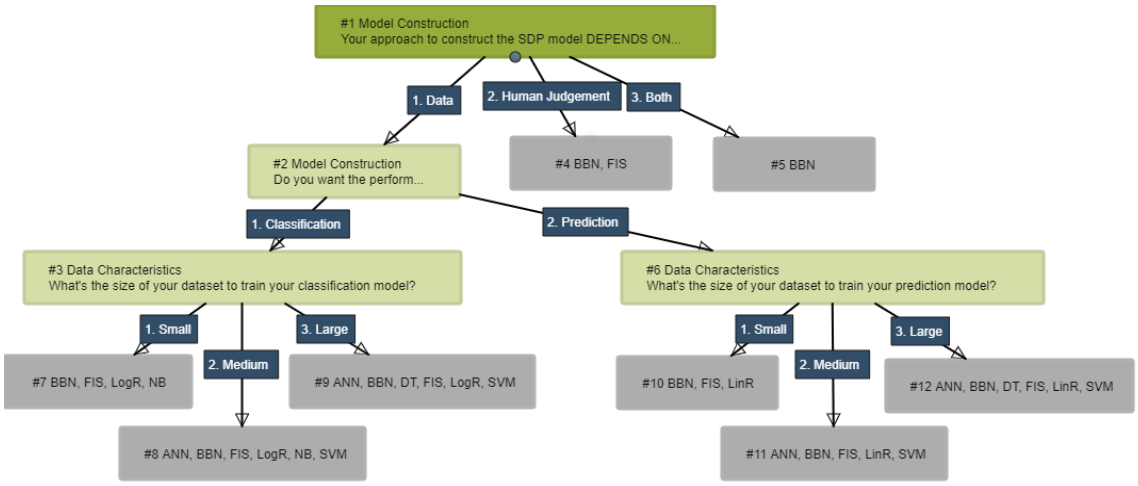


Figure 4.3. Decision tree for the phase–1 of the decision analysis process

4.6.2. Phase - 2: MCDA (Fuzzy TOPSIS)

According to the weights of all criteria presented in Table 4.5 and the base matrix given in Table 4.6, the second phase of the decision analysis process was designed based on the MCDA concept. The last two criteria groups ("data quality" and "method characteristics") were included in the construction of MCDA model. Proposed MCDA was intended to rank the subset of alternatives in line with the needs and requirements of the practitioner.

Fuzzy TOPSIS method was adopted for MCDA application due to its success in addressing the incomplete or vague information into the decision analysis process of an uncertain context like SDP method selection [115]. For ranking the alternatives, the implementation of the Fuzzy TOPSIS method was applied by following the steps proposed by Chen [78]. As given in the previous sections, the criteria and alternatives were identified and are summarized in Table 4.8. Below, the steps for the implementation of the Fuzzy TOPSIS algorithm is explained.

Table 4.8. Defined criteria and alternatives for Fuzzy TOPSIS application

Criteria ID	Criteria Name	Alternative ID	Alternative Name
C1	Causality	A1	ANN
C2	Uncertainty	A2	BBN
C3	Missing Data	A3	DT
C4	Outlier	A4	FIS
C5	Interpretability	A5	LinR
C6	Complexity	A6	LogR
C7	Performance	A7	NB
C8	Maintainability	A8	SVM
C9	Speed		

Step 1. Define the linguistic variables and their corresponding fuzzy values

The linguistic values used in the expert opinion study and their corresponding fuzzy values were determined as given in Figure 4.4.

Linguistic Values	Corresponding Fuzzy Values
Very Low (VL)	(1,1,3)
Low (L)	(1,3,5)
Medium (M)	(3,5,7)
High (H)	(5,7,9)
Very High (VH)	(7,9,9)

Figure 4.4. Linguistic variables and their corresponding fuzzy values

Here, we chose using triangular fuzzy numbers for the five linguistic variables as presented in [116] and applied a scale of 1 to 9 for weighting the identified criteria. In Table 4.9, we introduced the aggregated fuzzy weights for each criterion that we calculated based on the formula presented in [80]. Where, *K* is the total number of decision makers (in our case there are eight experts), and *W* is the weight of a criterion, the aggregated fuzzy weight $w\tilde{}_j = (w_{j1}, w_{j2}, w_{j3})$ for a criterion C_j is calculated as follows:

$$W_{j1} = \min_k\{w_{j1}^k\},\ W_{j2} = \frac{1}{K}\sum_{k=1}^{K} w_{j2}^k,\ W_{j3} = \max_k\{w_{j3}^k\} \qquad \text{(Eq. 4.1)}$$

Table 4.9. The aggregated fuzzy weights for the criteria under DQ and MCh

Criteria Group	**Criteria**	**Aggregated Fuzzy Weights**		
Data Quality (DQ)	Causality	1.00	5.75	9.00
	Uncertainty	1.00	7.50	9.00
	Missing Data	1.00	6.25	9.00
	Outlier	3.00	6.50	9.00
Method Characteristics (MCh)	Interpretability	1.00	6.75	9.00
	Complexity	1.00	6.25	9.00
	Performance	5.00	8.00	9.00
	Maintainability	1.00	7.00	9.00
	Speed	1.00	5.00	9.00

Step2. The decision makers (DMs) evaluate the ratings for each alternative with respect to each criterion by using the determined linguistic variables as shown in Figure 4.5.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
A1	Medium	Medium	High	High	Very Low	High	Medium	Low	Low
A2	Medium	Medium	High	Medium	Medium	High	High	Medium	Medium
A3	Medium	Medium	High	Very Low	Very High	Very Low	High	High	Very High
A4	Low	Low	Medium	Medium	Medium	High	Medium	Low	Medium
A5	Low	Medium	High	Medium	Very High	Very Low	Medium	High	Very High
A6	Medium	Medium	High	High	Very High	Very Low	High	High	Very High
A7	Medium	High	High	High	Very High	Very Low	High	High	Very High
A8	Medium	Medium	High	High	Medium	High	High	Medium	Medium

Figure 4.5. Decision matrix for DM1

Step 3: The ratings of the DMs is converted into the fuzzy decision matrix (Figure 4.6) by using triangular fuzzy numbers given in Figure 4.4.

	C1			C2			C3			C4			C5			C6			C7			C8			C9		
A1	3	5	7	3	5	7	5	7	9	5	7	9	1	1	3	5	7	9	3	5	7	1	3	5	1	3	5
A2	3	5	7	3	5	7	5	7	9	3	5	7	3	5	7	5	7	9	5	7	9	3	5	7	3	5	7
A3	3	5	7	3	5	7	5	7	9	1	1	3	7	9	9	1	1	3	5	7	9	5	7	9	7	9	9
A4	1	3	5	1	3	5	3	5	7	3	5	7	3	5	7	5	7	9	3	5	7	1	3	5	3	5	7
A5	1	3	5	3	5	7	5	7	9	3	5	7	7	9	9	1	1	3	3	5	7	5	7	9	7	9	9
A6	3	5	7	3	5	7	5	7	9	5	7	9	7	9	9	1	1	3	5	7	9	5	7	9	7	9	9
A7	3	5	7	5	7	9	5	7	9	5	7	9	7	9	9	1	1	3	5	7	9	5	7	9	7	9	9
A8	3	5	7	3	5	7	5	7	9	5	7	9	3	5	7	5	7	9	5	7	9	3	5	7	3	5	7

Figure 4.6. Fuzzy matrix for DM1

Step 4. The fuzzy decision matrix of each DM is merged and calculated as a combined (aggregated) fuzzy decision matrix.

The aggregated fuzzy decision matrix is given in Table 4.6. Where, *N* is the total number of decision makers (in our case there are four experts), the aggregated fuzzy rating $x\tilde{}_{ij} = (a_{ij}, b_{ij}, c_{ij})$ of i^{th} alternative with regard to j^{th} is calculated as follows:

$$a_{ij} = \min_n\{a_{ij}^n\},\ b_{ij} = \frac{1}{N}\sum_{n=1}^{N} b_{ij}^n,\ c_{ij} = \max_n\{c_{ij}^n\} \qquad \text{(Eq. 4.2)}$$

Step 5. The combined decision matrix is converted into normalized by using cost / benefit criteria.

Since our aim is maximizing benefit and minimizing cost for the criteria, we convert our fuzzy decision matrix into normalized fuzzy decision matrix denoted by *R* by using:

$$\tilde{r_{ij}} = \left(\frac{a_{ij}}{c_j^*}, \frac{b_{ij}}{c_j^*}, \frac{c_{ij}}{c_j^*}\right) \text{ and } c_j^* = \max_i\{c_{ij}\} \text{ } (benefit\ criteria)$$

(Eq. 4.3)

$$\tilde{r_{ij}} = \left(\frac{a_j^-}{c_{ij}}, \frac{a_j^-}{b_{ij}}, \frac{a_j^-}{a_{ij}}\right) \text{ and } c_j^- = \min_i\{a_{ij}\} \text{ } (cost\ criteria)$$

In our scenario, Compexity (C6) is the cost criterion while the rest is benefit criteria.

Step 6. Compute the weighted normalized fuzzy decision matrix.

The weighted normalized fuzzy decision matrix is

$$\tilde{V} = (\tilde{v}_{ij}), \text{ where } \tilde{v}_{ij} = \tilde{r}_{ij} \times w_j \quad \text{(Eq. 4.4)}$$

Step 7. Compute the Fuzzy Positive Ideal Solution (FPIS) and Fuzzy Negative Ideal Solution (FNIS).

Since the best alternative is calculated by selecting the one that is that is nearest to the FPIS and farthest from the FNIS.

$$A^* = (\tilde{v}_1^*, \tilde{v}_2^*, \dots, \tilde{v}_n^*), \text{ where } \tilde{v}_j^* = \max_i \{v_{ij3}\}$$
$$A^- = (\tilde{v}_1^-, \tilde{v}_2^-, \dots, \tilde{v}_n^-), \text{ where } \tilde{v}_j^- = \min_i \{v_{ij1}\} \quad \text{(Eq. 4.5)}$$

Step 8. Compute the distance from each alternative to the FPIS and to the FNIS by using Euclidian distance.

$$d_i^* = \sum_{j=1}^n d(\tilde{v}_{ij,}\tilde{v}_j^*), \; d_i^- = \sum_{j=1}^n d(\tilde{v}_{ij,}\tilde{v}_j^-) \quad \text{(Eq. 4.6)}$$

Step 9. Compute the closeness coefficient (CC) for each alternative.

$$CC_i = \frac{d_i^-}{d_i^- + d_i^*} \quad \text{(Eq. 4.7)}$$

Step 10. Rank the alternatives based on their CC.

The best alternative with highest closeness coefficient is selected.

4.6.3. Decision Analysis Tool: MCDA for ESDP

A web-based tool was implemented using Java, Angular and Spring Boot Framework for utilizing the application of the two-phase decision analysis. In Figure 4.7 and Figure 4.8, screenshots of the Phase-1 and Phase-2 application are given for the scenario of Case Study 1A, respectively.

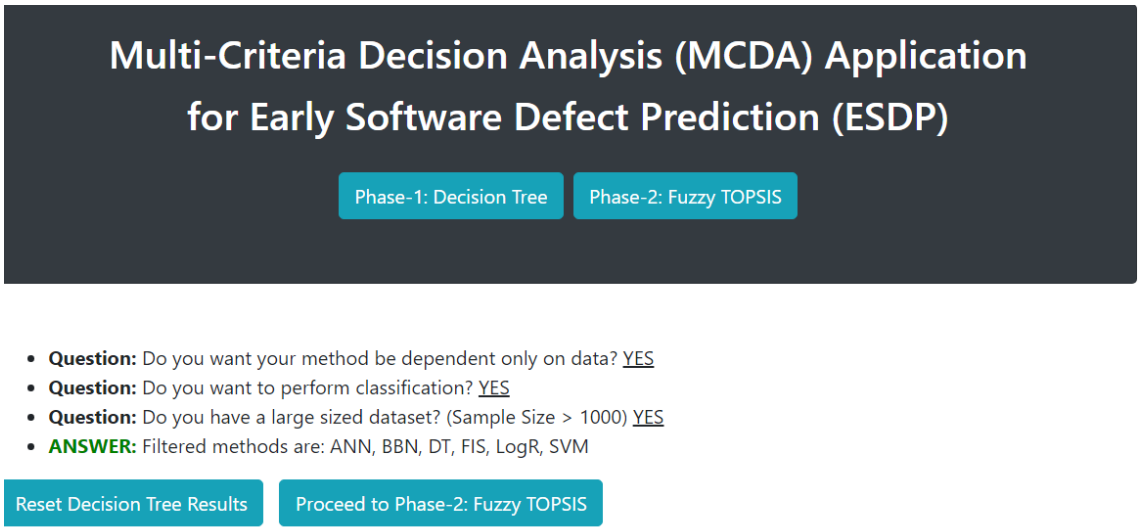


Figure 4.7. Screenshot of Phase-1: Decision Tree Analysis for case study 1A

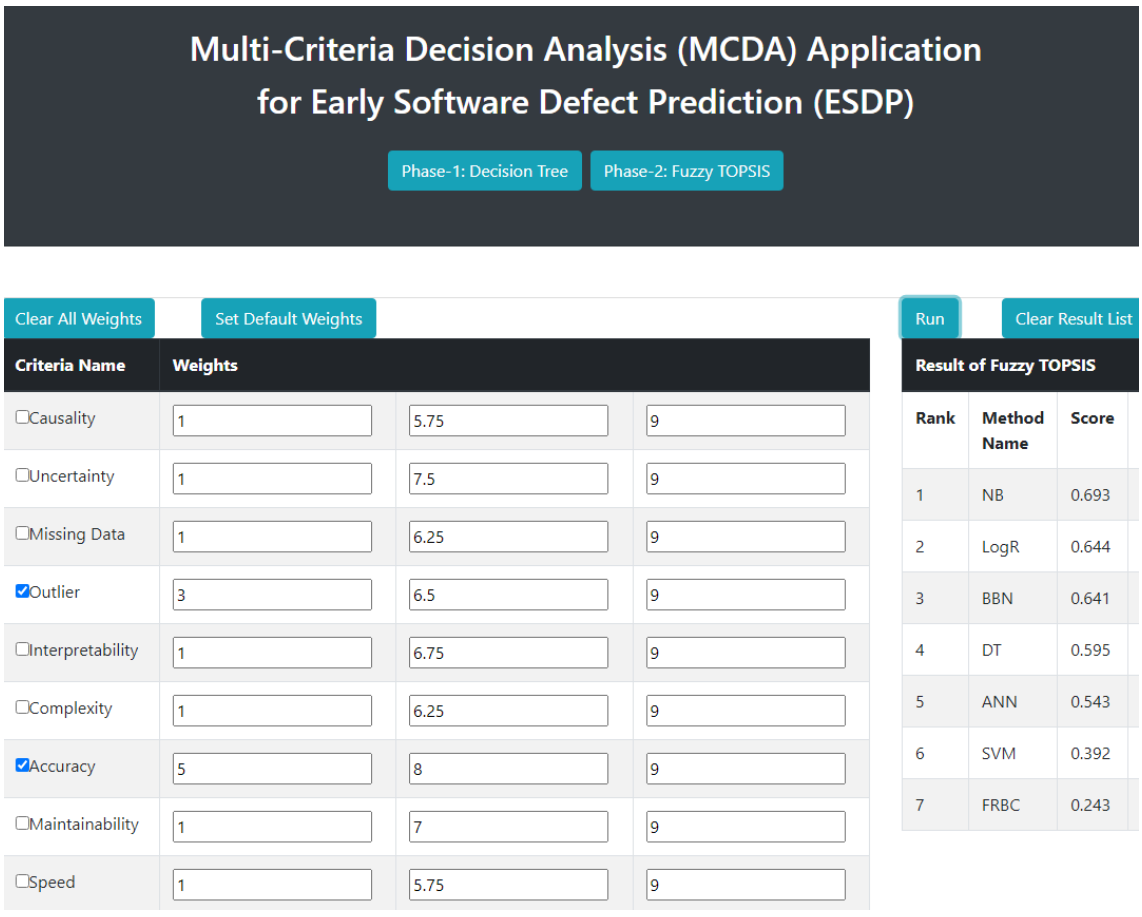


Figure 4.8. Screenshot of Phase-2: Fuzzy TOPSIS Application for case study 1A

5. CASE STUDY

"RQ 5. How should we investigate the trustworthiness of the proposed SDP method selection approach through case studies?". In order to answer this RQ, five case studies in three different contexts were conducted.

5.1. Design of the Multiple Case Study

An embedded multi-case design proposed by Yin [26] was applied for defining the context, case and units of analysis. For Case-1A, 2A and 3, it was aimed to observe that if the ranking recommended by the decision analysis considering the performance criteria was compatible with the ranking obtained from the performance values of the experimental results. For Case-1B and 2B, the aim was to observe that if the ranking recommended by the decision analysis considering the speediness criteria was compatible with the ranking obtained using the time measures of the experimental results. The rest of the cases were also constructed regarding performance or speed criteria. Figure 5.1 demonstrates our multiple case study design, which includes NASA dataset [117], NASA-93 [60] dataset and Fenton dataset [20] as the contexts.

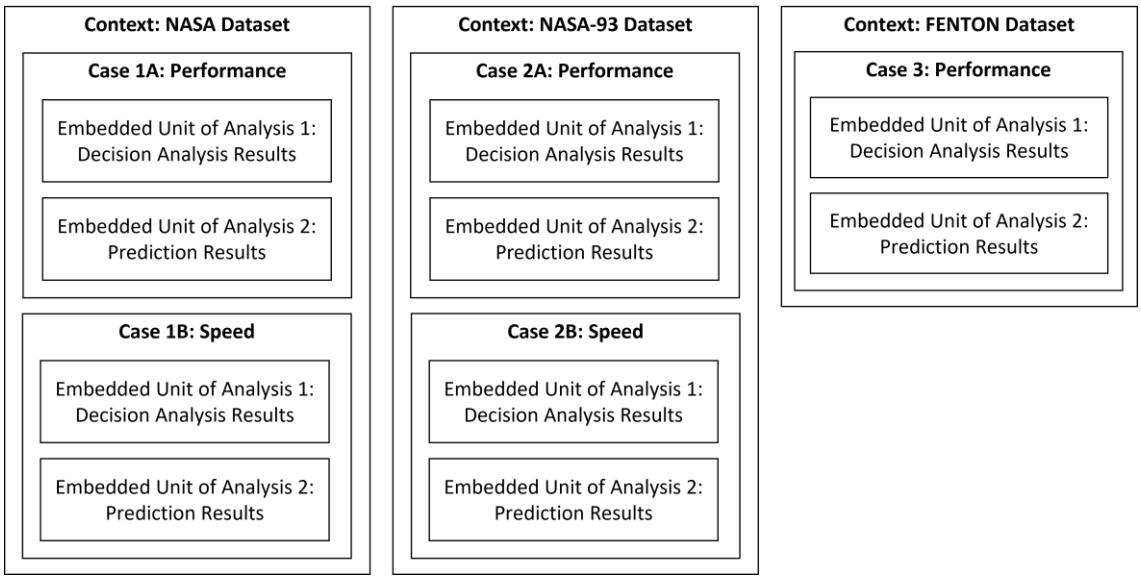


Figure 5.1. Multiple case study design

The rest of this chapter is structed as follows. In Chapter 5.2, research questions of the case study are given. The RQs are answered for NASA, NASA-93, and Fenton datasets in Chapter 5.3, 5.4, and 5.5, respectively. For those sections, first, case study design is given for the related context. Second, the application of the decision analysis process based on the questionnaire and results are given. The questionnaire filled based on the specifications of each dataset and its context information, where “1” denotes “Yes” and “0” indicates “No” for each case study. Third, the details on the experimental study and the prediction results are covered. Next, based on the results of decision analysis and experiments, we analyze whether there is any similarity between the ranking recommended by the decision analysis and the ranking obtained from the actual prediction results. Lastly, we share our observations on the related case study results.

5.2. Research Questions

The primary goal of the case study was to investigate the trustworthiness of the decision analysis approach by answering the following research questions:

- RQ5.1: Which SDP methods are primarily suggested by decision analysis approach?
- RQ5.2: Which SDP methods do perform better in execution?
- RQ5.3: Are there any difference between the results of RQ5.1 and RQ5.2?

Generally speaking, the main goal is to see the similarities between the ranking of the SDP methods recommended by the decision analysis approach and the ranking of the methods with regard to the performance and time measurements obtained from the cases after predictions on the datasets. Below, the motivation for defining the research questions for each case study are discussed.

Motivation for Case Study 1A & Case Study 2A & Case Study 3. In order to verify the effect of "performance" criterion on the ranking of the alternative methods in decision analysis, it is necessary to understand how the classifiers and predictors are ranked according to their actual performance values on the dataset.

Motivation for Case Study 1B & Case Study 2B. In order to verify the effect of "speed" criterion on the ranking of the alternative methods in decision analysis, it is necessary to observe the actual execution times of the classifiers and predictors, and how they are ranked according to those durations.

RQ5.1 is answered by applying the decision analysis process on datasets, while RQ5.2 is answered by executing the classifications and predictions. Afterwards, the results for each case are evaluated by comparing the decision analysis results and prediction results for RQ5.3. Cluster analysis methodology [118] and Friedman analysis with Nemenyi post-hoc test [119] were applied where applicable. To ensure transparency, we have made the resulted values for the experiments available online at [120].

5.3. Case Study 1 - Classification Based on Design Phase Data

5.3.1. Case Study Design

Case study 1A and 1B were performed in the context of the most known public repository for SDP, NASA Metrics Data Program (MDP). We preferred the cleaned version of the dataset [117], which excludes duplicated and inconsistent samples.

As given in Table 2.1, NASA projects are useful for making a classification of the defective classes as it contains ‘defectiveness’ as the dependent variable. Each project has various features ranging from 22 to 40. The independent variables that are used as inputs are all numeric values. Each dataset has different number of samples, which causes a variation in the size of the datasets. As shown in Table 2.1, CM1 project falls into the “Small” category for the dataset size, where PC1 is defined as “Medium”, and JM1 is defined as “Large” dataset [81,94,107]. As for data quality criteria, it can be said that there is not any dependency between attributes, nor uncertainty in the data. Besides, there are no missing values. However, there are outliers in the dataset.

Four features were selected to use within the scope of the case study. These metrics are suitable for the early-phase defect prediction problem because they can be gathered

during the design phase. McCabe metrics are method level metrics and also the most used metrics in SDP area [85]. The most important feature that separates design metrics from code metrics is the opportunity to extract them from design phase diagrams such as UML [121]. These metrics are summarized below [122]:

- Cyclomatic Complexity (v (G)) measures the complexity of the decision structure of a module. It is expressed as the number of linearly independent paths which is actually the minimum number of paths to be tested. “v(G)” is calculated by "v(G) = e - n + 2", where "G" is a program's flowgraph, "e" is the number of arcs in the flowgraph, and "n" is the number of nodes in the flowgraph.
- Module Design Complexity (iv (G)) measures the complexity of the module with reduced design and immediately reflects the complexity of the module's calling patterns to its submodules. Thus, it distinguishes between modules that would seriously complicate the design of any program to which they belong, and modules with complex computational logic.
- Essential Complexity (ev (G)) is a measure that expresses the degree of structuredness and the quality of the code by measuring the degree to which a module contains unstructured code pieces. "ev(G)" is calculated using "ev(G) = v(G) - m" where “m” is the number of subflowgraphs (of "G") that are D-structured primes, in other words “proper one-entry one-exit subflowgraphs”.
- Lines of code (LOC) is measured according to McCabe’s line counting conventions. This metric can be estimated in the early phase of projects by various methods.

It is stated that there is no need to find the best software metrics group for SDP because the performance variation of models using different metrics is not significant [123]. Therefore, to benchmark the performance of the different classifiers, this metric group from design-phase would be sufficient enough.

5.3.2. Decision Analysis (RQ5.1)

5.3.2.1. Gathering case study requirements through questionnaire

The questionnaire was filled based on the specifictions of this dataset and its context information as seen in Table 5.1. We can perform classification since the independent variable is categorical. As there is enough sample point for this dataset, the different dataset sizes would be evaluated in the decision analysis. The performance and speed criteria matter for building an SDP model in this context to compare them.

Table 5.1. Questionnaire filled for case study 1

Question	**Case #1A**	**Case#1B**
Do you want your method to be dependent on data?	1	1
Do you want to address human judgement?	0	0
Do you want to perform classification?	1	1
Do you want to make a numeric prediction?	0	0
Do you have a large sized dataset to train an SDP model?	1[a]	1[a]
Do you have a medium sized dataset to train an SDP model?	1[b]	1[b]
Do you have a small sized dataset to train an SDP model?	1[c]	1[c]
Is there any dependency between data attributes? If yes, do you want to address it?	0	0
Is there any uncertainty in the data? If yes, do you want to address the uncertainty?	0	0
Is there any missing point in the data? If yes, do you want to handle the missing data?	0	0
Is there any outlier in the data? If yes, do you want to handle these outliers?	1	1
Is it important that SDP method has high interpretability?	0	0
Is it important that SDP method has low complexity?	0	0
Is it important that SDP method has high performance?	1	0
Is it important that SDP method has high maintainability?	0	0
Is it important that SDP method has high speed?	0	1

a. Sample size >= 1000 has been considered as Large-sized dataset
b. Sample size >= 500 and < 1000 has been considered as Medium-sized dataset
c. Sample size < 500 has been considered as Small-sized dataset

5.3.2.2. Phase-1: Decision Tree Analysis

In the first phase of the decision analysis process, the subset of alternatives recommended by the decision tree applied according to the answers of questionnaire can be seen in

Figure 5.2. The execution of the decision tree traversal can be summarized with the node numbers: #1, #2 and #3.

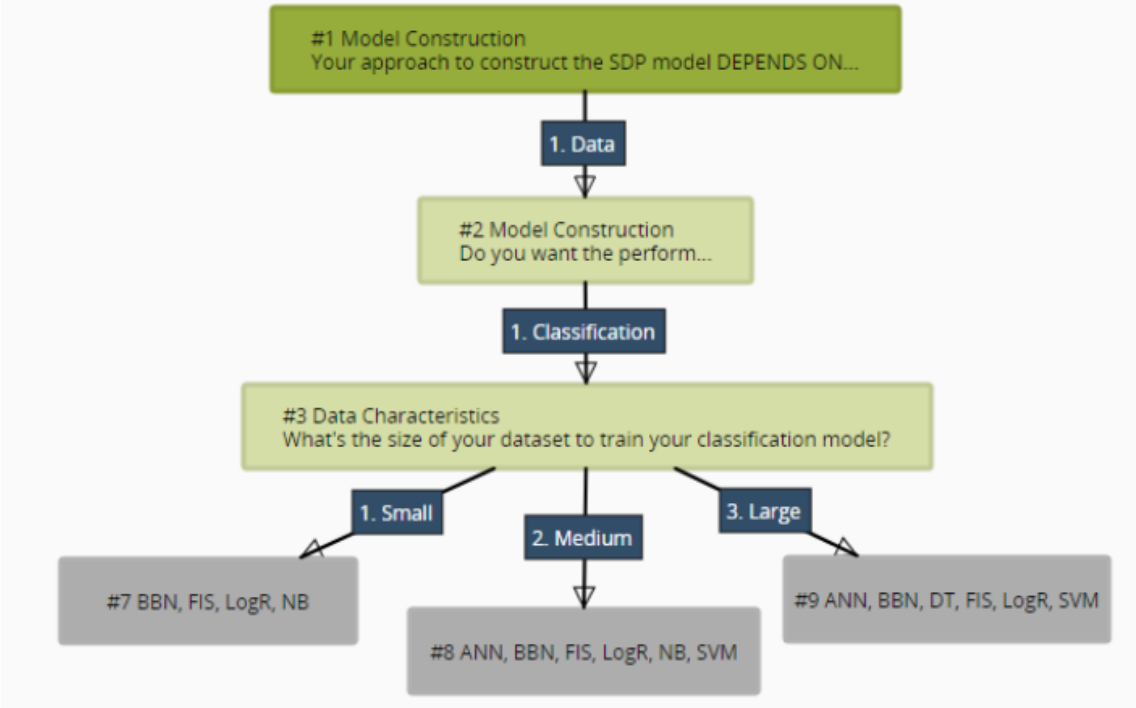


Figure 5.2. Decision tree analysis for case study 1

5.3.2.3. Phase-2: Fuzzy TOPSIS Analysis

The results related to the execution of the Fuzzy TOPSIS application for both cases are given in Table 5.2. It is important to note that, size of the dataset information was not after the decision tree phase, as we aim to make a comparison for all methods that are suitable for classification, regardless of dataset size. In other words, all the methods proposed by decision tree in node #3 (see Figure 5.2) were evaluated within this scope and considered for Fuzzy TOPSIS application.

As seen in the table, the decision analysis process was resulted with the selection of Naive Bayes (NB) method as the best SDP method for both Case Study 1A and Case Study 1B, with a score of 0.693 and 0.702, respectively. On the other hand, Fuzzy Rule Based Classifier (FRBC) was ranked last for both cases, with a score of 0.243 and 0.289, respectively.

Table 5.2. The score and rankings of the methods recommended by Fuzzy TOPSIS

	Case Study 1A		Case Study 1B	
Rank	**Method**	**Score**	**Method**	**Score**
1	NB	0.693	NB	0.702
2	LogR	0.644	BBN	0.636
3	BBN	0.641	LogR	0.627
4	DT	0.595	DT	0.613
5	ANN	0.543	SVM	0.393
6	SVM	0.392	ANN	0.377
7	FRBC	0.243	FRBC	0.289

5.3.3. Experimental Study (RQ5.2)

Within the scope of the case study, different ML-based models were constructed and applied on NASA dataset by using WEKA tool with the version of 3.8.5 [124].

Each classifier in Weka tool was used with their default values, in other words, no optimization has been made for any of the classifiers. The main reason for that is to be able to compare the performances of the classifiers as they are. Besides, no preprocess or cleaning operation has been performed on the data for the sake of consistency among datasets. The classifiers used for the experiments are summarized below:

- Artificial Neural Network (ANN): weka.classifiers.functions.MultilayerPerceptron
- Bayesian Belief Network (BBN): weka.classifiers.bayes.BayesNet
- Decision Tree (DT): weka.classifiers.trees.REPTree
- Fuzzy Rule Based (FRBC): weka.classifiers.rules.MultiObjectiveEvolutionaryFuzzyClassifier
- Logistic Regressin (LogR): weka.classifiers.functions.SimpleLogistic
- Naïve Bayes (NB): weka.classifiers.bayes.NaiveBayes
- Support Vector Machines (SVM): weka.classifiers. functions.LibLINEAR

A 10-fold cross-validation approach was adopted in the training and testing stages of the classifiers. Cross-validation operations were run 10 times for different random segments, resulting in a total of 100 iterations.

The resulted performance values of the classifiers are reported in terms of AUC in Table 5.3, where the best AUC value of the classifiers is given in bold for each dataset. It can be said that the most successful method was Logistic Regression (LogR) for all types of datasets, where Fuzzy Rule-Based Classifier was the worst in terms of AUC.

Table 5.3. Resulting AUC values of the classifiers

	LogR	ANN	NB	BBN	DT	SVM	FRBC
CM1	**0.699**	0.671	0.646	0.563	0.513	0.540	0.494
JM1	**0.692**	0.691	0.602	0.675	0.658	0.546	0.516
KC1	0.671	**0.674**	0.658	0.652	0.637	0.534	0.541
KC3	0.639	0.550	**0.651**	0.489	0.517	0.510	0.547
MC1	**0.731**	0.727	0.699	0.663	0.532	0.511	0.500
MC2	0.664	0.631	**0.677**	0.622	0.601	0.561	0.587
MW1	**0.780**	0.768	0.747	0.734	0.522	0.537	0.533
PC1	**0.815**	0.808	0.601	0.720	0.599	0.548	0.523
PC2	0.773	**0.778**	0.676	0.478	0.510	0.519	0.502
PC3	**0.728**	0.726	0.652	0.714	0.635	0.529	0.500
PC4	0.758	**0.794**	0.702	0.671	0.651	0.551	0.513
PC5	0.720	0.719	0.684	**0.738**	0.704	0.589	0.594
Avg	**0.723**	0.711	0.666	0.643	0.590	0.540	0.529
Rank	1	2	3	4	5	6	7

Below the performance values in terms of AUC and the training time of the classifiers are given with regard to the size of the datasets, in Figure 5.3 and Figure 5.4, respectively.

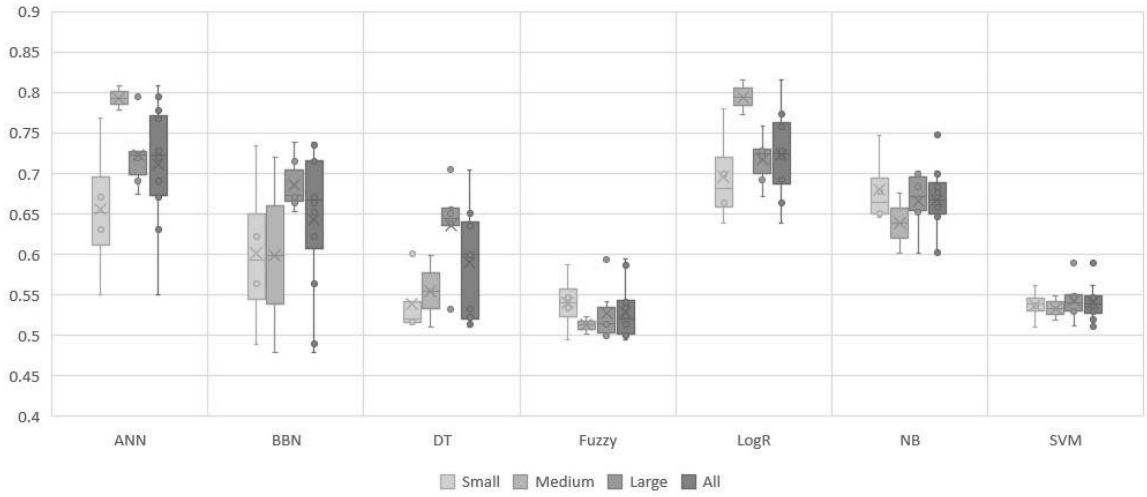
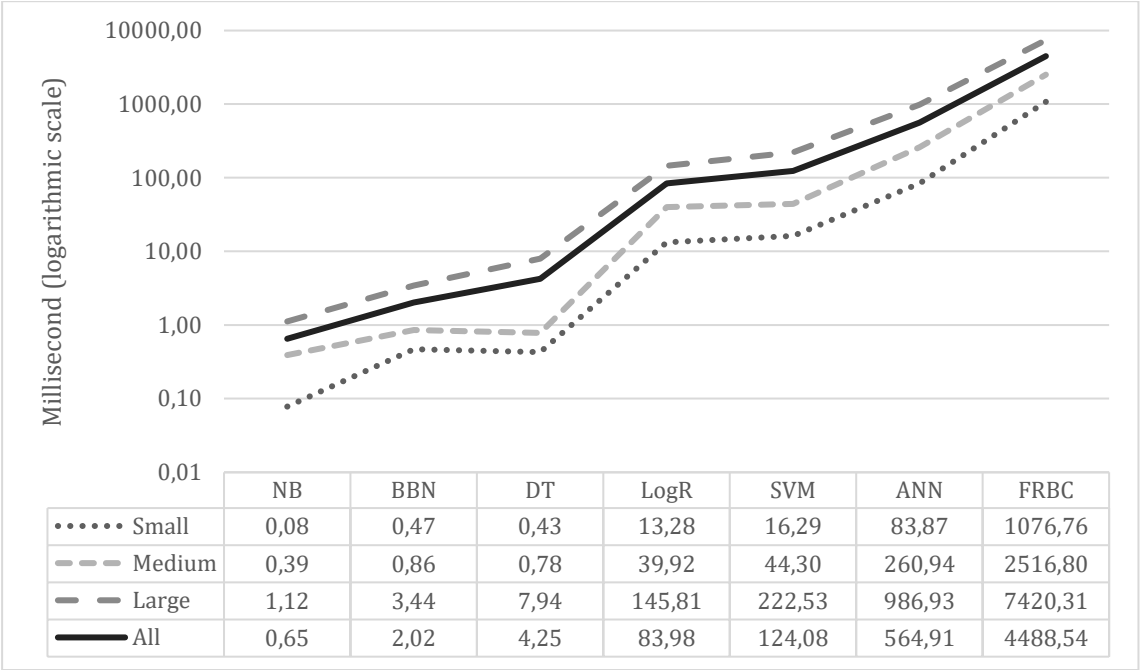


Figure 5.3. AUC values of the classifiers with regard to dataset sizes



	NB	BBN	DT	LogR	SVM	ANN	FRBC
Small	0,08	0,47	0,43	13,28	16,29	83,87	1076,76
Medium	0,39	0,86	0,78	39,92	44,30	260,94	2516,80
Large	1,12	3,44	7,94	145,81	222,53	986,93	7420,31
All	0,65	2,02	4,25	83,98	124,08	564,91	4488,54

Figure 5.4. Average training time of the classifiers with regard to dataset size

5.3.4. Results Comparison (RQ5.3)

5.3.4.1. Case Study 1A

Cluster Analysis

Approach. Firstly, the decision analysis process was executed by setting only the "performance" criterion in order to evaluate the performance of each method. To do that,

the default values were used for the criteria other than "performance", and the decision analysis was executed five times for each fuzzy weight for the "performance" criterion, which are VL, L, M, H and VH, respectively. Based on the resulted rankings, a statistical clustering on Minitab was performed for all the methods. Secondly, AUC values from our experiment results were gathered for all projects in NASA dataset and a statistical clustering on Minitab was performed for all the methods. Lastly, the two rankings were analyzed if they show any similarity.

Results. According to the cluster analysis on Minitab, resulted clusters for decision analysis ranking and experimental performance results (with respect to AUC values) are given in Figure 5.5.a and Figure 5.5.b, respectively. It can be seen that, the decision analysis approach recommended a ranking grouped as follows: (1) BBN, LogR, NB, (2) ANN, DT, and (3) FRBC, SVM; where the performance values of the experiments resulted in a grouping like: (1) ANN, LogR, NB, (2) BBN, DT, and (3) FRBC, SVM. When we look at the clusters, it is seen that the recommendation presented by the decision analysis approach in consideration of the “performance” criterion is in line with the ranking of the AUC values, except for BBN and ANN. Hence, it can be concluded that the decision analysis approach presented a reasonable recommendation based on the characteristics of the NASA dataset.

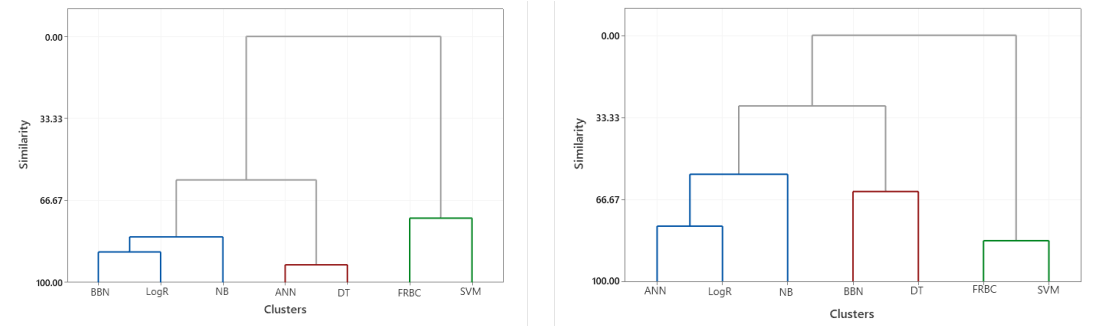


Figure 5.5.a Cluster Analysis of DA-Performance (left), b. Cluster Analysis of Prediction Performance (right)

Friedman analysis with Nemenyi post-hoc test

Approach. In order to see if there were statistically significant differences in prediction performances of the classifiers, a non-parametric Friedman test was carried out [125]. It is used to test for differences between groups when the dependent variable is ordinal or continuous, and samples do not need to be normally distributed. It is recommended for the evaluation of multiple classifiers' prediction performance or computation times [126]. It is also known as the best alternative to the one-way ANOVA with repeated measures. It is important to note that the Friedman test shows whether there are overall differences among the groups but does not specify which groups differ from each other. To do this, we need to run post hoc tests, such as Nemenyi, Wilcoxon signed rank and Dunn test [127]. Nemenyi test is a post-hoc test intended to find the groups of data that differ after Friedman test has rejected the null hypothesis that the performance of the comparisons on the groups of data is similar. The test makes pairwise comparisons using Nemenyi-Wilcoxon-Wilcox all-pairs test for a two-way balanced complete block design. We applied the Nemenyi post-hoc test on RStudio by using PMCMRplus package [119].

Results. We can see that there is an overall statistically significant difference in prediction performance based on the AUC values of the classifiers in Figure 5.6 ($\chi 2(6) = 54.607$, $p < 0.05$).

Ranks

	Mean Rank
ANN	6.00
BBN	4.00
DT	2.92
FRBC	1.67
LogR	6.50
NB	4.83
SVM	2.08

Test Statistics[a]

N	12
Chi-Square	54.607
df	6
Asymp. Sig.	.000

a. Friedman Test

Figure 5.6. Friedman test results for prediction performance (based on AUC)

Based on the Nemenyi test results, there is statistically significant difference in several methods' prediction performances as given in Table 5.4. As we already know the rankings

of the classifiers based on their prediction performance results from Figure 5.6, we can say that ANN performs better than DT, FRBC and SVM, LogR performs better than DT, FRBC and SVM, and NB performs better than FRBC and SVM ($p < 0.05$). Among these results, only the comparison between ANN and DT is different from our decision analysis results.

Table 5.4. Friedman with Nemenyi post-hoc test results for classifier performances

	ANN	**BBN**	**DT**	**FRBC**	**LogR**	**NB**
BBN	0.25975	-	-	-	-	-
DT	0.00856	0.8833	-	-	-	-
FRBC	1.80E-05	0.11252	0.79266	-	-	-
LogR	0.99769	0.06877	0.00095	8.90E-07	-	-
NB	0.84145	0.9652	0.31007	0.00609	0.48706	-
SVM	0.00018	0.31007	0.9652	0.99918	1.10E-05	0.0301

5.3.4.2. Case Study 1B

Cluster Analysis

Approach. Firstly, the decision analysis process was performed by setting only the "speed" criterion to evaluate the speediness of each method. To do this, the default values for the criteria other than "speed" were used. The decision analysis was executed five times for each fuzzy weight (i.e. VL, L, M, H and VH). Based on the rankings obtained, a statistical clustering on Minitab was performed for all the methods. Secondly, "UserCPU_Time_millis_training" values were collected from WEKA for all projects in NASA dataset and a statistical clustering was performed on Minitab for all the methods. Finally, the two rankings were analyzed whether they show any similarity.

Results. Based on the cluster analysis on Minitab, result sets for both decision analysis ranking and experimental time to train the model are given in Figure 5.7.a and Figure 5.7.b, respectively. It can be seen that, the decision analysis approach recommended a ranking grouped as follows: (1) BBN, LogR, DT, NB, (2) ANN, SVM and (3) FRBC; where the training time values of the experiments resulted in a grouping like: (1) NB,

BBN, DT, LogR, SVM, (2) ANN, and (3) FRBC. It can be seen that the recommendation presented by the decision analysis approach, taking into account the "speed" criterion, is somewhat compatible with the ranking of the time values, except for SVM. Hence, it can be concluded that the decision analysis approach presented a reasonable recommendation.

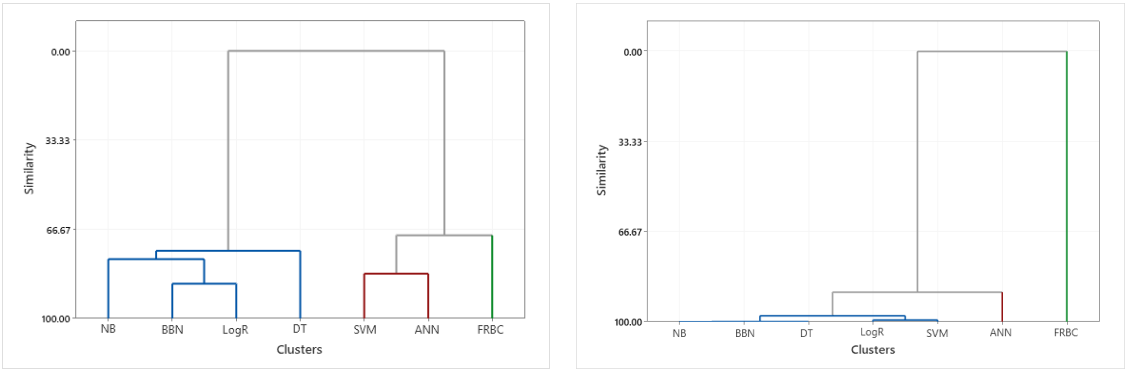


Figure 5.7.a Cluster Analysis of DA-Speed (left), b. Cluster Analysis for Training Time of the Classifiers (right)

Friedman analysis with Nemenyi post-hoc test

We applied the same statistical analysis that is given for Case Study 1A in Section 5.3.4.1. According to the results of the Friedman test given in Figure 5.8, it can be seen that there is an overall statistically significant difference in speediness of the classifiers based on the training time values ($\chi 2(6) = 70.388$, $p < 0.05$). It is important to note that while the training time of a classifier increases, its rank is expected to decrease as there is a opposite relationship between training time and ranking based on speediness.

Ranks

	Mean Rank
NB	1.08
BBN	2.33
DT	2.58
LogR	4.00
SVM	5.00
ANN	6.00
FRBC	7.00

Test Statistics[a]

N	12
Chi-Square	70.388
df	6
Asymp. Sig.	0.000

a. Friedman Test

Figure 5.8. Friedman test results for speed criterion (based on time to train)

Based on the Nemenyi test results, there is statistically significant difference in several methods' training times. According to Table 5.5, it can be said that NB performs faster than LogR, SVM, ANN and FRBC, BBN performs faster than SVM, ANN and FRBC, DT performs faster than ANN and FRBC, and LogR performs faster than FRBC ($p < 0.05$). These results are fully consistent with the rankings suggested by the decision analysis.

Table 5.5. Friedman with Nemenyi post-hoc test results for classifier performances

	NB	**BBN**	**DT**	**LogR**	**SVM**	**ANN**
BBN	0.79266	-	-	-	-	-
DT	0.61549	0.99996	-	-	-	-
LogR	0.01641	0.48706	0.67827	-	-	-
SVM	0. 00018	0.04010	0.08849	0.91777	-	-
ANN	5.2e-07	0.00064	0. 00206	0. 25975	0. 91777	-
FRBC	4.1e-10	2.5e-06	1.1e-05	0.01192	0.25975	0. 91777

5.3.5. Observations

Observations for the classifiers used in the case study are summarized below:

- Classifiers generally have given higher performance results as the size of the dataset has increased.
- LogR method has had the best results for most of the datasets.
    - It is the most successful classifier in all datasets except PC3 and PC4 (large-sized datasets), while it ranks second on decision analysis.
    - The difference in the AUC values is statistically significant for more than one dataset (for JM1, PC1, PC2; with $p = 0.05$)
    - It also has an average speed among other methods as seen in decision analysis, and its training time is increasing excessively when dataset size increases.

- NB classifiers have performed better on small datasets.
    - It ranks second in small size datasets, while decision analysis suggested it to be ranked first.
    - As the size of the dataset increases, the performance of the NB decreases.
    - It has been found to train very fast, as suggested by decision analysis.
- BBN method has generally yielded good results as expected by decision analysis.
    - It performs better than SVM and DT, worse than LogR and NB, as recommended by decision analysis.
    - It is also very fast like NB, especially for small- and medium-sized datasets, as suggested by decision analysis.
- ANN method has the best AUC values for three datasets (mostly large ones) and ranks second in average, while it ranked in 5th place in the decision analysis.
    - For JM1, the largest dataset, it has performed better than all the methods except LogR (and it is statistically significant).
    - For the large datasets, unlike the recommendation of the decision analysis, it can be said that ANN might be preferred in scenarios where other criteria are not important since it shows high performance.
    - On the other hand, the training time has been much longer for ANN than other classifiers. This situation may cause that although the performance of the ANN is high, it might not be preferred for use in practice.
- DT method has been ranked in fourth place according to the decision analysis, it has not shown the expected performance in the prediction results and has been ranked the 5th.
    - It may be due to the selected implementation of DT.
    - Random Forests may result in higher performances than DTs (since it is an ensemble method, we have not included Random Forests in our decision analysis and predictions).
    - It is also very fast like NB and BBN as it is recommended by decision analysis, however, its training time increases for large-sized datasets.

- When looking at the SVM classifier results, decision analysis and experiment results have been in parallel.
    - According to the AUC values, SVM has been the second to the last as recommended in decision analysis.
    - SVM ranks the third from the last in terms of training time, as well as in the decision analysis.
    - Although there are different SVM implementations on Weka, it might give better results if the model is optimized.
- Looking at the FRBC results, it has the worst AUC and time to train values as expected.
    - It is known that the performance of Fuzzy models increases with the preparation of the expert opinion, so when the parameter adjustment is made, higher performance of the FRBC might be obtained.
    - Besides, the training time of FRBC is much longer than other classifiers, and therefore, it might not be preferred in practical use as suggested by decision analysis.

5.3.6. Investigating Evidence from Literature

In the context of our case study on NASA dataset, it would be helpful to investigate the literature in addition to the experiments we conducted, so that we will further solidify the evidence we have.

One of the most important and valuable studies published in the field of SDP is a systematic review study examining 208 experimental studies published between 2000-2010 [8]. The main purpose of the study is to evaluate the effects of software context, SDP techniques and independent variables on the performance of SDP models. The main findings they obtained based on the performance of the models can be summarized as follows:

- Naive Bayes based models generally have the best performance values. In addition, the main reason why Naive Bayes is widely used is that it is a well-understood and simple technique.

- Models using Logistic Regression are also found to perform well in general.
- SVM method does not have good performance values as expected. The reason may be the default Weka settings, which are not optimal for SVM.
- The performance of the models using C4.5 technique (its equivalent in Weka is J48 classifier) is quite average.

In another systematic literature review with important findings [82], 64 studies were examined, the performance of machine learning techniques for SDP was analyzed. In terms of AUC, Random Forest (RF) gave the best performance (AUC = 0.83), ANN, NB and BN models ranked second (AUC = 0.78). Then DT method (C4.5) took place with an AUC value of 0.77, while SVM models showed the worst performance with 0.70 AUC. The accuracy values were found to be between 75% and 85%. This result shows that ML techniques have reasonable prediction ability.

Aside from these secondary studies, there are several primary studies that utilized NASA dataset for SDP.

Catal and Diri [11] investigated the effects of dataset size, metrics set, and feature selection techniques on SDP and conducted experiments on NASA datasets using different ML algorithms, such as Random Forest, Decision Tree, Naïve Bayes. They reported that Naive Bayes is the best prediction method for small sized datasets, while Random Forest classifiers provide better performance on large datasets. They also mentioned the most important selection in SDP is the algorithm and not the metrics suite.

A similar performance comparison study was conducted on the cleaned version of NASA MDP datasets [128]. In this study, the authors concluded that the predictive performance of classifiers is significantly different, and the choice of classifier is important for defect prediction. The overall ranking of the simple classifiers was reported in ascending order as follows: Simple Logistic, Naïve Bayes, Decision Tree, and Support Vector Machines.

A more recent study was conducted on the cleaned NASA dataset [129]. In this study, the authors performed analysis with 10 different classifiers and reported the results with several performance measures. Based on the overall AUC values they reported, the performance of the classifiers can be ordered from best to worst as follows: Naïve Bayes, Artificial Neural Network, Decision Tree, and Support Vector Machines.

In addition, we investigated the literature with a focus on early SDP and analyzed several studies that report comparison of the performance values of different classifiers on NASA dataset, especially by using the requirement or design phase metrics. The performances of several classifiers were synthesized based on reported AUC values. All the data was presented using a box-and-whisker chart demonstrated in Figure 5.9. Overall, the performance data from a total of four papers were extracted as the following: [121,130–132]. The performance values of each classifier were collected and presented in the papers that intersect with the methods we chose (i.e., Decision Tree, Logistic Regression and Naïve Bayes) for all datasets. We grouped the datasets (12 NASA projects) based on their sizes. It is important to note that the notation "<classifier name> (n = <number of datasets>)" was used in the chart to provide the AUC values, where “n” denotes the total number of datasets that are reported in all four papers.

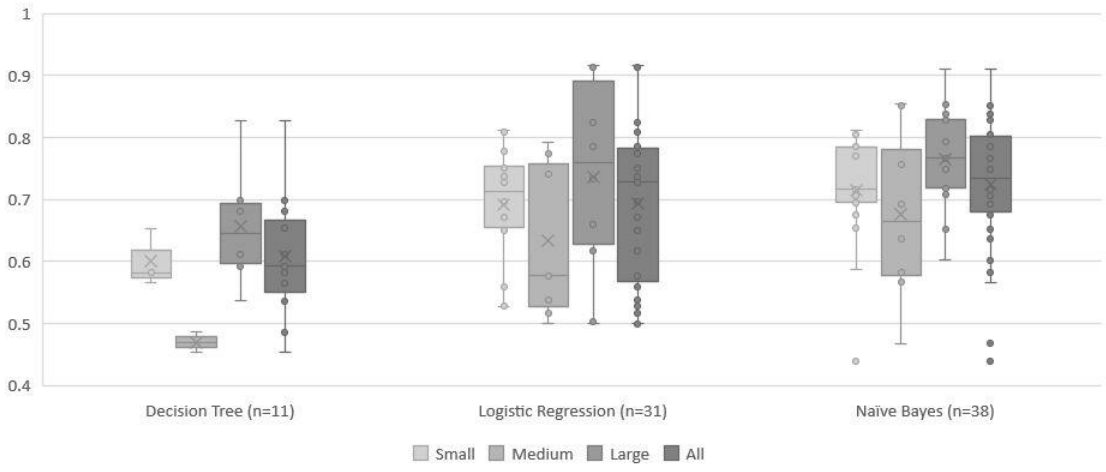


Figure 5.9. AUC values of the classifiers regarding dataset size in the literature

As we can see from the figure, both LogR and NB give good results in all datasets, where NB is slightly better than LogR. However, it is observed that DT is not as successful as

the others. Since performance results of other methods, such as ANN, BBN, SVM and FRBC, were not reported in the relevant publications, we cannot provide an AUC analysis of them.

While LogR, ANN and NB are observed as the best performing methods in our experiments, there are also similar results in the literature accordingly. Likewise, SVM and FRBC were among the worst performing methods in our experiments. Although there are not many results about FRBC in the literature, findings about SVM indicate that it performs poorly. Based on this, we can conclude that the results in the literature are in the same axis with our experimental results.

When we evaluate the validity of the ranking recommendations of decision analysis, we observed that 5 out of 7 methods (excluding ANN and BBN) for performance criteria, and 6 methods (excluding SVM) for speed criteria are compatible with our own experiments and the literature. Besides, the benchmarks in the literature usually examine only the prediction performance of the classifiers, but do not address the speed factor. By reporting the speed results of the classifiers in our second case study, we contributed with a discussion on the training time of the classifiers, which can be an important criterion for selection the suitable SDP method.

Another point that should be emphasized is that we have executed the decision analysis application independent of the dataset size in order to demonstrate the rankings of all methods in the case study. If we had dealt with the dataset size in decision analysis, we would not be able to see the raking of some filtered methods due to the dataset size. Nevertheless, we included our observations on the size of the dataset in Chapter 5.3.5.

5.4. Case Study 2 - Prediction Based on Product, Process and Resource

5.4.1. Case Study Design

Case study 2A and 2B were performed in the context of NASA-93 dataset [60]. After removing the attributes which have repeating values, we had a total of 17 attributes for our case study. The final set can be seen in Figure 5.10.

Attribute	Abbreviation	Type
Process Maturity	pmat {l,n,h}	Nominal
Required software reliability	rely {l,n,h,vh}	Nominal
Database size	data {l,n,h,vh}	Nominal
Product Complexity	cplx {l,n,h,vh,xh}	Nominal
Execution Time Constraint	time {n,h,vh,xh}	Nominal
Main Storage Constraint	stor {n,h,vh,xh}	Nominal
Platform Volatility	pvol {l,n,h}	Nominal
Analysts' capability	acap {n,h,vh}	Nominal
Programmers' capability	pcap {n,h,vh}	Nominal
Application experience	apex {l,n,h,vh}	Nominal
Platform experience	plex {vl,l,n,h}	Nominal
Language and Tool Experience	ltex {vl,l,n,h}	Nominal
Use of Software Tools	tool {n,h}	Nominal
Required Development Schedule	sced {n,l,h}	Nominal
Equivalent physical 1000 lines of source code	kloc	Numeric
Development effort in months	effort	Numeric
Number of defects	defects	Numeric

Figure 5.10. Selected metrics from NASA-93 dataset

5.4.2. Decision Analysis (RQ5.1)

5.4.2.1. Gathering case study requirements through questionnaire

The questionnaire was filled based on the specifictions of NASA-93 dataset and its context information as seen in Table 5.6. We can perform numerical prediction since the independent variable is numerical. As there is only 93 sample points for this dataset, the dataset size can be considered as small. The performance and speed criteria matter for building our SDP model in this context.

Table 5.6. Questionnaire filled for case study 2

Question	**Case #2A**	**Case #2B**
Do you want your method to be dependent on data?	1	1
Do you want to address human judgement?	0	0
Do you want to perform classification?	0	0
Do you want to make a numeric prediction?	1	1
Do you have a large sized dataset to train an SDP model?	0	0
Do you have a medium sized dataset to train an SDP model?	0	0
Do you have a small sized dataset to train an SDP model?	1	1
Is there any dependency between data attributes? If yes, do you want to address it?	0	0
Is there any uncertainty in the data? If yes, do you want to address the uncertainty?	0	0
Is there any missing point in the data? If yes, do you want to handle the missing data?	0	0
Is there any outlier in the data? If yes, do you want to handle these outliers?	0	0
Is it important that SDP method has high interpretability?	0	0
Is it important that SDP method has low complexity?	0	0
Is it important that SDP method has high performance?	1	0
Is it important that SDP method has high maintainability?	0	0
Is it important that SDP method has high speed?	0	1

5.4.2.2. Phase-1: Decision Tree Analysis

In the first phase of the decision analysis process, the subset of alternatives recommended by the decision tree applied according to the answers of questionnaire can be seen in Figure 5.11. The execution of the decision tree traversal can be summarized with the node numbers: #1, #2 and #6.

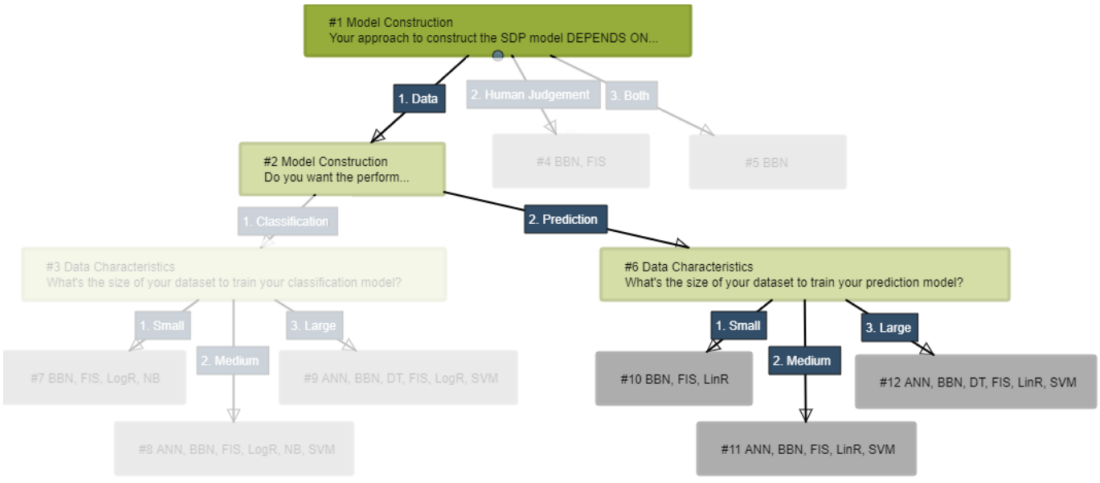


Figure 5.11. Decision tree analysis for case study 2

5.4.2.3. Phase-2: Fuzzy TOPSIS Analysis

For the decision analysis, Fuzzy TOPSIS process was applied according to the answers of the questionnaire in order to make a ranking-based selection among SDP methods. The results related to the execution of the Fuzzy TOPSIS application for both cases are given in Table 5.7. It is important to note that we only included ANN, DT, LinR and SVM methods, as we aim to make a comparison for the methods that are suitable for our empirical design as explained in Section 5.4.3. In other words, suitable methods proposed by decision tree in node #6 (see Figure 5.11) were evaluated within this scope and considered for Fuzzy TOPSIS application.

Table 5.7. The score and rankings of the methods recommended by Fuzzy TOPSIS

	Case Study 2A		**Case Study 2B**	
Rank	**Method**	**Score**	**Method**	**Score**
1	DT	0.690	DT	0.763
2	LinR	0.542	LinR	0.675
3	ANN	0.423	SVM	0.371
4	SVM	0.279	ANN	0.226

As seen in the table, the decision analysis process was resulted with the selection of Decision Tree (DT) method as the best SDP method for both cases, with a score of 0.690 and 0.763, respectively. On the other hand, SVM was ranked last for Case Study 2A, where ANN was the last selected method for Case Study 2B with a score of 0.279 and 0.226, respectively.

5.4.3. Experimental Study (RQ5.2)

Each predictor in Weka tool was used with their default values, in other words, no optimization has been made. The predictors used for the experiments are given below:

- **Artificial Neural Network (ANN):** weka.classifiers.functions.MultilayerPerceptron
- **Decision Tree (DT):** weka.classifiers.trees.M5P
- **Linear Regression (LinR):** weka.classifiers.functions.LinearRegression
- **Support Vector Machines (SVM):** weka.classifiers.functions.SMOreg

A 10-fold cross-validation approach was adopted in the training and testing stages of the predictors. Cross-validation operations were run 10 times for different random segments, resulting in a total of 100 iterations. The resulted performance values of the predictors are reported in terms of R^2, RAE and RRSE in Table 5.8. It can be said that the most successful method was DT, where SVM was the worst in terms of R^2.

Table 5.8. Resulting performance values of the predictors

	DT	LinR	ANN	SVM
R^2	0.965[a]	0.893[b]	0.879	0.859
RAE (%)	25.839[a]	35.048	40.771	40.392
RRSE (%)	33.952[a]	44.491[b]	50.149	52.874
Rank	1	2	3	4

[a] DT performed significantly better than LinR, SVM and ANN (α=0.05)
[b] LinR performed significantly better than SVM (α=0.05)

The training time of the predictors are given in Table 5.9 in terms of "UserCPU_Time_millis_training". Since the training times can be different for each iteration, we analyzed the results for each execution of training and calculated the average values. Based on the Paired T-Test results, there is no significant difference between LinR and DT models, however, they have performed significantly faster than SVM and ANN models ($p < 0.05$). Besides, SVM has performed significantly faster than ANN.

Table 5.9. Training times (millisecond) for each predictor regarding to iterations

#Iteration	**LinR**	**DT**	**SVM**	**ANN**
1	3.13	4.69	153.13	978.13
2	1.56	3.13	148.44	982.81
3	1.56	4.69	153.13	992.19
4	3.13	6.25	165.63	1001.56
5	1.56	6.25	198.44	985.94
6	3.13	3.13	196.88	1010.94
7	0.00	6.25	190.63	1006.25
8	1.56	6.25	178.13	1039.06
9	0.00	6.25	185.94	1104.69
10	3.13	4.69	153.13	1090.63
Avg.	1.88[a]	5.16[a]	172.34[b]	1019.22
Rank	1	2	3	4

[a] LinR and DT results are significantly better than SVM and ANN results (α=0.05)
[b] SVM is significantly better than ANN (α=0.05)

5.4.4. Results Comparison (RQ5.3)

Due to the lack of enough sample data points, we could not perform cluster analysis or Friedman test for Case Study 2A and Case Study 2B. Instead, we reported the rankings of the predictors for both decision analysis process and experimental study results. According to the resulted rankings for decision analysis and experimental results given in Table 5.10, we can see that, the decision analysis approach recommended a ranking as follows: (1) DT, (2) LinR, (3) ANN and (4) SVM; where the performance values of the experiments resulted in the same ranking.

When we evaluate the results in terms of performance criteria, it was seen that the ranking recommended by the decision analysis and the ranking based on the performance obtained from the experiments were the same (DT, LinR, ANN, SVM).

Table 5.10. Decision Analysis and Empirical Results for Case Study 2A

	Case Study 2A		**Case Study 2B**	
Rank	**Decision Analysis**	**Empirical Result**	**Decision Analysis**	**Empirical Result**
1	DT	DT	DT	LinR
2	LinR	LinR	LinR	DT
3	ANN	ANN	SVM	SVM
4	SVM	SVM	ANN	ANN

It can be seen that DT method is quite successful for predicting the numerical dependent variable. It has been observed that this performance is also valid for small datasets (data point < 100) of tree-based learning methods. Therefore, the need arises for an arrangement in the first phase of the decision analysis process, which consists of the DT method as an alternative for small datasets. LinR method was also found to be the most successful method after DT. Although ANN and SVM methods performed above expectations, they were placed last in the ranking. This result confirms the assumption that the relevant methods are not successful enough for small datasets.

When we evaluate in terms of the speed criterion, it was seen that the ranking recommended by the decision analysis (DT, LinR, SVM, ANN) and the ranking based on

the learning durations of the prediction models obtained from the experiments (LinR, DT, SVM, ANN) were not exactly the same, and there was a difference in the order between LinR and DT.

In general, it has been observed that regression models are built faster, while tree models learn slower than regression models. In line with this information, it is thought that it may be necessary to make an adjustment in the structuring of the decision analysis process and increase the score of LinR method on speed criteria.

5.5. Case Study 3 - Lack of Data: Prediction Based on Expert Opinion

5.5.1. Case Study Design

Case study 3 was performed by using Fenton dataset [21], which is suitable for early software defect prediction problem since it includes phase information of software metrics gathered. The context information reported about the public dataset is given in Table 5.11.

Table 5.11. Basic information of Fenton dataset [21]

Business Domain	Consumer electronics	**Size (KLOC)**	Min: 0.9 – Max: 155.2
Programming Language	C, VC++, MFC	**Effort (Hour)**	Min: 1,308 – Max: 53,995
SDLC Methodology	Waterfall life cycle	**Defects (Number)**	Min: 5 – Max: 1,906
Total Project Number	31	**Missing data values**	Yes (32 missing data points out of 930)
Number of features	30	**Outlier values**	Yes

As we can see from Table 5.11, Fenton dataset is useful for making a numerical prediction as it contains 'number of defects' as the dependent variable (output). The dataset has a total of 31 data points (samples).

As shown in Table 5.12, there are metrics with categorical data ranging from very low (VL) to very high (VH) that can be used as input metrics in the dataset. It also has the size information given in thousands of lines of code (KLOC) which can be used as a normalizer metric to predict the number of defects. In addition, since the qualitative data collection process of this dataset was conducted through a questionnaire [21], it is noted that there might be some data quality problems, such as dependency between attributes, uncertainty, which should be considered when evaluating the prediction methods. Besides, it is mentioned that there are missing values and outliers in the dataset, with a total of 32 missing data values and three projects with the number of defects more than 1500, respectively. the first three data points are given as an example in Table 5.12.

Table 5.12. Example data from public dataset [21]

Project ID	KLOC	RFD	RS	RIW	ERT	Number of Defects
1	6	H	L	VH	H	148
2	0.9	H	H	VH	H	31
3	53.9	VH	H	VH	H	209

5.5.2. Decision Analysis (RQ5.1)

5.5.2.1. Gathering case study requirements through questionnaire

The questionnaire was filled based on the specifictions of this dataset and its context information. As there is not enough sample point for this dataset, it can be assumed that there is a lack of data, therefore expert opinion matters for building an SDP model in this context. Besides, the dependent variable is the number of defects, which leads us to generate a numerical prediction model. In terms of data quality of the dataset, as given in Table 5.13, there is uncertainty, missing points, and outliers in the data, as well as dependency between the attributes. Moreover, since it will be an expert opinion-based model, it would be beneficial for SDP model to have the specified method characteristics.

Table 5.13. Questionnaire filled for case study 3

Question	Case Study 3
Do you want your method to be dependent on data?	0
Do you want to address human judgement?	1
Do you want to perform classification?	0
Do you want to make a numeric prediction?	1
Do you have a large sized dataset to train an SDP model?	0
Do you have a medium sized dataset to train an SDP model?	0
Do you have a small sized dataset to train an SDP model?	0
Is there any dependency between data attributes? If yes, do you want to address it?	1
Is there any uncertainty in the data? If yes, do you want to address the uncertainty?	1
Is there any missing point in the data? If yes, do you want to handle the missing data?	1
Is there any outlier in the data? If yes, do you want to handle these outliers?	1
Is it important that SDP method has high interpretability?	1
Is it important that SDP method has low complexity?	1
Is it important that SDP method has high performance?	1
Is it important that SDP method has high maintainability?	1
Is it important that SDP method has high speed?	1

5.5.2.2. Phase-1: Decision Tree Analysis

In the first phase of the decision analysis process, the subset of alternatives recommended by the decision tree applied according to the answers of Questionnaire Phase-1 can be listed as follows: BBN and FIS. The execution of the decision tree traversal can be summarized with the node numbers: #1 and #4, as demonstrated in Figure 5.12.

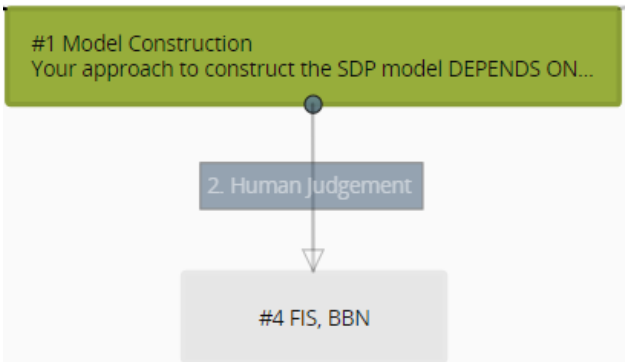


Figure 5.12. Execution of the decision tree for case study 3

Here, the decision tree can be executed with the answer of the first question of the questionnaire, and it was decided that human judgment methods (FIS and BBN) would be used. In other words, decision tree does not require further questions in the questionnaire since the subset of alternatives can be eliminated by the "MC - Approach to construct the model" criteria.

5.5.2.3. Phase-2: Fuzzy TOPSIS Analysis

As for the second phase, Fuzzy TOPSIS process was applied according to the answers of the questionnaire to make a ranking between the two selected methods. The results related to the execution of the Fuzzy TOPSIS application for Case Study 3 is given in Table 5.14.

Table 5.14. The score and rankings of the methods recommended by Fuzzy TOPSIS

Rank	Method	Score
1	FIS	0.505
2	BBN	0.495

As seen in the table, the decision analysis process is resulted with the selection of FIS method as the best SDP method for this case, with a score of 0.505. The second convenient method is identified as BBN with a score of 0.495.

5.5.3. Experimental Study (RQ5.2)

5.5.3.1. Selected software metrics

The most important motivation for choosing the selected input metrics for this case study is the extensive literature review that we performed in our previous work [31], which was mainly focused on the process based metrics used for early software defect prediction. According to the results of our literature review study, the most used process-based metrics are related to the effort of the review activities, stability of the requirements, and the number of defects found from the review activities. Therefore, we considered three process metrics that can be gathered at the requirement phase. We also select a resource related metric, experience of the requirement team, which we found as the most used

resource related metric in the literature [15]. The selected software metrics and their explanation is given below.

Requirement fault density (RFD): This metric measures the ratio between the total number of defects obtained from the requirements analysis phase and the size of the software. The defects can be found during the review activities in the requirement phase. Also, the size of software can be estimated by function point (FP) at the beginning of a software project. Therefore, this metric can be collected during the requirement phase of SDLC. The number of defects to be predicted is assumed to be directly proportional to the value of the RFD.

Requirement Stability (RS): Requirement changes can be at any time during the development of the software project. However, it is better to minimize the changes in requirements in order to reduce the impact of the defects occurred during the addition, deletion or modification of the requirements. This metric describes the stability of the software requirements. Hence, the number of defects to be predicted is assumed to be inversely proportional to the value of the RS.

Review, inspection and walkthrough (RIW): This metric describes the consistency, feasibility and completeness of the artifacts produced during the requirement analysis phase. Software reviews are activities necessary to identify and correct defects during the development life cycle. In addition, it is aimed to produce reliable software on time and budget with regular review, inspection and walkthrough activities. For this reason, the indicator of reviews was included as a critical metric for the early stages and included in the model. As similar to RS, the number of defects to be predicted is assumed to be inversely proportional to the measure of RIW.

Experience of requirement team (ERT): This metric gives the information about the experience, knowledge and skill of the requirement team members in analyzing and generating requirements. We can say that if requirement analysis team consists of experienced people, we can ensure that the artifacts related to the requirement phase are

of higher quality and therefore transfers less defects to the later phases. The number of defects to be predicted is assumed to be inversely proportional to the experience of requirement team.

Aside from these input metrics, we also used the size of the software as a normalizer parameter for calculating the predicted number of defects as the last step of the model implementation. Although, it is not possible to measure the size of software exactly in the early phases of SDLC, there are many methods to estimate the size of the software, such as function points or feature points. In general, it is reported that larger and smaller software may include more errors than medium-sized software. For this reason, size is used in the model as a normalizer metric for predicting the number of defects.

5.5.3.2. Empirical design of FIS based model

FIS based SDP model was implemented by using the MATLAB Fuzzy Logic Toolbox [133]. The architectural design for the model is presented in Figure 5.13. Implementation steps for the proposed model are as follows:

1. Selection of the software metrics that will be used as inputs for the fuzzy model.
2. Determination of membership functions of input and output metrics.
3. Designing fuzzy logic rules.
4. Performing fuzzy inference.
5. Defuzzification and calculation of the crisp values of number of defects.

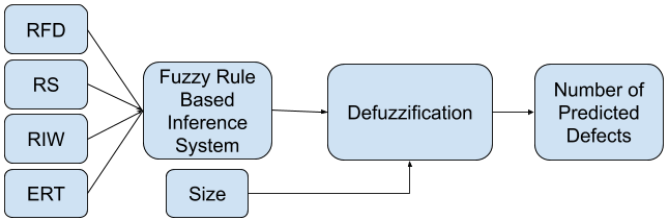


Figure 5.13. The design of the proposed FIS based model

As for the membership functions of input metrics RFD, RS and ERT, logarithmic scale was considered, while for input metric RIW, linear scale was used. As shown in Figure 5.14, the linguistic values of all input metrics were considered as five scale values, which are Very High (VH), High (H), Medium (M), Low (L) and Very Low (VL). As for the output variable, we used seven-scale linear fuzzy profile as given in Figure 5.15, which are Very Very High (VVH), Very High (VH), High (H), Medium (M), Low (L), Very Low (VL) and Very Very Low (VVL). Figure 5.16 shows a portion of the fuzzy rule set which has 625 (5^4) rules in total.

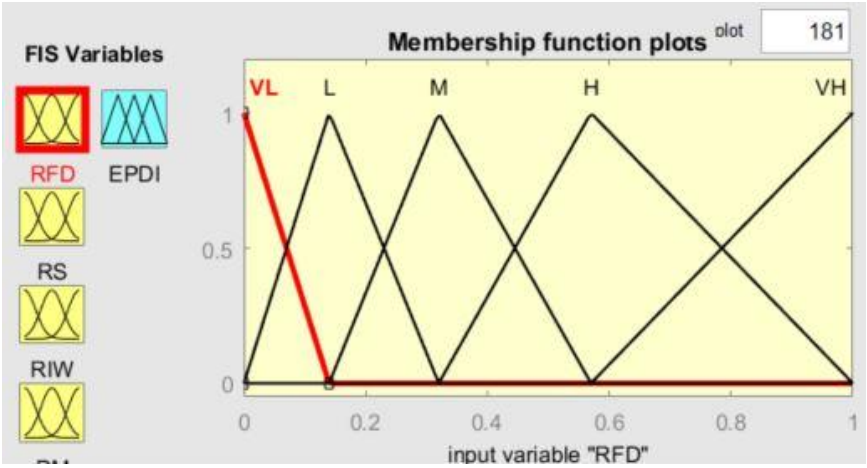


Figure 5.14. Membership function of the input variable ‘RFD’

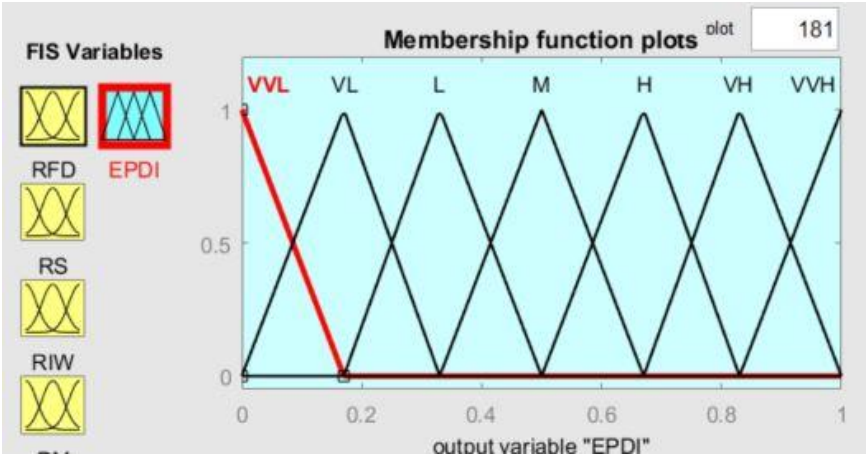


Figure 5.15. Membership function of the output variable

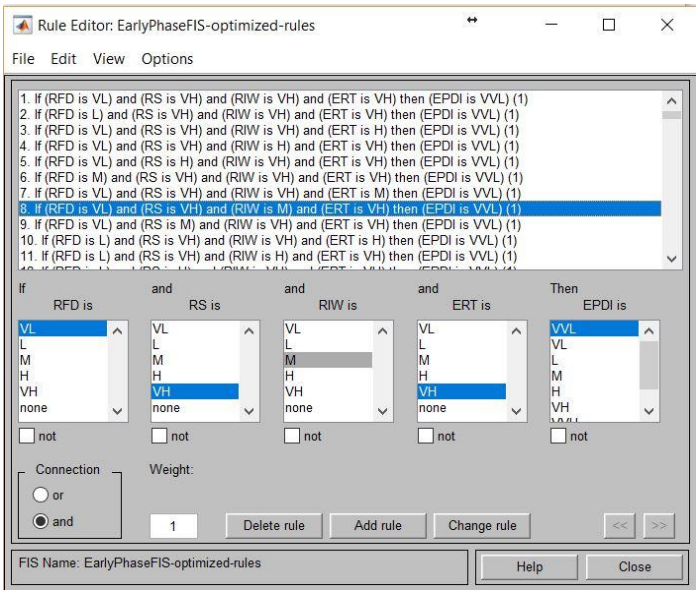


Figure 5.16. A portion of the fuzzy rule set

5.5.3.3. Empirical design of BBN based model

BBN based SDP model was implemented by using the WEKA tool [124]. The architectural design for the model is presented in Figure 5.17. Implementation steps for the proposed model are as follows:

1. Selection of the software metrics that will be used as inputs for the bayes network.
2. Construction of causal relationships (network structure) between selected metrics
3. Determining the probability tables of the nodes in the network
4. Compilation of bayes network
5. Finding the probabilistic values of the predicted number of defects
    a. Entering the qualitative values of the model's inputs (metrics) into the compiled BN
    b. Obtaining categorical (VL, L, M, H, VH) outputs and probabilistic values of the defects
6. Calculation of the number of defects using categorical output, probability values and size information

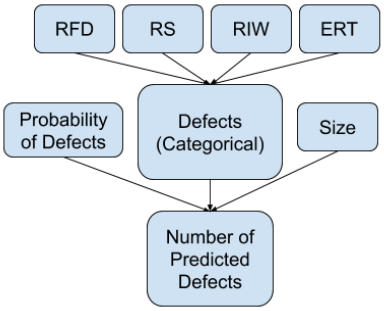


Figure 5.17. The design of the proposed BBN based model

In this context, six different model structures that reflect the expert opinion have been formed. The experiments conducted based on these six different model designs, however, in this paper, the design with the best prediction performance is reported. Figure 5.18 shows the structure of the proposed BBN based model. In this design, ERT influences RIW and RFD metrics, RIW metric is affecting RFD and RS metric is affecting RFD.

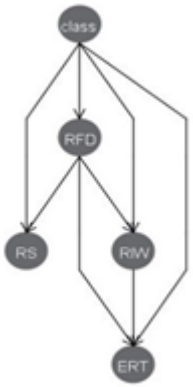


Figure 5.18. The structure of the proposed BBN based model

5.5.3.4. Prediction Results

In Table 5.15, the resulted performance evaluation measures are reported in terms of MMRE, BMMRE and R^2. When we compare the prediction performance results of our proposed FIS based model with the existing models in the literature, we can say that our results are close to the fuzzy rule-based model results presented by D. K. Yadav et al. [114] (R^2 = 0.951) and Chatterjee and Maji [134] (R^2 = 0.971). When we compare the prediction performance results of our proposed BBN based model with the existing models in the literature, we can say that our results are better than the Bayesian Network based prediction model by [21] (R^2 = 0.899).

Table 5.15: Performance results of the proposed models

Model	**MMRE**	**BMMRE**	**R^2**
FIS based model	0.430	0.685	0.926
BBN based model	0.588	1.249	0.913

It can be said that the prediction results are showing a good performance for early software defect prediction.

5.5.4. Results Comparison (RQ5.3)

The ranking based on the prediction results of the experimental study is in line with the ranking recommended by the decision analysis approach, in which fuzzy model was found to be the most convenient method for prediction of the potential software defects in the requirement phase with process and resource-based metrics. While the performance results of BBN based model are also satisfactory, FIS based model is slightly successful in terms of all the performance measures. Therefore, we can confirm that the decisions recommended by the decision analysis process is trustworthy in terms of the performance comparison of the first and second alternative methods in ESDP context.

5.5.5. Investigating Evidence from Literature

Decision analysis results are more significant when the evidence from other studies conducted on the same dataset in the literature were synthesized. Firstly, the primary studies presenting and using the same dataset were listed. After that, the subsequent studies that use this dataset to investigate the suitability of the SDP models built in the early phases of SDLC by using BBN and FIS methods were examined.

BBN based models

Fenton et al. [21] proposed an early life cycle defect prediction model based on the Bayesian network method to predict the number of defects that may be found during independent tests or operational use. The results showed that early prediction models could be used as an effective decision support mechanism in the early stages of

development. They performed performance evaluation with R^2 and various error rates (MMRE, BMMRE etc.). Based on some example scenarios, they also showed how the model can be used for decision support in the operational environment. In addition, they conducted a sensitivity analysis to determine the most effective factors on the number of defects, and reported those factors as the size, complexity, and distributed communication level of the project. It is stated that the model can be used from the early stages of the life cycle since it does not require detailed domain knowledge in the context of the project. Moreover, they have made an important contribution to the literature with the dataset published publicly.

Kumar and Yadav [135] proposed a BBN based defect prediction model constructed with six different metrics (software complexity, requirement stability, experience of teams, review effort, quality of outputs, and rate of new development) that can be obtained from requirements analysis, design and coding phases. The performance evaluation of the model applied on the Fenton dataset was performed with MMRE and BMMRE, and they reported better results than Fenton et al. [21].

Chatterjee and Maji [136] presented a BBN based model to predict the number of defects in the early phase of the software development process. In order to construct the belief network, six requirement and design phase-based metrics were selected. They chose a fault index (FI) as the target node of the BBN. They also used an ANN model to calibrate the final number of predicted defects, by using actual size of the software and FI values as input nodes for ANN. They randomly chose twenty projects to be used for network training, and the remaining six projects were used for simulation. For this reason, they reported the performance evaluation of the proposed model with regard to those selected six projects. According to the specific projects, they reported RMSE, NRMSE, MMRE, BMMRE, and R^2 values better than Fenton et al. [21].

In Table 5.16, we summarized the number of metrics used by the models proposed in these studies, the phase information of the metrics belong to, and reported performance values with the number of projects included in performance evaluation.

Table 5.16. BBN based SDP models and reported performance values for Fenton's dataset presented in the literature

BBN model	Number of metrics	Phase information [a]	Number of projects included in performance evaluation (out of 31)	MMRE	BMMRE	R^2
Fenton et al. [21]	31	R,D,C,T	31	0.960	0.300	0.931
Kumar and Yadav [135]	11	R,D,C	10	0.069	0.075	-
Chatterjee and Maji [136]	6	R,D	6	0.400	0.410	0.930

a. R=Requirement Analysis, D=Design, C=Coding, T=Testing phase

FIS based models

Pandey and Goyal [137] proposed a model based on FIS method that uses metrics for requirements analysis, design, coding, and test phases. The model is structured on a phase-based basis and the output of each phase (predicted number of defects) is used as input in the next phase, and the output from the test phase is constructed to yield the result of the model. In general, it is stated that the first phases of the software life cycle should be handled more carefully than the later phases. They confirmed these models with experiments using Fenton's dataset and reported their performance with MAPE (Mean Absolute Percent Error).

Yadav et al [114] proposed a FIS based model, which was constructed by using three metrics related to software size and requirement analysis phase. Fenton's dataset was used for validation. The predictive performance of the proposed approach was compared with the existing models over the values of MMRE and BMMRE and the model presented was reported to be better than Fenton et al. [21].

Yadav and Yadav [138] proposed a model based on FIS method that uses seven metrics from requirements analysis, design and coding phases. The model is structured similar to

the model proposed by Pandey and Goyal [137]. They reported the results via MMRE and BMMRE, which are better than Fenton et al. [21].

Yadav and Yadav [139] extended their previous work by adding two more metrics of test phase to the FIS based SDP model. They reported the results by MMRE and BMMRE, which are better than Fenton et al. [21]. They also performed a sensitivity analysis and reported that software metrics, which can detect the defects in the early phases of SDLC, need to be considered with more attention than the metrics that become available in the later phases.

Chatterjee and Maji [134] presented a FIS based model that uses four metrics related to the requirement phase. In this study, unlike previous studies, a new algorithm is proposed instead of expert opinion to develop fuzzy rule-based system. It is stated that the weight of each metric as well as the target reliability are taken into consideration in order to develop the proposed fuzzy rule algorithm. As a result, the proposed model showed better performance results than other models.

Kumar and Ranjan [140] proposed a phase based FIS model that uses five metrics from requirements to testing phases. The model is structured similar to the model proposed by Pandey and Goyal [137]. They also reported the results via MMRE and BMMRE, which are better than Fenton et al. [21].

Chatterjee et al. [141] proposed an interval type-2 FIS based SDP model that can be used separately in the requirement analysis, design and coding phases of the software life cycle. In addition to the expert opinion and human reasoning, a new algorithm developed to form a generalized consistent fuzzy if-then rule base for FIS. One of the advantages of the proposed SDP model is that it predicts the number of defects three times during each of the SDLC phases, therefore it can be helpful during the early development process about potential software defects. They presented the performance evaluation of the proposed model by using nine software projects from Fenton's dataset. Based on these

selected projects, they reported RMSE, NRMSE, MMRE, BMMRE, and R^2 values better than Fenton et al. [21].

In Table 5.17, we summarized the number of metrics used by the models proposed in these studies, the phase information of the metrics belong to, and reported performance values with the number of projects included in performance evaluation.

Table 5.17. FIS based SDP models and reported performance values for Fenton's dataset presented in the literature

FIS model	**# of metrics**	**Phase information** [a]	**# of projects included in performance evaluation (out of 31)**	**# of FIS Rules**	**MMRE**	**BMMRE**	**R^2**
Pandey and Goyal [137]	10	R,D,C,T	15	1350	0.226 [b]	0.231 [b]	0.953 [b]
D. K. Yadav et al [114]	3	R	20	27	0.361	0.419	0.951 [b]
Yadav and Yadav [138]	7	R,D,C	20	117	0.069	0.076	-
Yadav and Yadav [139]	9	R,D,C,T	20	162	0.047	0.048	-
Chatterjee and Maji [134]	4	R	20	625	0.286	-	0.971
Kumar and Ranjan [140]	5	R,D,C,T	20	130	0.181	0.188	0.992 [b]
Chatterjee et al. [141]	9	R,D,C	9	125	0.473	0.621	0.936

a. R=Requirement Analysis, D=Design, C=Coding, T=Testing phase
b. Calculated manually by the author

When we investigate the studies using Fenton's dataset, FIS based models seem to be outnumbered. This may be interpreted as the relevant dataset is more suitable for ESDP using FIS based models. Furthermore, when the performance evaluation tables are examined, it can be seen that FIS based models performed slightly better than BBN based

ones, with the values of $R^2 > 0.93$ for FIS based models and $R^2 = 0.93$ for BBN based models. In this regard, it is clear that the results of the decision analysis approach that we propose correspond with the preferences of the studies in the literature.

Given the nature of the Fenton's dataset, machine learning based methods, in particular ANN and SVM, are not suitable for use due to learning constraints from a sufficient number of sample datasets. For example, a study suggesting neural network based prediction models implementing different ANN algorithms [142] reported the accuracy of experiments performed on the sample dataset between 0.41 and 0.77. These low performance values may be due to the fact that ANN approach does not work well with datasets that contain a small number of data points.

In addition, other machine learning based methods such as Naïve Bayes or Regression are not preferred to apply to the example dataset since they remain very simple to explain the various and complex properties of the given data. Although, Decision Tree method may be considered as appropriate because of its various capabilities such as modeling uncertain data, ease of use with low complexity and high performance in most cases [37], it does not work well with small sized datasets. Therefore, it can be concluded that the evidence for the methods used in the literature is consistent with our decision analysis results.

6. RECOMMENDATIONS

By considering the findings that were obtained during this thesis, we highly recommend that early software defect prediction models be constructed especially in requirements or design phases, using metrics that can be collected over early-stage artifacts as well as the metrics that focus on early-stage processes and resources. Most critical metrics would be based on the size or complexity of the early artifacts, effort of the review activities, stability of the requirements, maturity level of the organization, and experience of the project staff. Apart from the chosen metrics, the methods and techniques for building the prediction model are important for the nature of the data used. Most particularly, we recommend using fuzzy rule-based models in order to handle the qualitative and incomplete data of the early stages. Including contextual information is very important while designing the prediction models and reporting their results, which makes it possible to repeat the study and compare model performances. In addition, stronger empirical studies will increase the reliability of the ESDP models and build confidence in the predictive performance of these models.

The recommendations are grouped on the basis of the factors that have been discussed throughout the thesis.

SDLC phase and development methodology

Once again, it is important to note that the "early" statement of the development phase can be evaluated differently according to the SDLC. While in developments based on waterfall model it corresponds directly to the early stage that coincides with the beginning of the project, in developments based on incremental model it may coincide with the early stage within each increment. Thus, it may be necessary to use a feedback supporting model (such as Recurrent Neural Network) for performing SDP early on the iterations, which is beyond the scope of this thesis. Nevertheless, several important factors can be mentioned as follows: Outputs such as the documents of review activities or user stories that emerge during the life cycle are crucial for agile development methodologies. The effort information of the review activities and the number of defects found on reviews are the most important indicators. As project teams progress through iterations, static code metrics become more important since the number of lines of code increases.

Consequently, the lessons learned and suggestions regarding Waterfall development methodology are summarized below. Figure 6.1 demonstrates suitable SDP methods grouped with the metrics that give the most successful result within the phases related to Waterfall development methodology.

Waterfall

- Begin with building ESDP models using expert-based methods at the earliest.
    - Consider designing FIS based model by using requirement phase-based process data for a fresh start.
    - No need to design complex fuzzy rules at first, usage of tools like MATLAB Fuzzy Logic Toolbox is recommended.
    - Prefer process and resource-based metrics (effort for review activities, stability of requirements, maturity of the organization (i.e. CMMI level), and experience of the staff etc.).
    - Need for experts who know the related process and resource factors, which will affect possible defects.
    - Context information may undertake the task of guiding and can be helpful to build simple and effective models.
    - Make use of Bayes Network based models when qualitative and dependent data proliferate.
    - Use publicly available Bayes Network design and adapt it based on project's needs (AgenaRisk or Weka is recommended to start).
- Consider statistically based methods when the outputs of the requirement and design phases are obtained.
    - Regression based models are easy and accurate for a rapid start.
    - Analyze requirement and design phase outputs in order to gather product-based metrics (such as size of the requirement specifications or number of use cases).

- Benefit from process-based metrics to increase prediction performance.
- Make use of data from past projects (historical data) similar to the context of the current project, where available.

- Assess ML-based methods during coding phase.
  - Source code-based metrics are indispensable for building an ML-based model accurately.
  - Start with easy-to-use and high-performance methods like Naïve Bayes.

    When the dataset gains large number of data points, consider building ANN based models to increase accuracy, if practitioners are familiar with it.

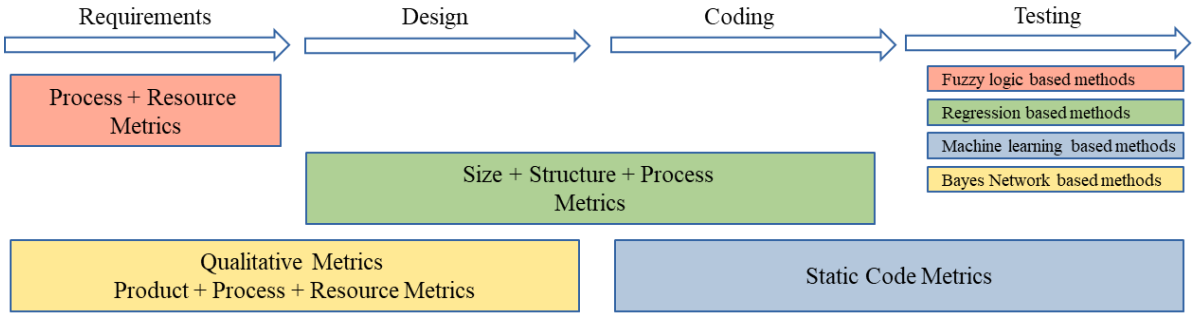


Figure 6.1. Recommended methods related to Waterfall phases with the most successful metric suites

Data – originated metrics

Different types of metrics can be evaluated at the requirements or design stages that originate metrics to build prediction models. According to Table 3.2, resulted metrics were mapped with the most suitable methods as follows:

Product-based

- Size metrics: number of uses cases, LOC.
  - Regression based methods should be preferred for accuracy.
  - FIS-based methods are also helpful at the earliest stages.
- Halstead size metrics:

- ML based method primarily preferable, especially Naïve Bayes.

- Structure: McCabe metrics.
  - Both regression and NB methods are applicable.
  - OO metrics: Regression based methods should be preferred primarily.

Process-based

- Number of defects: statistical.
- Effort, time, stability, process maturity: FIS based.

Resource-based

- Human characteristics: FIS based, BBN based
- Project characteristics: BBN based

Data – type and quality aspect

- Qualitative data
  - FIS-based or BN-based methods should be preferred.
- Quantitative data
  - Give priority to ML-based, Regression-based or BN-based methods, respectively

If data has:

- Dependency between attributes (causality):
  - Use of BN based model is recommended.
- Outliers:
  - ANN should be preferred first, however DT works well, too.
- Missing data points:
  - Bayesian methods, i.e., Naïve Bayes and Bayes Network should be preferred.

- Uncertainty and incompleteness:
    - Bayesian methods, i.e., Naïve Bayes and Bayes Network should be preferred first, yet some of ML-based methods suit well (such as ANN, DT).
- Vagueness, imprecision, and inconsistency:
    - FIS-based methods gain importance when experts are available.

7. CONCLUSION

In this chapter, the summary of this thesis, the contributions to the literature and threats to validity of the study are summarized. In addition, planned future work is presented.

7.1. Summary of Thesis

Software quality is the main proof of compliance for software products that enable proper and correct implementation of customer needs. Software quality assurance is therefore important not only in the later phases and also in the earlier phases of the SDLC. Software prediction models can help in the early detection of software defect proneness. Although it is relatively hard to collect early-stage data for every kind of project, findings from software process assessments or process audits might be significant in gathering the information needed by the early prediction models.

In this thesis, a decision analysis approach is proposed in order to select the best-fit SDP method according to the dataset characteristics and stakeholder needs that can be elicited via presented questionnaire in the early phases of the SDLC.

The introduction to the subject and overall goal of this thesis is presented in Chapter 1, as well as the research methods used throught the thesis and the main contributions. In Chapter 2, the detail information on the background is introduced. Chapter 3 identifies the related work from different aspects. Most importantly, evidence on the trends and maturity of research as well as the success and usefulness of early software defect prediction are investigated systematically. Besides, the survey is presented to gain insight into the current situation on SDP in Turkey. In Chapter 4, the details on decision analysis approach are presented, which includes solution architecture of the approach, alternatives and criteria, knowledge base, and the methodology for decision analysis, as well as the questionnaire. Chapter 5 demonstrates the investigation of the decision analysis approach through case studies implementing SDP on early phases and provides discussion on the results with regard to related literature. Chapter 6 highlights the recommendations. Lastly, Chapter 7 concludes the thesis by presenting the overall summary and contributions,

explaines the validity threats with the actions taken to minimize them, and reveals the future work.

According to our findings based on the systematic literature map and review; few studies reporting evaluation research indicate the need for conducting stronger empirical validation for future contributions in the context of ESDP. Product, process, and resource-based software metrics play an important role in building ESDP models, and there is a constant increase in the studies published especially as journal papers. The performances of many categorical studies and few continuous studies demonstrate evidence on the success of the ESDP models. Although the included studies mostly report that early prediction models are beneficial and useful to make effective resource planning, there is a need for further quantitative evidence on the benefits of using the ESDP models in practice.

According to the evidence obtained from the literature and our empirical studies, the prediction performance is satisfactory using early-stage data and supports the result of our decision analysis approach, in which Naïve Bayes and Logistic Regression based models are the most convenient methods for predicting the potential software defects in the early phases using NASA dataset. Similarly, the results obtained for NASA-93 dataset are quite consistent. The results comparison shows that the decision analysis recommends the same ranking with the experiments focused on performance criteria, in which Decision Tree based models are firstly chosen. Also, in terms of speed criteria, the rankings are very similar except DT and LinR. Finally, we perform an experiment based on the human judgement for Fenton’s dataset, which lacks sufficient data points for applying a learning algorithm. According to the case study results, the ranking between the FIS and BBN based models in terms of the performance criteria is the same as the experiment results.

As a main conclusion of this thesis, when evaluating possible alternative SDP methods and choosing a suitable one to build an SDP model in the early phases of the SDLC, we can say that we need to consider different aspects to address stakeholders' needs and various factors related to the defect dataset. Evidence from the case studies and the related

literature confirm the results of the decision analysis approach, in which the resulted scores reflect both the constraints of the given dataset within its specific context, and the requirements arising from these constraints. Thus, given which phase of SDLC we are in, what kind of data is available and what needs of stakeholders appear primarily, certain defect prediction methods may be more appropriate than others and should be preferred.

The decision analysis approach could be helpful and beneficial for software practitioners in deciding which prediction method they should use based on their specific needs. It might also serve as a guideline for stakeholders, especially for software project managers, in building their early SDP models that could support the management of the software development projects with effective resource, schedule and cost planning, thus ensuring higher quality software from the earliest phases of the projects.

Moreover, the systematic literature review in the field of early phase SDP may structure a protocol that future researchers can model. For instance, it can be improved by focusing on the early stages for iterative incremental and agile methods, rather than the requirements and design stages focused on in this thesis. In addition, it is thought that innovations such as expert opinion surveys prepared for SDP method selection and decision analysis approach in which different methods are used together will benefit researchers in many fields of software engineering.

It is also important to note that, the proposed decision analysis approach can be adapted not only in the context of software defect prediction, but also in different fields, such as software effort, cost, and reliability estimation. It can even be evaluated in areas that are outside the field of software engineering but require revising the questionnaire for that specific area, and also comparing many alternative classification of prediction methods and choosing the best. Having said that, it is also important to mention the possibility of the transformation of this proposed approach as a meta-learning model. It would be quite possible and useful to configure the proposed decision analysis approach as a meta-learning framework. It should be emphasized that this transformation may be possible not only in the SDP domain, but also in software engineering area where machine learning methods will be benchmarked in a wider scope.

7.2. Contributions

The contributions of this thesis to the literature can be highlighted in three main headings.

Extensive literature review on Early Software Defect Prediction

- Identifying the primary studies on ESDP and the main characteristics of their prediction models with regard to prediction methods, software metrics, datasets, contextual parameters, and performance evaluation approaches, as well as the addressed SDLC phases;
- Providing a classification scheme and mapping;
- Analysis of the prediction model design in the studies as well as their prediction performances, benefits or advantages of the early software defect prediction.

Proposing a new approach for selecting the most suitable prediction method

- Identifying the alternatives and criteria to be used in decision analysis process;
- Creation of a knowledge base regarding the evaluation of the various SDP methods according to identified criteria;
- Presenting a questionnaire that gathers the preferences of the stakeholders in the early phases of SDLC and the characteristics of the dataset subject to SDP;
- Proposing a two-phase decision analysis approach that combines decision tree and MCDA methodologies;
- Presenting case studies using public datasets and investigating the trustworthiness of the proposed decision analysis approach.

Making the outputs from the research available

- Providing the paper repository regarding a classification scheme;
- Publishing the criteria, alternatives, and their evaluations in the context of ESDP within the knowledge base;

- Publishing the source code of the web-based application that includes the implementation of the Fuzzy TOPSIS evaluation and demonstrates its usage on ESDP.

7.3. Threads to Validity

The potential threats to validity of the thesis have been systematically identified and addressed by taking steps to minimize or mitigate them. Below, the main threats to the validity of this study are discussed based on the checklist adopted from Wohlin et al. [27].

7.3.1. Internal Validity

Threats to internal validity are influences that can affect the independent variable with respect to causality, without the researcher's knowledge.

Perception of the term "early"

The most important threat in this study may be the author's consideration of the term "early" to refer to requirements and/or design phases. However, a pre-release phase could be thought as early in incremental or agile development projects. Although the initial searches showed that there were few primary studies on software defect prediction in agile or incremental development, we could not have the chance to retrieve all such studies by the selected search strategy. There may be a need for conducting further research that investigates studies on pre-release defect prediction in such contexts. Especially in recent years, the interest in agile development methods has increased a lot. Since the transition to agile methodologies is relatively recent compared to the use of plan-based methods such as waterfall, there is limited literature or data on the related topics, i.e., defect prediction at early stages of the development. Therefore, the phase information discussed in this thesis does not fully correspond to the agile environment. In this context, it is worth emphasizing once again that the term "early" should be interpreted as independent from the software development model employed, and as dependent to the phases of requirements or design that originate metrics for building prediction models where applicable. Consequently, in order not to perceive the term

"early" for different contexts, the requirement and design phases were frequently used throughout the thesis, to mitigate this validity.

Researcher bias

The researcher bias is one of the main internal validity threats for the literature analysis part of this thesis since the author conducted most of the search process for the literature review. To reduce this threat, some actions have been taken. Firstly, the review protocol of the research was prepared with thesis supervisor to ensure the clarity of the design of research methodology. To increase the reliability of our review protocol, we reviewed and considered the protocols of other secondary studies. Secondly, a detailed data extraction form was defined to make the extraction well-structured. The classification scheme and extracted data were peer-reviewed by the supervisor on a test-set of including papers with sampling, which shows a very high degree of agreement to involve the related papers. Few corrections required after the peer-review was reflected in the overall data extraction and analysis.

Selecting the decision analysis methodology

As reported in Chapter 2.3.2, there are different implementations of MCDA, such as AHP, PROMETHEE, ELECTRE etc. and the selection of an appropriate MCDA tool is not an easy task. At this point, we followed the guideline by [143] as a general framework for selecting a suitable MCDA methodology for the specified area of decision analysis. Thus, we considered using Fuzzy TOPSIS approach, since the hybrid usage of MCDA methodology and fuzzy set theory provides solutions for decision makers to handle incomplete, vague, and ambiguous knowledge.

Identifying participants to the expert opinion studies

Another threat might have occurred due to selection bias of the experts who volunteered to participate in the expert opinion surveys. Due to the extensive literature search in the field, the expert profiles from the academy were gathered systematically by considering their contributions to the SDP area and Google Scholar profiles. In addition, thanks to the

authors' 10+ years of experience in the industry, access has been provided to experts from the industry who are relevant to the subject.

7.3.2. Construct validity

Construct validity is concerned with the relation between the study structure and the actual reflection of the research.

Suitability of literature review

Threats related to this type of validity might be suitability of research questions and classification scheme used for data extraction through systematic literature review. Research questions were answered based on a classification scheme, which was designed based on the standards adapted from [24]. Also, we finalized the scheme through several iterations until we extracted all related information with the research questions.

Defining criteria and evaluation in decision analysis

Threats to construct validity could also occur while identifying the criteria and rating them for each alternative. To minimize this threat, we conducted two expert opinion studies, so that we could design the decision analysis approach in a more reliable and robust way. First, an expert opinion questionnaire was administered to finalize and rank the initially defined criteria. After, another expert opinion survey was applied to evaluate the alternatives on the basis of criteria, and the support of experts from both academia and the sector was received. Thus, the resulting knowledge base was supported and strengthened by expert opinions as well as basing it on the literature. Besides, we explained our mindset and reasoning by providing the comprehensive literature review results through the base decision matrix within a knowledge base and by grounding the definitions we mentioned throughout the decision analysis process.

Method selection as alternatives to the decision analysis

Another threat would be about selection of the prediction methods to evaluate in the decision analysis. We did not include any ensemble methods (e.g., Random Forest), since

the choice and implementation of the algorithm could have impacted the prediction results; so we only included basic algorithm implementations in our case studies. Besides, we did not have any knowledge base about ensemble algorithms, as they did not appear in the literature review at the first place, thus we did not include them in our surveys. Nevertheless, one may replicate the decision analysis study using today's popular prediction methods, such as Random Forest and Deep Learning.

Lack of parameter optimization

Last and most important validity threat is related to the parameter setting. Although there are several studies that recommends parameter tuning [144,145], we used the default parameters for each classifier used in our study and did not apply any optimization on the parameters in order to minimize this threat. By doing so, we could compare the performances of the classifiers as they were. However, especially for some methods like ANN, DT or SVM, parameter optimization may affect the overall ranking of the classifiers, where the improment in the prediction performance after the optimization is non-negligible for some of the classification techniques in [145]. Conversely, the performance of the NB model built with default values is as stable as optimized models. Therefore, we believe that our comparison will be the base for possible future studies, which may perform deeper experiments by using parameter tuning.

7.3.3. Conclusion validity

This type of validity is mainly concerned with the reliability of the proposed study by examining its reproducibility.

Reliability of the proposed approach

It is concerned with the reliability of the decision analysis that if it is applied by other researchers, the outcome shall be the same. By defining details of the methodologies and processes followed, we ensure reliability to the extent that researchers who apply our approach with the same requirements would come up with the same results. Moreover, a web application of the decision analysis was developed using Angular, Java and Spring framework. The source code made available and shared on GitHub to enable researchers

or practitioners to perform the decision analysis using the determined criteria, weights, and the list of selected alternatives. When the application is executed according to the scenarios given in the case studies, SDP methods and their ranking recommended by the decision analysis can be validated.

Replicability of literature review

This type of validity also considers the relationship between the data collected from primary studies and the results/conclusions. We wanted to make sure that there was a high degree of traceability between data and conclusions. In order to ensure this, we publicly shared the spreadsheet of the data gathered from literature review [99].

7.3.4. External validity

The threat to external validity is related to the ability to generalize the results of the empirical studies in broader contexts.

As only three contexts are used for validation of the proposed decision analysis approach, threat to external validity is a major threat to this thesis. Although we presented the process for selecting the most convenient SDP method to build a prediction model in the early phases within our proposed approach, it would be obviously misleading to argue that the decision analysis approach will give the correct result under all circumstances. It is important to say that as the number of experiments carried out by collecting metrics from software projects involving different context information increases, the base on which the decision analysis approach is depending on would be strengthened and the results could be more trustworthy.

7.4. Future Work

Today, most of the SDP studies in the literature focus on developing, improving, and evaluating prediction models empirically with the help of various software metrics. Such studies contribute significantly to the research on software engineering, but few studies create research-based tools for the practical application and advancement of SDP

research. In the future, we plan to focus on improving our proposed approach to create a tool that guides practitioners in building and incorporating SDP models from the beginning of their software projects. To do this, first the usability of our decision analysis application package should be investigated. Then, our package can also be integrated with widely used project management tools available in the industry to facilitate the usage.

While the results of this thesis provide insight for future research on the context of ESDP, further evidence on different software projects are necessary in order to enhance decision analysis process and make stronger inferences. For future work, the validation of the proposed decision analysis approach will be investigated by conducting further empirical studies with data from different software projects. Besides, we plan to study on improvements of the knowledge base by including further experts and their opinions. There is also some space to the validation of the proposed decision analysis approach by conducting further empirical studies with data collected from industry in Turkey. Accordingly, we plan to include software development methodology information to the decision analysis process in our future studies.

8. BIBLIOGRAPHY

[1] B. Boehm, V.R. Basili, Software Defect Reduction Top 10 List, Computer (Long. Beach. Calif). 34 (2001) 135–137. https://doi.org/10.1109/2.962984.

[2] C. Jones, O. Bonsignour, The Economics of Software Quality, 1st ed., Addison-Wesley Professional, 2011.

[3] B.W. Boehm, Software engineering economics, Prentice Hall , 1981.

[4] B. Haskins, J. Stecklein, B. Dick, G. Moroney, R. Lovell, J. Dabney, Error Cost Escalation Through the Project Life Cycle, 2004. https://ntrs.nasa.gov/citations/20100036670 (accessed April 24, 2022).

[5] C. Smidts, M. Stutzke, R.W. Stoddard, Software reliability modeling: an approach to early reliability prediction, IEEE Trans. Reliab. 47 (1998) 268–278. https://doi.org/10.1109/24.740500.

[6] L.C. Briand, J. Wüst, Empirical Studies of Quality Models in Object-Oriented Systems, Adv. Comput. 56 (2002) 97–166. https://doi.org/10.1016/S0065-2458(02)80005-5.

[7] B. Cukic, J.H. Hayes, The Virtues of Assessing Software Reliability Early, IEEE Softw. 22 (2005) 50–53. https://doi.org/10.1109/MS.2005.79.

[8] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, IEEE Trans. Softw. Eng. 38 (2012) 1276–1304. https://doi.org/10.1109/TSE.2011.103.

[9] J.S. Collofello, S.N. Woodfield, Evaluating the effectiveness of reliability-assurance techniques, J. Syst. Softw. 9 (1989) 191–195. https://doi.org/10.1016/0164-1212(89)90039-3.

[10] C. Catal, B. Diri, A systematic review of software fault prediction studies, Expert Syst. Appl. 36 (2009) 7346–7354. https://doi.org/10.1016/j.eswa.2008.10.027.

[11] C. Catal, B. Diri, Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem, Inf. Sci. (Ny). 179 (2009) 1040–1058. https://doi.org/10.1016/J.INS.2008.12.001.

[12] A.K. Pandey, N.K. Goyal, Early Software Reliability Prediction, in: Early Softw. Reliab. Predict., Springer, India, 2013: pp. 17–33. https://doi.org/10.1007/978-81-

322-1176-1.

[13] N.E. Fenton, I.C. Society, M. Neil, I.C. Society, A Critique of Software Defect Prediction Models, 25 (1999) 675–689.

[14] Q. Song, Z. Jia, M. Shepperd, S. Ying, J. Liu, A general software defect-proneness prediction framework, IEEE Trans. Softw. Eng. 37 (2011) 356–370.

[15] R. Özakıncı, A. Tarhan, Early software defect prediction: A systematic map and review, J. Syst. Softw. 144 (2018) 216–239. https://doi.org/10.1016/j.jss.2018.06.025.

[16] IEEE, IEEE Std 1012-2016 (Revision of IEEE Std 1012-2012/ Incorporates IEEE Std 1012-2016/Cor1-2017) : IEEE Standard for System, Software, and Hardware Verification and Validation., (2017).

[17] S.H. Kan, Metrics and models in software quality engineering, Second Edi, Addison Wesley, 2003.

[18] A. Aydin, A. Tarhan, Investigating defect prediction models for iterative software development when phase data is not recorded lessons learned, in: 9th Int. Conf. Eval. Nov. Approaches to Softw. Eng., 2014: pp. 1–11.

[19] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy, Cross-project defect prediction, Proc. 7th Jt. Meet. Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. Eur. Softw. Eng. Conf. Found. Softw. Eng. Symp. - E. (2009) 91. https://doi.org/10.1145/1595696.1595713.

[20] N. Fenton, M. Neil, W. Marsh, P. Hearty, Łu. Radliński, P. Krause, Project data incorporating qualitative factors for improved software defect prediction, in: Proc. - ICSE 2007 Work. Third Int. Work. Predict. Model. Softw. Eng. PROMISE'07, 2007.

[21] N. Fenton, M. Neil, W. Marsh, P. Hearty, Ł. Radliński, P. Krause, On the effectiveness of early life cycle defect prediction with Bayesian nets, Empir. Softw. Eng. 13 (2008) 499–537. https://doi.org/10.1007/s10664-008-9072-x.

[22] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Inf. Softw. Technol. 64 (2015) 1–18. https://doi.org/10.1016/j.infsof.2015.03.007.

[23] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, 2007. https://doi.org/10.1145/1134285.1134500.

[24] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic Mapping Studies in Software Engineering, 12th Int. Conf. Eval. Assess. Softw. (2008) 68–77.

[25] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering - A systematic literature review, Inf. Softw. Technol. 51 (2009) 7–15. https://doi.org/10.1016/j.infsof.2008.09.009.

[26] R.K. Yin, Case Study Research and Applications: Design and Methods, 6th ed., SAGE Publications, Inc, 2017.

[27] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Publishing Company, Incorporated, 2012. https://doi.org/10.1007/978-3-642-29044-2.

[28] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2009) 131–164. https://doi.org/10.1007/s10664-008-9102-8.

[29] B. Kitchenham, L. Pickard, S.L. Pfleeger, Case studies for method and tool evaluation, IEEE Softw. 12 (1995) 52–62. https://doi.org/10.1109/52.391832.

[30] J.W. Creswell, Research Design - Qualitative, Quantitative, and Mixed Methods Approaches, 4rd ed., Sage Publications, Thousand Oaks, California, 2014. https://doi.org/0.1177/2050312117740990.

[31] R. Ozakinci, A. Tarhan, The role of process in early software defect prediction: Methods, attributes and metrics, in: Commun. Comput. Inf. Sci., 2016: pp. 287–300. https://doi.org/10.1007/978-3-319-38980-6_21.

[32] R. Özakıncı, A. Tarhan, Yazilim gelistirmede erken asamalarda toplanan verinin hata tahmini performansina etkisi, in: CEUR Workshop Proc., 2016: pp. 532–543. http://ceur-ws.org/Vol-1721/UYMS16_paper_120.pdf (accessed October 12, 2017).

[33] R. Özakıncı, A. Tarhan, An Evaluation Approach for Selecting Suitable Defect Prediction Method at Early Phases, in: Proc. - 45th Euromicro Conf. Softw. Eng.

Adv. Appl. SEAA 2019, 2019: pp. 199–203. https://doi.org/10.1109/SEAA.2019.00040.

[34] R. Ozakinci, A. Kolukısa Tarhan, A Decision Analysis Approach for Selecting Software Defect Prediction Method in the Early Phases, Softw. Qual. J. (2022). (Under Minor Revision).

[35] IEEE, IEEE Standard Classification for Software Anomalies (IEEE 1044 - 2009), IEEE Std. 1044 (2009) 1–4.

[36] IEEE Reliability Society, IEEE Std 1633™-2008, IEEE Recommended Practice on Software Reliability, IEEE Std. (2008) 1–69.

[37] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Elsevier Inc., 2012. https://doi.org/10.1016/B978-0-12-381479-1.00001-0.

[38] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Pratical Machine Learning Tool and Tecniques, 3rd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[39] G.J. Klir, B. Yuan, Fuzzy sets and fuzzy logic : theory and applications, Prentice Hall PTR, 1995.

[40] Y. Zhou, N. Fenton, M. Neil, C. Zhu, Incorporating Expert Judgement into Bayesian Network Machine Learning, in: Proc. Twenty-Third Int. Jt. Conf. Artif. Intell. Inc., 2013: pp. 3249–3250.

[41] M. Kuhn, K. Johnson, Applied predictive modeling, Springer, 2013. https://doi.org/10.1007/978-1-4614-6849-3.

[42] M.M.T. Thwin, T.S. Quah, Application of neural networks for software quality prediction using object-oriented metrics, J. Syst. Softw. 76 (2005) 147–156. https://doi.org/10.1016/j.jss.2004.05.001.

[43] T.M. Khoshgoftaar, E.B. Allen, J.P. Hudepohl, S.J. Aud, Application of neural networks to software quality modeling of a very large telecommunications system, IEEE Trans. Neural Networks. 8 (1997) 902–909. https://doi.org/10.1109/72.595888.

[44] Q. Wang, B. Yu, J. Zhu, Extract rules from software quality prediction model based on neural network, in: Proc. - Int. Conf. Tools with Artif. Intell. ICTAI, 2004: pp.

191–195. https://doi.org/10.1109/ICTAI.2004.62.

[45] G.J. Pai, J.B. Dugan, Empirical analysis of software fault content and fault proneness using Bayesian methods, IEEE Trans. Softw. Eng. 33 (2007) 675–686. https://doi.org/10.1109/TSE.2007.70722.

[46] T. Menzies, J.S. DiStefano, A.S. Orrego, Assessing Predictors of Software Defects, in: Work. Predict. Softw. Model., 2004.

[47] S. Amasaki, Y. Takagi, O. Mizuno, T. Kikuno, A Bayesian belief network for assessing the likelihood of fault content, in: Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE, IEEE Computer Society, 2003: pp. 215–226. https://doi.org/10.1109/ISSRE.2003.1251044.

[48] T.M. Khoshgoftaar, N. Seliya, Fault prediction modeling for software quality estimation: Comparing commonly used techniques, Empir. Softw. Eng. 8 (2003) 255–283. https://doi.org/10.1023/A:1024424811345.

[49] W. Afzal, R. Torkar, A comparative evaluation of using genetic programming for predicting fault count data, in: Proc. - 3rd Int. Conf. Softw. Eng. Adv. ICSEA 2008, Incl. ENTISY 2008 Int. Work. Enterp. Inf. Syst., 2008: pp. 407–414. https://doi.org/10.1109/ICSEA.2008.9.

[50] M. Reformat, A fuzzy-based meta-model for reasoning about the number of software defects, in: Lect. Notes Artif. Intell. (Subseries Lect. Notes Comput. Sci., Springer Verlag, 2003: pp. 644–651. https://doi.org/10.1007/3-540-44967-1_77.

[51] X. Yuan, T.M. Khoshgoftaar, E.B. Allen, K. Ganesan, An application of fuzzy clustering to software quality prediction, in: Proc. - 3rd IEEE Symp. Appl. Syst. Softw. Eng. Technol., Institute of Electrical and Electronics Engineers Inc., 2000: pp. 85–90. https://doi.org/10.1109/ASSET.2000.888052.

[52] S. Sup So, S. Deok Cha, Y. Rae Kwon, Empirical evaluation of a fuzzy logic-based software quality prediction model, Fuzzy Sets Syst. 127 (2002) 199–208. https://doi.org/10.1016/S0165-0114(01)00128-2.

[53] IEEE Standard for a Software Quality Metrics Methodology, IEEE Std 1061-1998. (1998) i-. https://doi.org/10.1109/IEEESTD.1998.243394.

[54] N. Fenton, J. Bieman, Software Metrics: A Rigorous and Practical Approach, 2014. https://doi.org/10.1201/b17461.

[55] S.R. Chidamber, C.F. Kemerer, A metrics suite for object oriented design, IEEE Trans. Softw. Eng. 20 (1994) 476–493. https://doi.org/10.1109/32.295895.

[56] R. Jabangwe, J. Börstler, D. Šmite, C. Wohlin, Empirical evidence on the link between object-oriented measures and external quality attributes: a systematic literature review, Empir. Softw. Eng. 20 (2014) 640–693. https://doi.org/10.1007/s10664-013-9291-7.

[57] D. Aslan, A. Tarhan, ve O. Demirörs, How Process Enactment Data Affects Product Defectiveness Prediction - A Case Study, Softw. Eng. Res. Manag. Appl. Stud. Comput. Intell. 496 (2014) 151–166. https://doi.org/10.1007/978-3-319-00948-3_10.

[58] T. Menzies, R. Krishna, D. Pryor, The Promise Repository of Empirical Software Engineering Data, (2016). http://openscience.us/repo.

[59] M. Shepperd, Q. Song, Z. Sun, C. Mair, NASA MDP Dataset, A Backup Site NASA Defect Datasets That Were Orig. Publ. by Shepperd Al., (2013). (2013).

[60] T. Menzies, nasa93, (2008). https://doi.org/10.5281/ZENODO.268419.

[61] C. Abts, B. Clark, S. Devnani-Chulani, E. Horowitz, R. Madachy, D. Reifer, R. Selby, B. Steece, Cocomo II Model Definition Manual, (1998).

[62] Y. Jiang, B. Cukic, Y. Ma, Techniques for evaluating fault prediction models, Empir. Softw. Eng. 13 (2008) 561–595. https://doi.org/10.1007/s10664-008-9079-3.

[63] T.J. Ostrand, E.J. Weyuker, How to measure success of fault prediction models, in: Fourth Int. Work. Softw. Qual. Assur. Conjunction with 6th ESEC/FSE Jt. Meet. - SOQUA '07, ACM Press, New York, New York, USA, 2007: p. 25. https://doi.org/10.1145/1295074.1295080.

[64] E. Erturk, E. Akcapinar Sezer, Iterative software fault prediction with a hybrid approach, Appl. Soft Comput. J. 49 (2016) 1020–1033. https://doi.org/10.1016/J.ASOC.2016.08.025.

[65] D. Baker, D. Bridges, R. Hunter, G. Johnson, J. Krupa, J. Murphy, K. Sorenson, Guidebook to decision-making methods, USA, 2001.

[66] J. Fulop, Introduction to Decision Making Methods, 2005.

https://doi.org/10.1.1.86.6292.

[67] J. Dodgson, M. Spackman, A. Pearman, L. Phillips, Multi-criteria analysis: a manual, Department for Communities and Local Government, London, 2009.

[68] J.R. Quinlan, Decision Trees and Decision-making, IEEE Trans. Syst. Man Cybern. 20 (1990) 339–346. https://doi.org/10.1109/21.52545.

[69] V. Belton, T. Stewart, Multiple Criteria Decision Analysis: An Integrated Approach, Springer US, 2002. https://doi.org/10.1007/978-1-4615-1495-4.

[70] W.A. Goh, Applying Multi-Criteria Decision Analysis for Software Quality Assessment Methods, Blekinge Institute of Technology, Sweden, 2010.

[71] T.L. Saaty, Axiomatic Foundation of the Analytic Hierarchy Process, Manage. Sci. 32 (1986) 841–855. https://doi.org/10.1287/mnsc.32.7.841.

[72] J.R. Figueira, V. Mousseau, B. Roy, ELECTRE methods, Int. Ser. Oper. Res. Manag. Sci. 233 (2016) 155–185. https://doi.org/10.1007/978-1-4939-3094-4_5.

[73] C. Hwang, K. Yoon, Multiple Attribute Decision Making: Methods and Applications, A State of the Art Survey, Springer Berlin Heidelberg, 1981. https://doi.org/10.1007/978-3-642-48318-9.

[74] J.-P. Brans, B. Mareschal, PROMETHEE methods, in: Int. Ser. Oper. Res. Manag. Sci., Springer New York LLC, 2005: pp. 163–195. https://doi.org/10.1007/0-387-23081-5_5.

[75] R.E. Bellman, L.A. Zadeh, Decision-Making in a Fuzzy Environment, Manage. Sci. 17 (1970) B-141-B-164. https://doi.org/10.1287/mnsc.17.4.b141.

[76] S.-J. Chen, C.-L. Hwang, Fuzzy Multiple Attribute Decision Making: Methods and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992. https://doi.org/10.1007/978-3-642-46768-4.

[77] E. Triantaphyllou, Multi-criteria Decision Making Methods: A Comparative Study, Springer US, 2000. https://doi.org/10.1007/978-1-4757-3157-6.

[78] C.T. Chen, Extensions of the TOPSIS for group decision-making under fuzzy environment, Fuzzy Sets Syst. 114 (2000) 1–9. https://doi.org/10.1016/S0165-0114(97)00377-1.

[79] C.T. Chen, C.T. Lin, S.F. Huang, A fuzzy approach for supplier evaluation and

selection in supply chain management, Int. J. Prod. Econ. 102 (2006) 289–301. https://doi.org/10.1016/j.ijpe.2005.03.009.

[80] S. Nădăban, S. Dzitac, I. Dzitac, Fuzzy TOPSIS: A General View, Procedia Comput. Sci. 91 (2016) 823–831. https://doi.org/10.1016/j.procs.2016.07.088.

[81] D. Radjenović, M. Heričko, R. Torkar, A. Živkovič, Software fault prediction metrics: A systematic literature review, Inf. Softw. Technol. 55 (2013) 1397–1418. https://doi.org/10.1016/j.infsof.2013.02.009.

[82] R. Malhotra, A systematic review of machine learning techniques for software fault prediction, Appl. Soft Comput. J. 27 (2015) 504–518. https://doi.org/10.1016/j.asoc.2014.11.023.

[83] R.S. Wahono, A Systematic Literature Review of Software Defect Prediction : Research Trends , Datasets , Methods and Frameworks, J. Softw. Eng. 1 (2015) 1–16.

[84] J. Murillo-Morera, C. Quesada-López, M. Jenkins, Software Fault Prediction: A Systematic Mapping Study, in: CIBSE 2015 - XVIII Ibero-American Conf. Softw. Eng., 2015: pp. 446–459.

[85] C. Catal, Software fault prediction: A literature review and current trends, Expert Syst. Appl. 38 (2011) 4626–4636. https://doi.org/10.1016/j.eswa.2010.10.024.

[86] M. Jureczko, L. Madeyski, A review of process metrics in defect prediction studies, Methods Appl. Comput. Sci. 1 (2011) 133–145.

[87] P.K. Singh, D. Agarwal, A. Gupta, A Systematic Review on Software Defect Prediction, in: 2nd Int. Conf. Comput. Sustain. Glob. Dev., 2015: pp. 1793–1797.

[88] D. Wahyudin, R. Ramler, S. Biffl, A framework for defect prediction in specific software project contexts, in: Proc. Third IFIP TC 2 Cent. East Eur. Conf. Softw. Eng. Tech., Springer-Verlag Berlin, Heidelberg, Brno, Czech Republic, 2008: pp. 261–274. https://doi.org/10.1007/978-3-642-22386-0_20.

[89] S.N. Das Dôres, L. Alves, D.D. Ruiz, R.C. Barros, A meta-learning framework for algorithm recommendation in software fault prediction, Proc. ACM Symp. Appl. Comput. April (2016) 1486–1491. https://doi.org/10.1145/2851613.2851788.

[90] F. Porto, L. Minku, E. Mendes, A. Simao, A Systematic Study of Cross-Project

Defect Prediction With Meta-Learning, 2018. https://doi.org/https://doi.org/10.48550/arXiv.1802.06025.

[91] Y. Ma, G. Luo, X. Zeng, A. Chen, Transfer learning for cross-company software defect prediction, Inf. Softw. Technol. 54 (2012) 248–256. https://doi.org/10.1016/J.INFSOF.2011.09.007.

[92] Q. Cao, Q. Sun, Q. Cao, H. Tan, Software defect prediction via transfer learning based neural network, in: Proc. 2015 1st Int. Conf. Reliab. Syst. Eng. ICRSE 2015, Institute of Electrical and Electronics Engineers Inc., 2015. https://doi.org/10.1109/ICRSE.2015.7366475.

[93] S. Tang, S. Huang, C. Zheng, E. Liu, C. Zong, Y. Ding, A novel cross-project software defect prediction algorithm based on transfer learning, Tsinghua Sci. Technol. 27 (2022) 41–57. https://doi.org/10.26599/TST.2020.9010040.

[94] S.S. Rathore, S. Kumar, A decision tree logic based recommendation system to select software fault prediction techniques, Computing. 99 (2017) 255–285. https://doi.org/10.1007/s00607-016-0489-6.

[95] A.O. Balogun, A.O. Bajeh, V.A. Orie, A.W. Yusuf-asaju, Software Defect Prediction Using Ensemble Learning: An ANP Based Evaluation Method, J. Eng. Technol. 3 (2018) 50–55.

[96] Y. Peng, G. Kou, G. Wang, W. Wu, Y. Shi, Ensemble of Software Defect Predictors: an Ahp-Based Evaluation Method, Int. J. Inf. Technol. Decis. Mak. 10 (2011) 187–206. https://doi.org/10.1142/s0219622011004282.

[97] W. Wu, Extension of Analytic Hierarchy Model for High-Efficiency Clustering in Software Defect Prediction, Int. J. Manag. Sci. 2 (2015) 13–20.

[98] G. Kou, Y. Peng, Y. Shi, W. Wu, Classifier evaluation for software defect prediction, Stud. Informatics Control. 21 (2012) 117–126. https://doi.org/10.24846/v21i2y201201.

[99] R. Özakıncı, A. Tarhan, Paper Repository and References for “Early software defect prediction: A systematic map and review,” (2017). https://doi.org/10.5281/ZENODO.3621223.

[100] K. Petersen, Measuring and predicting software productivity: A systematic map and review, Inf. Softw. Technol. 53 (2011) 317–343.

https://doi.org/10.1016/j.infsof.2010.12.001.

[101] A. Idri, A. Abran, Analogy-based software development effort estimation : A systematic mapping and review, Inf. Softw. Technol. 58 (2015) 206–230.

[102] W.A. Florac, R.E. Park, A. Carleton, Practical Software Measurement: Measuring for Process Management and Improvement, 1997.

[103] K. Petersen, C. Wohlin, Context in industrial software engineering research, in: 2009 3rd Int. Symp. Empir. Softw. Eng. Meas. ESEM 2009, 2009: pp. 401–404. https://doi.org/10.1109/ESEM.2009.5316010.

[104] M. Dixon-Woods, S. Agarwal, D. Jones, B. Young, A. Sutton, Synthesising qualitative and quantitative evidence: a review of possible methods, J Heal. Serv Res Policy. 10 (2005) 45–53. https://doi.org/10.1258/1355819052801804.

[105] S. Hosseini, B. Turhan, D. Gunarathna, A Systematic Literature Review and Meta-Analysis on Cross Project Defect Prediction, IEEE Trans. Softw. Eng. 45 (2017) 111–147. https://doi.org/10.1109/TSE.2017.2770124.

[106] R. Rana, Software Defect Prediction Techniques in Automotive Domain : Evaluation , Selection and Adoption, Chalmers University of Technology & University of Gothenburg, 2015. https://doi.org/10.13140/RG.2.1.1452.8160.

[107] V.U.B. Challagulla, F.B. Bastani, I.L. Yen, R.A. Paul, Empirical assessment of machine learning based software defect prediction techniques, Int. J. Artif. Intell. Tools. 17 (2008) 389–400. https://doi.org/10.1142/S0218213008003947.

[108] A. Motro, Sources of Uncetainty, Imprecision, and Inconsistency in Information Systems, Uncertain. Manag. Inf. Syst. (1996) 9–34. https://doi.org/10.1080/03639040801928762.

[109] W. Zhang, Y. Yang, Q. Wang, Handling missing data in software effort prediction with naive Bayes and em algorithm, in: 7th Int. Conf. Predict. Model. Softw. Eng. (Promise '11), 2011. https://doi.org/10.1145/2020390.2020394.

[110] O. Alan, C. Catal, An outlier detection algorithm based on object-oriented metrics thresholds, in: 2009 24th Int. Symp. Comput. Inf. Sci. Isc. 2009, 2009: pp. 567–570. https://doi.org/10.1109/ISCIS.2009.5291882.

[111] W.J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, B. Yu, Definitions, methods,

and applications in interpretable machine learning, Proc. Natl. Acad. Sci. U. S. A. 116 (2019) 22071–22080. https://doi.org/10.1073/pnas.1900654116.

[112] I. Portugal, P. Alencar, D. Cowan, The use of machine learning algorithms in recommender systems: A systematic review, Expert Syst. Appl. 97 (2018) 205–227. https://doi.org/10.1016/j.eswa.2017.12.020.

[113] I. Mahdavi, A. Heidarzade, B. Sadeghpour-Gildeh, N. Mahdavi-Amiri, A general fuzzy TOPSIS model in multiple criteria decision making, Int. J. Adv. Manuf. Technol. 45 (2009) 406–420. https://doi.org/10.1007/s00170-009-1971-5.

[114] D.K. Yadav, S.K. Chaturvedi, R.B. Misra, Early software defects prediction using fuzzy logic, Int. J. Performability Eng. 8 (2012) 399–408.

[115] F. Sitorus, J.J. Cilliers, P.R. Brito-Parada, Multi-criteria decision making for the choice problem in mining and mineral processing: Applications and trends, Expert Syst. Appl. 121 (2019) 393–417. https://doi.org/10.1016/j.eswa.2018.12.001.

[116] B. Sodhi, T.V. Prabhakar, A Simplified Description of Fuzzy TOPSIS, (2012) 1–4.

[117] M. Shepperd, Q. Song, Z. Sun, C. Mair, Data quality: Some comments on the NASA software defect datasets, IEEE Trans. Softw. Eng. 39 (2013) 1208–1215. https://doi.org/10.1109/TSE.2013.11.

[118] Minitab 18.1, Statistical & Data Analysis Software Package | Minitab, (2017).

[119] T. Pohlert, PMCMRplus: Calculate Pairwise Multiple Comparisons of Mean Rank Sums Extended, (2022). https://cran.r-project.org/web/packages/PMCMRplus/index.html.

[120] R. Özakıncı, A. Tarhan, A Decision Analysis Approach for Selecting Software Defect Prediction Method in the Early Phases - Case Study Data, Experiments, and Results, (2021). https://doi.org/10.5281/zenodo.6478564.

[121] Z. Jiang, Y., Lin, J., Cukic, B., Lin, S., & Hu, Replacing Code Metrics in Software Fault Prediction with Early Life Cycle Metrics, in: Third Int. Conf. Inf. Sci. Technol., 2013: pp. 516–523. https://doi.org/10.1109/SCC.2014.108.

[122] T.J. McCabe, A Complexity Measure, IEEE Trans. Softw. Eng. SE-2 (1976) 308–320. https://doi.org/10.1109/TSE.1976.233837.

[123] T. Menzies, J. Greenwald, A. Frank, Data mining static code attributes to learn defect predictors, IEEE Trans. Softw. Eng. 33 (2007) 2–13. https://doi.org/10.1109/TSE.2007.256941.

[124] E. Frank, H. Mark A., W. Ian H., The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques," (2016).

[125] M. Friedman, A Comparison of Alternative Tests of Significance for the Problem of m Rankings, Ann. Math. Stat. 11 (1940) 86–92. https://www.jstor.org/stable/2235971 (accessed March 28, 2022).

[126] J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[127] D.G. Pereira, A. Afonso, F.M. Medeiros, Overview of Friedmans Test and Post-hoc Analysis, Commun. Stat. Simul. Comput. 44 (2015) 2636–2653. https://doi.org/10.1080/03610918.2014.931971.

[128] B. Ghotra, S. McIntosh, A.E. Hassan, Revisiting the impact of classification techniques on the performance of defect prediction models, in: Proc. - Int. Conf. Softw. Eng., IEEE Computer Society, 2015: pp. 789–800. https://doi.org/10.1109/ICSE.2015.91.

[129] A. Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana, M. Ahmad, A. Husen, Performance analysis of machine learning techniques on software defect prediction using NASA datasets, Int. J. Adv. Comput. Sci. Appl. 10 (2019) 300–308. https://doi.org/10.14569/ijacsa.2019.0100538.

[130] Y. Ma, S. Zhu, K. Qin, G. Luo, Combining the requirement information for software defect estimation in design time, Inf. Process. Lett. 114 (2014) 469–474. https://doi.org/10.1016/j.ipl.2014.03.012.

[131] P. Singh, S. Verma, O.P. Vyas, Software fault prediction at design phase, J. Electr. Eng. Technol. 9 (2014) 1739–1745. https://doi.org/10.5370/JEET.2014.9.5.1739.

[132] P. Singh, S. Verma, Cross Project Software Fault Prediction at Design Phase, Int. J. Comput. Electr. Autom. Control Inf. Eng. 9 (2015) 800–805. https://doi.org/10.5370/JEET.2014.9.4.742.

[133] MATLAB, https://www.mathworks.com/products/fuzzy-logic.html, (n.d.).

[134] S. Chatterjee, B. Maji, A new fuzzy rule based algorithm for estimating software faults in early phase of development, Soft Comput. 20 (2016) 4023–4035. https://doi.org/10.1007/s00500-015-1738-x.

[135] C. Kumar, D.K. Yadav, Software defects estimation using metrics of early phases of software development life cycle, Int. J. Syst. Assur. Eng. Manag. (2014). https://doi.org/10.1007/s13198-014-0326-2.

[136] S. Chatterjee, B. Maji, A bayesian belief network based model for predicting software faults in early phase of software development process, Appl. Intell. 48 (2018) 2214–2228. https://doi.org/https://doi.org/10.1007/s10489-017-1078-x.

[137] A.K. Pandey, N.K. Goyal, Fault Prediction Model by Fuzzy Profile Development of Reliability Relevant Software Metrics, Int. J. Comput. Appl. 11 (2010) 975–8887. https://doi.org/10.5120/1584-2124.

[138] H.B. Yadav, D.K. Yadav, Early software reliability analysis using reliability relevant software metrics, Int. J. Syst. Assur. Eng. Manag. (2014). https://doi.org/10.1007/s13198-014-0325-3.

[139] H.B. Yadav, D.K. Yadav, A fuzzy logic based approach for phase-wise software defects prediction using software metrics, Inf. Softw. Technol. (2015) 44–57. https://doi.org/10.1016/j.infsof.2015.03.001.

[140] S. Kumar, P. Ranjan, A proposed methodology for phase wise software testing using soft computing, Int. J. Appl. Eng. Res. 12 (2017) 15855–15875.

[141] S. Chatterjee, B. Maji, H. Pham, A fuzzy rule-based generation algorithm in interval type-2 fuzzy logic system for fault prediction in the early phase of software development, J. Exp. Theor. Artif. Intell. 31 (2019) 369–391. https://doi.org/10.1080/0952813X.2018.1552315.

[142] P.S. Sandhu, S. Lata, D.K. Grewal, Neural Network Approach for Software Defect Prediction Based on Quantitative and Qualitative Factors, Int. J. Comput. Theory Eng. 4 (2012) 298–303.

[143] J. Wątróbski, J. Jankowski, P. Ziemba, A. Karczmarczyk, M. Zioło, Generalised framework for multi-criteria method selection, Omega (United Kingdom). 86 (2019) 107–124. https://doi.org/10.1016/j.omega.2018.07.004.

[144] W. Fu, T. Menzies, X. Shen, Tuning for software analytics: Is it really necessary?,

Inf. Softw. Technol. 76 (2016) 135–146. https://doi.org/10.1016/J.INFSOF.2016.04.017.

[145] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, K. Matsumoto, The Impact of Automated Parameter Optimization on Defect Prediction Models, IEEE Trans. Softw. Eng. 45 (2019) 683–711. https://doi.org/10.1109/TSE.2018.2794977.

[146] I. Sharma, M.P. Bano, A Combined Approach of Software Metrics and Software Fault Analysis to Estimate Software Reliability, IOSR J. Comput. Eng. 11 (2013) 01–14.

[147] P.S. Sandhu, M. Kaur, A. Kaur, A density based clustering approach for early detection of fault prone modules, in: ICEIE 2010 - 2010 Int. Conf. Electron. Inf. Eng. Proc., 2010: pp. 525–530. https://doi.org/10.1109/ICEIE.2010.5559753.

[148] A. Kaur, S. Gulati, A Framework for Analyzing Software Quality using Hierarchical Clustering, Int. J. Comput. Sci. Eng. 3 (2011) 854–861.

[149] V. Vashisht, M. Lal, G.S. Sureshchandar, S. Kamya, A Framework for Software Defect Prediction Using Neural Networks, J. Softw. Eng. Appl. (2015) 384–394. https://doi.org/10.9734/BJMCS/2016/26337.

[150] R. Bharathi, R. Selvarani, A framework for the estimation of OO software reliability using design complexity metrics, in: 2015 Int. Conf. Trends Autom. Commun. Comput. Technol., IEEE, 2015: pp. 1–7. https://doi.org/10.1109/ITACT.2015.7492648.

[151] A.K. Pandey, N.K. Goyal, A Fuzzy Model for Early Software Fault Prediction Using Process Maturity and Software Metrics, Int. J. Electron. Eng. 1 (2009) 239–245. https://doi.org/10.1007/978-81-322-1176-1.

[152] P.S. Sandhu, R. Goel, A.S. Brar, J. Kaur, S. Anand, A model for early prediction of faults in software systems, in: 2010 2nd Int. Conf. Comput. Autom. Eng., IEEE, 2010: pp. 281–285.

[153] P.S. Sandhu, S. Khullar, S. Singh, S.K. Bains, M. Kaur, G. Singh, A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm, World Acad. Sci. Eng. Technol. 48 (2010) 648–653.

[154] P.S. Sandhu, S. Khullar, S. Singh, S.K. Bains, M. Kaur, G. Singh, A Subtractive Clustering Based Approach for Early Prediction of Fault Proneness in Software

Modules, World Acad. Sci. Eng. Technol. 4 (2010) 1165–1169.

[155] M. Dhiauddin, M. Suffian, S. Ibrahim, A Systematic Approach to Predict System Testing Defects using Prior Phases Metrics for V-Model, Open Int. J. Informatics. 1 (2013) 1–17.

[156] S. Mohanta, G. Vinod, R. Mall, A technique for early prediction of software reliability based on design metrics, Int. J. Syst. Assur. Eng. Manag. 2 (2011) 261–281. https://doi.org/10.1007/s13198-011-0078-1.

[157] Y. Hong, J. Baik, I.Y. Ko, H.J. Choi, A value-added predictive defect type distribution model based on project characteristics, in: Proc. - 7th IEEE/ACIS Int. Conf. Comput. Inf. Sci. IEEE/ACIS ICIS 2008, Conjunction with 2nd IEEE/ACIS Int. Work. e-Activity, IEEE/ACIS IWEA 2008, 2008: pp. 469–474. https://doi.org/10.1109/ICIS.2008.36.

[158] M. Cartwright, M. Shepperd, An empirical investigation of an object-oriented software system, IEEE Trans. Softw. Eng. 26 (2000) 786–796. https://doi.org/10.1109/32.879814.

[159] K.S. Kumar, R.B. Misra, An Enhanced Model for Early Software Reliability Prediction Using Software Engineering Metrics, in: 2008 Second Int. Conf. Secur. Syst. Integr. Reliab. Improv., 2008: pp. 177–178. https://doi.org/10.1109/SSIRI.2008.32.

[160] Z.A. Rana, M.M. Awais, S. Shamail, An FIS for early detection of defect prone modules, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 5755 LNAI (2009) 144–153. https://doi.org/10.1007/978-3-642-04020-7_16.

[161] A. Nugroho, M.R. V Chaudron, E. Arisholm, Assessing UML design metrics for predicting fault-prone classes in a Java system, in: Proc. - Int. Conf. Softw. Eng., 2010: pp. 21–30. https://doi.org/10.1109/MSR.2010.5463285.

[162] Y. Jiang, B. Cukic, T. Menzies, N. Bartlow, Comparing Design and Code Metrics for Software Quality Prediction, in: 4th Int. Work. Predict. Model. Softw. Eng. PROMISE 2008, 2008: pp. 11–18.

[163] V. Vashisht, M. Lal, G.S. Sureshchandar, S. Kamya, Defect Prediction Framework Using Neural Networks for Software Enhancement Projects, Br. J. Math. Comput.

Sci. 16 (2016) 1–12. https://doi.org/10.9734/BJMCS/2016/26337.

[164] A.K. Pandey, N.K. Goyal, Early Fault Prediction Using Software Metrics and Process Maturity, in: Early Softw. Reliab. Predict., 2013. https://doi.org/10.1007/978-81-322-1176-1.

[165] R. Ratra, N.S. Randhawa, P. Kaur, Early Prediction of Fault Prone Modules using Clustering Based vs . Neural Network Approach in Software Systems, Reliab. Eng. 7109 (2011) 47–50.

[166] K.K. Mohan, A.K. Verma, A. Srividya, Early Qualitative Software Reliability Prediction and Risk Management in Process Centric Development Through a Soft Computing Technique, Int. J. Reliab. Qual. Saf. Eng. 16 (2009) 521–532. https://doi.org/10.1142/S0218539309003551.

[167] A. Kaur, P.S. Sandhu, A.S. Bra, Early Software Fault Prediction Using Real Time Defect Data, in: 2009 Second Int. Conf. Mach. Vis., 2009: pp. 242–245. https://doi.org/10.1109/ICMV.2009.54.

[168] B. Yang, L. Yao, H.-Z. Huang, Early Software Quality Prediction Based on a Fuzzy Neural Network Model, in: Third Int. Conf. Nat. Comput. (ICNC 2007), 2007: pp. 760–764. https://doi.org/10.1109/ICNC.2007.347.

[169] S. Yamada, Early-stage Software Product Quality Prediction Based on Process Measurement Data, in: Handb. Performability Eng., 2008: pp. 1227–1237.

[170] Y. Jiang, B. Cukic, T. Menzies, Fault Prediction using Early Lifecycle Data, in: 18th IEEE Int. Symp. Softw. Reliab. (ISSRE '07), 2007: pp. 237–246. https://doi.org/10.1109/ISSRE.2007.24.

[171] A.K. Pandey, N.K. Goyal, Multistage Model for Residual Fault Prediction, in: Early Softw. Reliab. Predict., 2013: pp. 117–130. https://doi.org/10.1007/978-81-322-1176-1.

[172] D.L. Gupta, A.K. Malviya, Observations on Fault Proneness Prediction Models of Object-Oriented System to Improve Software Quality, Int. J. Adv. Res. Comput. Sci. 2 (2011) 57–65. http://www.ijarcs.info/index.php/Ijarcs/article/view/367.

[173] T. Zimmermann, N. Nagappan, Predicting defects with program dependencies, in: 2009 3rd Int. Symp. Empir. Softw. Eng. Meas. ESEM 2009, 2009: pp. 435–438. https://doi.org/10.1109/ESEM.2009.5316024.

[174] R. Sehgal, D. Mehrotra, Predicting faults before testing phase using Halstead's metrics, Int. J. Softw. Eng. Its Appl. 9 (2015) 135–142. https://doi.org/10.14257/ijseia.2015.9.7.14.

[175] J. Ba, S. Wu, ProPRED: A probabilistic model for the prediction of residual defects, in: Proc. 2012 IEEE/ASME 8th IEEE/ASME Int. Conf. Mechatron. Embed. Syst. Appl., 2012: pp. 247–251. https://doi.org/10.1109/MESA.2012.6275569.

[176] S. Bibi, G. Tsoumakas, I. Stamelos, I. Vlahavas, Regression via Classification applied on software defect estimation, Expert Syst. Appl. 34 (2008) 2091–2101. https://doi.org/10.1016/j.eswa.2007.02.012.

[177] H. Euyseok, Software Fault-proneness Prediction using Random Forest, Int. J. Smart Home. 6 (2012) 147–152.

[178] C. Kumar, D.K. Yadav, Software Quality Modeling using Metrics of Early Artifacts, in: Conflu. 2013 4th Int. Conf. Next Gener. Inf. Technol. Summit, 2013: pp. 7–11. https://doi.org/10.1049/cp.2013.2285.

[179] W. Lee, J.K. Lee, J. Baik, Software Reliability Prediction for Open Source Software Adoption Systems Based on Early Lifecycle Measurements, in: 2011 IEEE 35th Annu. Comput. Softw. Appl. Conf., 2011: pp. 366–371. https://doi.org/10.1109/COMPSAC.2011.55.

[180] S. Wajahat, A. Rizvi, R.A. Khan, V.K. Singh, Software Reliability Prediction using Fuzzy Inference System: Early Stage Perspective, Int. J. Comput. Appl. 145 (2016) 16–23.

[181] M. Kläs, H. Nakao, F. Elberzhager, J. Münch, Support planning and controlling of early quality assurance by combining expert judgment and defect data- A case study, Empir. Softw. Eng. 15 (2010) 423–454. https://doi.org/10.1007/s10664-009-9112-1.

[182] P. Tomaszewski, L. Lundberg, H. Grahn, The accuracy of early fault prediction in modified code, in: Proc. Fifth Conf. Softw. Eng. Res. Pract. Sweden, 2005: pp. 57–63.

[183] K. El Emam, W. Melo, J.C. Machado, The prediction of faulty classes using object-oriented design metrics, J. Syst. Softw. 56 (2001) 63–75.

https://doi.org/10.1016/S0164-1212(00)00086-8.

[184] T.M. Khoshgoftaar, N. Seliya, Tree-based software quality estimation models for fault prediction, in: Proc. Eighth IEEE Symp. Softw. Metrics, 2002: pp. 203–214. https://doi.org/10.1109/METRIC.2002.1011339.

APPENDIX

APPENDIX 1 – Mapping references to ids of primary studies in [15]

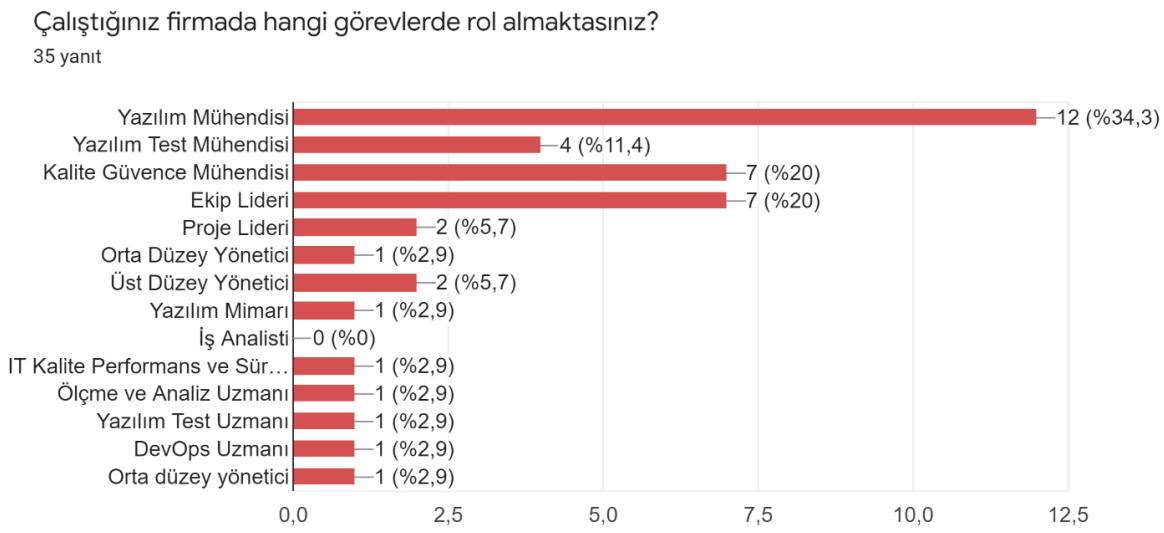
[S1]	[47]	S. Amasaki, Y. Takagi, O. Mizuno, T. Kikuno, A Bayesian belief network for assessing the likelihood of fault content, in: Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE, IEEE Computer Society, 2003: pp. 215–226.
[S2]	[146]	I. Sharma, M.P. Bano, A Combined Approach of Software Metrics and Software Fault Analysis to Estimate Software Reliability, IOSR J. Comput. Eng. 11 (2013) 01–14.
[S3]	[147]	P.S. Sandhu, M. Kaur, A. Kaur, A density based clustering approach for early detection of fault prone modules, in: ICEIE 2010 - 2010 Int. Conf. Electron. Inf. Eng. Proc., 2010: pp. 525–530.
[S4]	[148]	A. Kaur, S. Gulati, A Framework for Analyzing Software Quality using Hierarchical Clustering, Int. J. Comput. Sci. Eng. 3 (2011) 854–861.
[S5]	[149]	V. Vashisht, M. Lal, G.S. Sureshchandar, S. Kamya, A Framework for Software Defect Prediction Using Neural Networks, J. Softw. Eng. Appl. (2015) 384–394.
[S6]	[150]	R. Bharathi, R. Selvarani, A framework for the estimation of OO software reliability using design complexity metrics, in: 2015 Int. Conf. Trends Autom. Commun. Comput. Technol., IEEE, 2015: pp. 1–7.
[S7]	[139]	H.B. Yadav, D.K. Yadav, A fuzzy logic based approach for phase-wise software defects prediction using software metrics, Inf. Softw. Technol. (2015) 44–57.
[S8]	[151]	A.K. Pandey, N.K. Goyal, A Fuzzy Model for Early Software Fault Prediction Using Process Maturity and Software Metrics, Int. J. Electron. Eng. 1 (2009) 239-245.
[S9]	[152]	P.S. Sandhu, R. Goel, A.S. Brar, J. Kaur, S. Anand, A model for early prediction of faults in software systems, in: 2010 2nd Int. Conf. Comput. Autom. Eng., IEEE, 2010: pp. 281–285.
[S10]	[134]	S. Chatterjee, B. Maji, A new fuzzy rule based algorithm for estimating software faults in early phase of development, Soft Comput. 20 (2016) 4023–4035.
[S11]	[153]	P.S. Sandhu, S. Khullar, S. Singh, S.K. Bains, M. Kaur, G. Singh, A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm, World Acad. Sci. Eng. Technol. 48 (2010) 648–653.
[S12]	[154]	P.S. Sandhu, S. Khullar, S. Singh, S.K. Bains, M. Kaur, G. Singh, A Subtractive Clustering Based Approach for Early Prediction of Fault Proneness in Software Modules, World Acad. Sci. Eng. Technol. 4 (2010) 1165–1169.
[S13]	[155]	M. Dhiauddin, M. Suffian, S. Ibrahim, A Systematic Approach to Predict System Testing Defects using Prior Phases Metrics for V-Model, Open Int. J. Informatics. 1 (2013) 1–17.
[S14]	[156]	S. Mohanta, G. Vinod, R. Mall, A technique for early prediction of software reliability based on design metrics, Int. J. Syst. Assur. Eng. Manag. 2 (2011) 261–281.
[S15]	[157]	Y. Hong, J. Baik, I.Y. Ko, H.J. Choi, A value-added predictive defect type distribution model based on project characteristics, in: Proc. - 7th IEEE/ACIS Int. Conf. Comput.

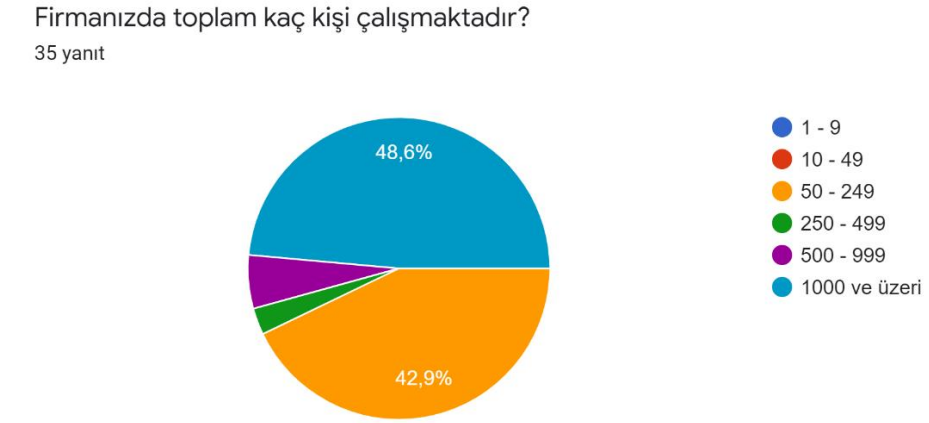
		Inf. Sci. IEEE/ACIS ICIS 2008, Conjunction with 2nd IEEE/ACIS Int. Work. e-Activity, IEEE/ACIS IWEA 2008, 2008: pp. 469–474.
[S16]	[158]	M. Cartwright, M. Shepperd, An empirical investigation of an object-oriented software system, IEEE Trans. Softw. Eng. 26 (2000) 786–796.
[S17]	[159]	K.S. Kumar, R.B. Misra, An Enhanced Model for Early Software Reliability Prediction Using Software Engineering Metrics, in: 2008 Second Int. Conf. Secur. Syst. Integr. Reliab. Improv., 2008: pp. 177–178.
[S18]	[160]	Z.A. Rana, M.M. Awais, S. Shamail, An FIS for early detection of defect prone modules, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 5755 LNAI (2009) 144–153.
[S19]	[161]	A. Nugroho, M.R. V Chaudron, E. Arisholm, Assessing UML design metrics for predicting fault-prone classes in a Java system, in: Proc. - Int. Conf. Softw. Eng., 2010: pp. 21–30.
[S20]	[130]	Y. Ma, S. Zhu, K. Qin, G. Luo, Combining the requirement information for software defect estimation in design time, Inf. Process. Lett. 114 (2014) 469–474.
[S21]	[162]	Y. Jiang, B. Cukic, T. Menzies, N. Bartlow, Comparing Design and Code Metrics for Software Quality Prediction, in: 4th Int. Work. Predict. Model. Softw. Eng. PROMISE 2008, 2008: pp. 11–18.
[S22]	[132]	P. Singh, S. Verma, Cross Project Software Fault Prediction at Design Phase, Int. J. Comput. Electr. Autom. Control Inf. Eng. 9 (2015) 800–805.
[S23]	[163]	V. Vashisht, M. Lal, G.S. Sureshchandar, S. Kamya, Defect Prediction Framework Using Neural Networks for Software Enhancement Projects, Br. J. Math. Comput. Sci. 16 (2016) 1–12.
[S24]	[164]	A.K. Pandey, N.K. Goyal, Early Fault Prediction Using Software Metrics and Process Maturity, in: Early Softw. Reliab. Predict., 2013.
[S25]	[165]	R. Ratra, N.S. Randhawa, P. Kaur, Early Prediction of Fault Prone Modules using Clustering Based vs . Neural Network Approach in Software Systems, Reliab. Eng. 7109 (2011) 47–50.
[S26]	[166]	K.K. Mohan, A.K. Verma, A. Srividya, Early Qualitative Software Reliability Prediction and Risk Management in Process Centric Development Through a Soft Computing Technique, Int. J. Reliab. Qual. Saf. Eng. 16 (2009) 521–532.
[S27]	[114]	D.K. Yadav, S.K. Chaturvedi, R.B. Misra, Early software defects prediction using fuzzy logic, Int. J. Performability Eng. 8 (2012) 399–408.
[S28]	[167]	A. Kaur, P.S. Sandhu, A.S. Bra, Early Software Fault Prediction Using Real Time Defect Data, in: 2009 Second Int. Conf. Mach. Vis., 2009: pp. 242–245.
[S29]	[168]	B. Yang, L. Yao, H.-Z. Huang, Early Software Quality Prediction Based on a Fuzzy Neural Network Model, in: Third Int. Conf. Nat. Comput. (ICNC 2007), 2007: pp. 760–764.
[S30]	[138]	H.B. Yadav, D.K. Yadav, Early software reliability analysis using reliability relevant software metrics, Int. J. Syst. Assur. Eng. Manag. (2014). https://doi.org/10.1007/s13198-014-0325-3.

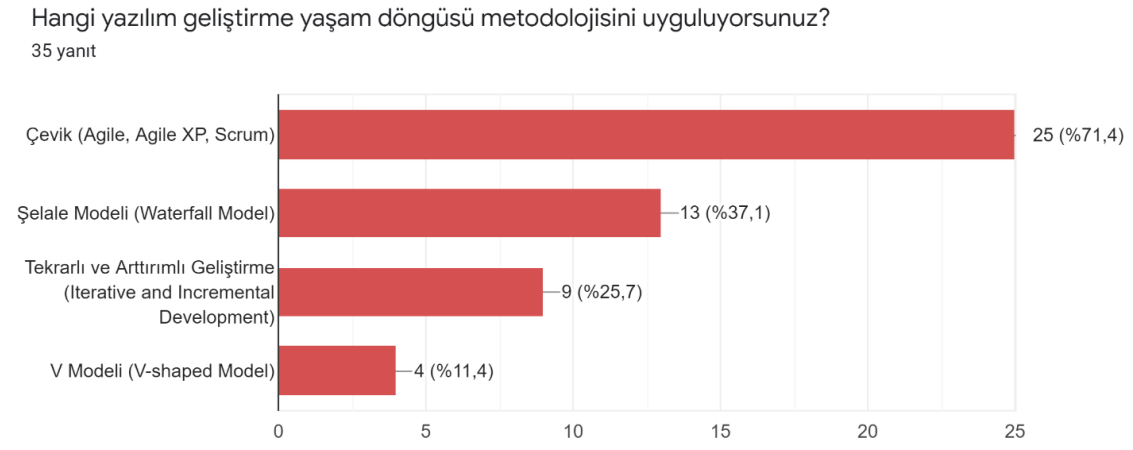
[S31]	[169]	S. Yamada, Early-stage Software Product Quality Prediction Based on Process Measurement Data, in: Handb. Performability Eng., 2008: pp. 1227–1237.
[S32]	[137]	A.K. Pandey, N.K. Goyal, Fault Prediction Model by Fuzzy Profile Development of Reliability Relevant Software Metrics, Int. J. Comput. Appl. 11 (2010) 975–8887.
[S33]	[170]	Y. Jiang, B. Cukic, T. Menzies, Fault Prediction using Early Lifecycle Data, in: 18th IEEE Int. Symp. Softw. Reliab. (ISSRE '07), 2007: pp. 237–246.
[S34]	[171]	A.K. Pandey, N.K. Goyal, Multistage Model for Residual Fault Prediction, in: Early Softw. Reliab. Predict., 2013: pp. 117–130.
[S35]	[142]	P.S. Sandhu, S. Lata, D.K. Grewal, Neural Network Approach for Software Defect Prediction Based on Quantitative and Qualitative Factors, Int. J. Comput. Theory Eng. 4 (2012) 298–303.
[S36]	[172]	D.L. Gupta, A.K. Malviya, Observations on Fault Proneness Prediction Models of Object-Oriented System to Improve Software Quality, Int. J. Adv. Res. Comput. Sci. 2 (2011) 57–65.
[S37]	[21]	N. Fenton, M. Neil, W. Marsh, P. Hearty, Ł. Radliński, P. Krause, On the effectiveness of early life cycle defect prediction with Bayesian nets, Empir. Softw. Eng. 13 (2008) 499–537.
[S38]	[173]	T. Zimmermann, N. Nagappan, Predicting defects with program dependencies, in: 2009 3rd Int. Symp. Empir. Softw. Eng. Meas. ESEM 2009, 2009: pp. 435–438.
[S39]	[174]	R. Sehgal, D. Mehrotra, Predicting faults before testing phase using Halstead's metrics, Int. J. Softw. Eng. Its Appl. 9 (2015) 135–142.
[S40]	[175]	J. Ba, S. Wu, ProPRED: A probabilistic model for the prediction of residual defects, in: Proc. 2012 IEEE/ASME 8th IEEE/ASME Int. Conf. Mechatron. Embed. Syst. Appl., 2012: pp. 247–251.
[S41]	[176]	S. Bibi, G. Tsoumakas, I. Stamelos, I. Vlahavas, Regression via Classification applied on software defect estimation, Expert Syst. Appl. 34 (2008) 2091–2101.
[S42]	[121]	Z. Jiang, Y., Lin, J., Cukic, B., Lin, S., & Hu, Replacing Code Metrics in Software Fault Prediction with Early Life Cycle Metrics, in: Third Int. Conf. Inf. Sci. Technol., 2013: pp. 516–523.
[S43]	[135]	C. Kumar, D.K. Yadav, Software defects estimation using metrics of early phases of software development life cycle, Int. J. Syst. Assur. Eng. Manag. (2014).
[S44]	[131]	P. Singh, S. Verma, O.P. Vyas, Software fault prediction at design phase, J. Electr. Eng. Technol. 9 (2014) 1739–1745.
[S45]	[177]	H. Euyseok, Software Fault-proneness Prediction using Random Forest, Int. J. Smart Home. 6 (2012) 147–152.
[S46]	[178]	C. Kumar, D.K. Yadav, Software Quality Modeling using Metrics of Early Artifacts, in: Conflu. 2013 4th Int. Conf. Next Gener. Inf. Technol. Summit, 2013: pp. 7–11.
[S47]	[179]	W. Lee, J.K. Lee, J. Baik, Software Reliability Prediction for Open Source Software Adoption Systems Based on Early Lifecycle Measurements, in: 2011 IEEE 35th Annu. Comput. Softw. Appl. Conf., 2011: pp. 366–371.

[S48]	[180]	S. Wajahat, A. Rizvi, R.A. Khan, V.K. Singh, Software Reliability Prediction using Fuzzy Inference System: Early Stage Perspective, Int. J. Comput. Appl. 145 (2016) 16–23.
[S49]	[181]	M. Kläs, H. Nakao, F. Elberzhager, J. Münch, Support planning and controlling of early quality assurance by combining expert judgment and defect data- A case study, Empir. Softw. Eng. 15 (2010) 423–454.
[S50]	[182]	P. Tomaszewski, L. Lundberg, H. Grahn, The accuracy of early fault prediction in modified code, in: Proc. Fifth Conf. Softw. Eng. Res. Pract. Sweden, 2005: pp. 57–63.
[S51]	[183]	K. El Emam, W. Melo, J.C. Machado, The prediction of faulty classes using object-oriented design metrics, J. Syst. Softw. 56 (2001) 63–75.
[S52]	[184]	T.M. Khoshgoftaar, N. Seliya, Tree-based software quality estimation models for fault prediction, in: Proc. Eighth IEEE Symp. Softw. Metrics, 2002: pp. 203–214.

APPENDIX 2 – Results of “Survey Study on SDP from Industry in Turkey”

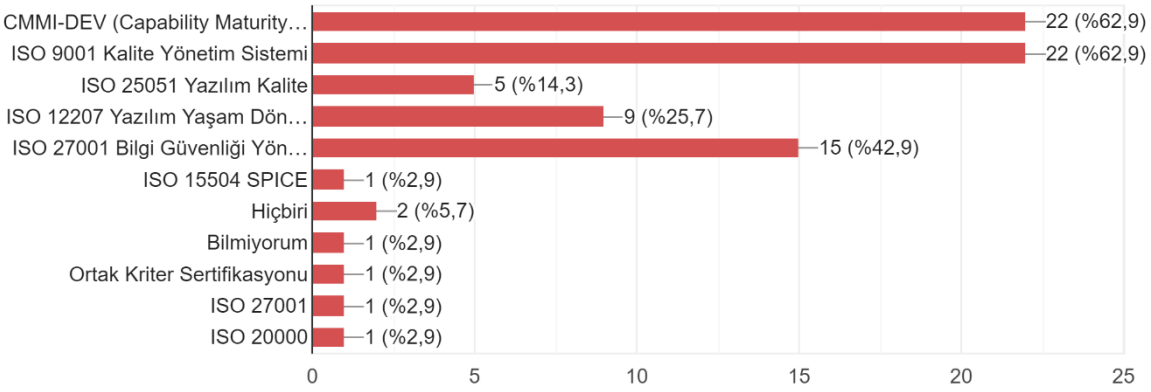






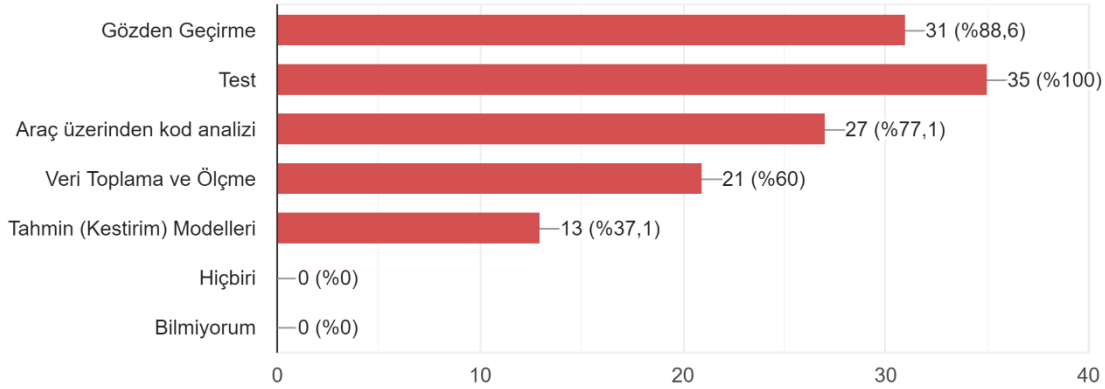
Firmanızda hangi kalite modellerini / standartlarını kullanıyorsunuz?

35 yanıt



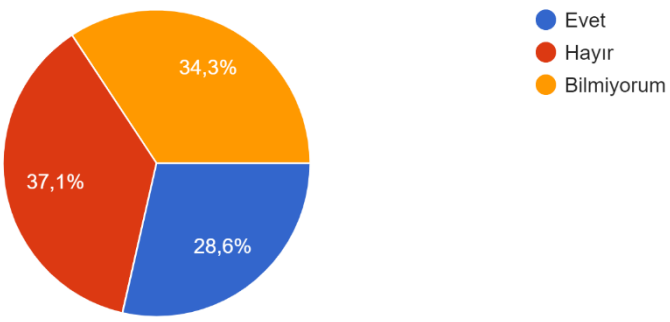
Firmanızda "yazılım kalite yönetimi" kapsamında hangi aktiviteler uygulanmaktadır?

35 yanıt



Firmanızda "yazılım hata tahmini" yapılmakta mıdır?

35 yanıt

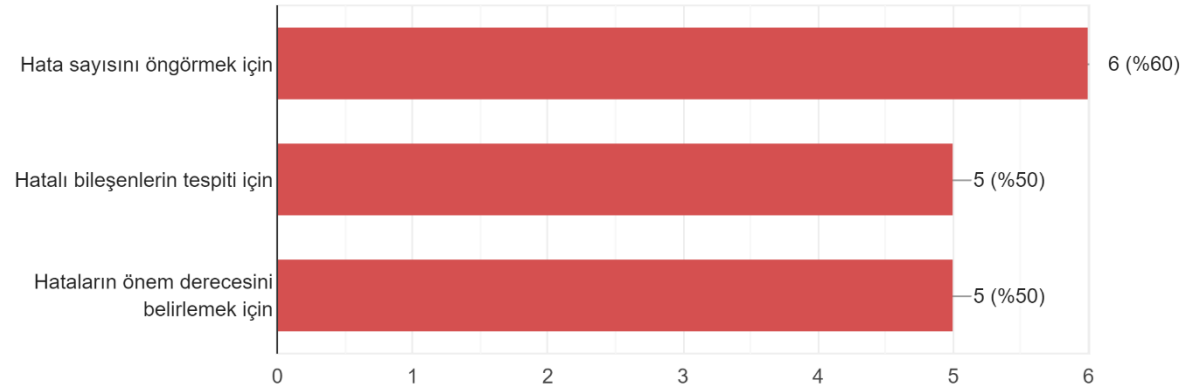


Yazılım hata tahminini nasıl işletiyorsunuz?

1	Statik Kod analizi yaparak, sürekli entegrasyon faaliyeti yapıp, çıkan hatalar analiz ediyor. Ayrıca hata takip sistemi ölçümleri alınıyor.
2	Poje başlangıcında tamamlanmış benzer projelerden kestirimde bulunarak
3	Her Sprint iterasyonunda ortaya çıkan geliştirme hataları ile iterasyon süresince sahadan gelen hatalar birleştirilerek kök nedenlerine göre sınıflandırılmaktadır. Geçmiş sprint içindeki özellik ekleme puan toplamı(velocity) ile iterasyon hataları koreledilerek yapılan sprint planlamasında sonraki sprintde beklenen hata oranı hesaplanmaktadır. Aynı zamanda bu velocity ve hata büyüklüğü toplamları bölünerek ekibin özellik ekleme hata oluşturma katsayısı hesaplanmakta ve bir kalite metriği olarak ekiplerin bu katsayıyı büyütmesi(daha fazla özellik puanı daha az hata) beklenmektedir.
4	Benzer proje verileri toplanmakta ve geliştirilicek olan yeni yazılım projesinde potansiyel hataya açık kesimlerin tahmini istatistiksel yöntemlerle yapılmaktadır.
5	Geçmiş verilere dayalı istatistiksel yöntemler kullanılarak oluşturulmuş hata tahmin modeli ile her sürümden önce ekiplerin ilgili sürümde kaç hata ortaya çıkacağını öngörebilmeleri sağlanıyor.
6	Code coverage, coupling/cohesion, Automated test tools vb. araçlar ve yöntemler yardımı ile test fazından önce hata tahmini ve ürün olgunluğu kontrolü yapılmaktadır.
7	Matematiksel bir model ile kontrol edilebilir değişkenlerden bir sonraki sürüm hata sayısı tahmini
8	Yazılımların yeniden kullanımının fazlaca olduğu projelerde (ürün hattı gibi) ve regresyon testi işletilen projelerde yazılım güvenilirlik tahmin modelleri uygulamaktayız.
9	Yalnızca öngörü tahmin ve sözlü, geçmiş deneyime binaen
10	Geçmiş deneyime binaen

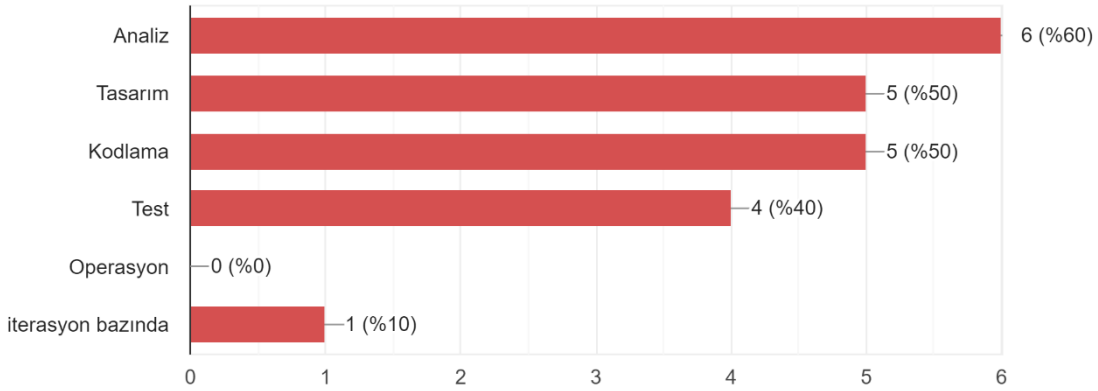
Yazılım hata tahminini hangi amaçla yapıyorsunuz?

10 yanıt



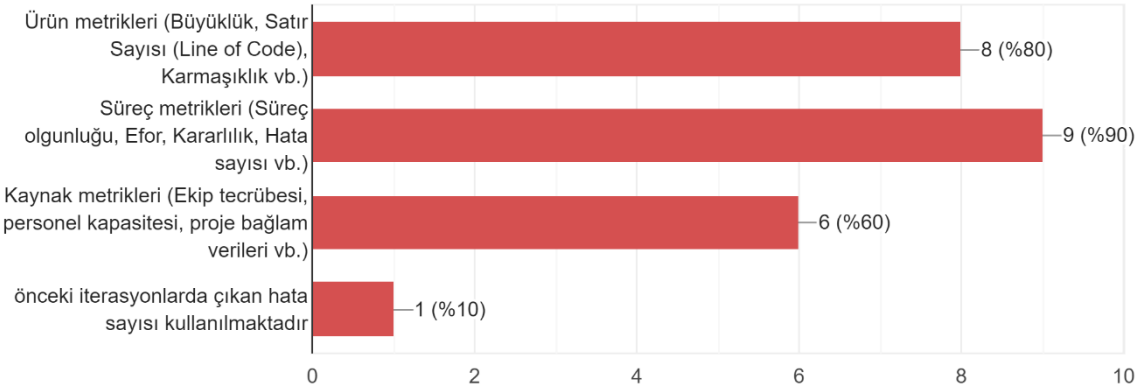
Yazılım geliştirme yaşam döngüsünün hangi aşamalarında hata tahmini yapıyorsunuz?

10 yanıt



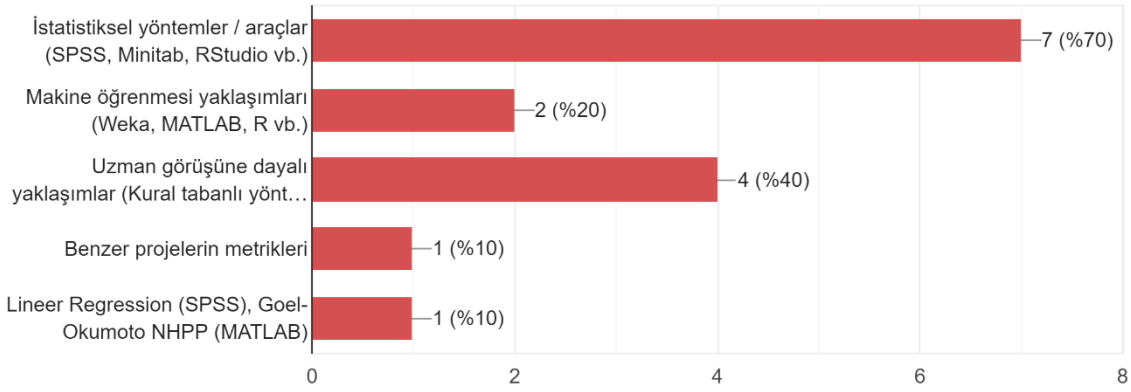
Yazılım hata tahmini için hangi metrikleri (ölçümleri) kullanıyorsunuz?

10 yanıt



Yazılım hata tahmini modelini oluşturmak için hangi yaklaşım(lar)ı ve/veya araçları kullanıyorsunuz?

10 yanıt



Sizce firmanızdaki yazılım hata tahmini uygulamalarının faydaları veya avantajları nelerdir?

1	Kaliteli ürün oluşturmak için oldukça faydalıdır.
2	Çıkabilecek hataların öngörülmesi projenin uzama riskini de ölçmeyi sağlıyor.
3	Öncelikle bir performans metriğ oluşturarak ekiplerin giderek az hata yapması için motivasyon sağlanmaktadır(hata tahminin aşılması aşılmaması). Aynı zamanda release planlama ve sprint commitmentlarının belirlenmesinde olası hata metriği riski öngörülebilir kılmaktadır.
4	Yeni başlanacak yazılım geliştirme projelerinde, gereksinimlerin belirlenmesi sürecinde ve ekiplerin oluşturulması aşamasında, önceki projelerden topladığımız metriklerle korelasyon yapıyoruz. Böylelikle olası hatalı bileşenleri önceden kestirip, bu noktalarda daha fazla kaynak ayırabiliyoruz.
5	Sürüm bazlı hata tahmini yapılmaktadır. Bu sayede ekipler yeni sürüm almadan önce, geçmiş sürüm verilerine dayalı olarak bir sonraki sürümde çıkacak hata sayısını öngörebiliyor ve buna dayalı olarak test ve gözden geçirme gibi kalite faaliyetlerine ağırlık vererek hata sayısını indirgeyebiliyor.
6	test / kabul fazından önce hata tahmini ve ürün olgunluğu kontrolü yapılması
7	Bir sonraki sürümde çıkacak hata sayısını kontrol altına almak
8	İterasyonlar bazında ortaya çıkabilecek hataların sayısına göre, sonraki iterasyonların ön kabulünden önce koşturulan testlerin sayısında ve kapsamında artış / azalışa gidebiliyoruz.
9	Faydası olduğunu düşünmüyorum.
10	Ürünün kalitesine doğrudan etkisi olmaktadır. Zamandan kazanım sağlamaktadır.

Sizce firmanızdaki yazılım hata tahmini uygulamalarının zorlukları veya dezavantajları nelerdir?

1	Dinamik sürüm üretme ve takvim sıkışıklığının tahminlemenin etkin yapılmasını önlemesi
2	Hata sayısı kestirimi her zaman sağlıklı yapılamıyor.
3	Hata sayısı ölçülebilir olsada büyüklük değerlendirmesi uzman tahminine dayılı olmaktadır. Bu nedenle kestirimlerde hatalı değerlendirme sonucu beklentiyle oluşan sonuç arasında ciddi farklılaşma olabilmektedir.
4	Her projenin dinamikleri birbirinden farklı olabiliyor, bu noktada benzerlikleri / farklılıkları çıkarmak kolay olmuyor.
5	Modelin belirli aralıklarla yeni verilerle birlikte kalibre edilmesi ve güncel tutulması gerekiyor. Süreçler tarafından tahmin modelinin kullanımı zorunlu kılınmadığı için ekipler arasında kullanımının yaygınlaştırılması ve faydaları konusunda farkındalığın arttırılması zorlayıcı olabiliyor.
6	Maliyet
7	Ekipler tarafından kullanılmaması
8	Daha fazla ölçüm içeren daha başarılı performans veren yöntemlere (makine öğrenmesi gibi) ihtiyacımız bulunmaktadır. Tasarımları konusunda know-how'ımız olmadığı için geliştiremiyoruz. Kullandığımız hata tahmin modelleri aslında yetersiz kalmaktadır, aşağıdaki bilgileri de ekleyerek daha geniş kapsamlı bir tahmin modeli ihtiyacımız vardır: iterasyon kapsamında eklenecek / silinecek kod satır sayısı yeni gereksinimlerin sayısı gereksinimlerim zorluk seviyesi gibi
9	Yorumsuz
10	Hata tahmini yapacak seviyede insan kaynağına sahip olmak zorlukları arasında yer almaktadır. Her projenin farklı dinamikleri olduğu için yapılan tahminlerde sapmalar olabilir.

Sizce firmanızda yazılım hata tahmini neden yapılmıyor?

1	Çalıştığım firmada geliştirilen yazılımların ortak yönleri ile ilgili bir çalışma yapılmamış ve bir envanter oluşturulmamış. Projeler yazılım ve donanım içeren sistem entegrasyon projeleri olduğundan hata tahmininin öncelikli görülmediğini düşünüyorum.
2	Bu durum firmanın ve ilgili süreçlerinin olgunluk seviyesiyle ilişkili. Olgunluk seviyesi arttıkça sadece düzeltici değil önleyici faaliyetler de artacaktır. Firmamızda şu an daha önce analistler tarafından yapılan test aktivitelerinin alanında uzman test mühendisleri tarafından yapılmasını önceliklendirmiş durumdayız. Bu çalışmanın olgunlaşmasıyla uzmanlığı test ve kalite olan ekipler önleyici tahminleme çalışmalarına odaklanabileceklerdir. Özetle, kaynak, önceliklendirme, sürece hazır olgunluk seviyesi sebepleri oluşturuyor.
3	Projelerin başında çok hızlı bir şekilde kod geliştirme sürecine geçilmek isteniyor çıktıların hızlı olması açısından. Bu nedenle diğer işlere vakit ayrılmıyor / önem verilmiyor.
4	Bilinmediğinden ve bunu destekleyecek ölçüm alt yapısının olmamasından
5	Firmamızda projeler kısıtlı zamanda düşük maliyetle yapılmaya çalışılıyor. Yazılım hata tahmini yapmak için zaman ve bütçe ayrılmıyor.
6	Uygulama şekli ve faydaları yeterince bilinmiyor,
7	Kullanılması planlanıyordur ancak farklı teknolojiler ve yazılım dilleri kullanıldığından dolayı maliyeti göz önünde bulundurulduğundan önceliklendirilmemiştir.
8	Yazılım hata tahmini yapmak için yetişmiş personel bulunmadığından yapılmıyor
9	Zaman kısıtından dolayı
10	İhtiyaç duyulmuyor

Firmanızda yazılım hata tahmini yapılıyor olsaydı sizce ne gibi faydaları olurdu?

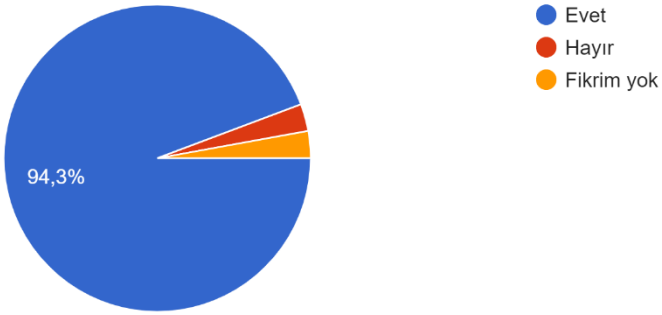
1	Çalışmanın başında projelerin sınıflandırılması ve büyüklüklerinin nasıl ölçülmesi gerektiği ortaya çıkardı. Farkındalık ve yeniden kullanılabilirlik artardı. Testler daha verimli geliştirilebilirdi.
2	Test süreçlerinin kalitesi artar, kaynak ve zaman yönetimi daha verimli yapılırdır.
3	Hızlı geliştirme aktiviteleri nedeniyle fazla sayıda bug çıkıyor. Sprintlerin 3'te 1'ini bug çözmek için ayırıyoruz ve bu süreç yeni bugların inject olmasına da neden olabiliyor. Kodlama öncesi aşamalarda hata tahmini yapılıyor olsaydı (tasarım dökümanlarındaki hatalar çoğunlukla yazılım hatası olarak doğuyor) daha az rework çıkardı diye düşünüyorum.
4	Geliştirmeyi buna göre yönlendirirdik
5	Üründen yapılan hatalı ya da eksik geri bildirimlerin azalması sağlanabilirdi
6	Geliştirilen yazılım daha kaliteli üretilebilirdi.
7	Ekibimde yer alan işgücünü yazılımın hatalı kısımları üzerinde yoğunlaştırır, o kısımlarda olan test aktivitelerini arttırırdım.
8	Projenin hem yazılım hem de test evresindeki süre kesinlikle kolaylaşacaktır.
9	Daha az hata oluşacağı için müşteri memnuniyeti ve ürün kalitesi artar, maliyet düşerdi.
10	Gerçek bir metrik bulunsaydı, yazılım geliştirmede önlem alırdık, koku araştırır daha kaliteli yazılım cikaririrdik. Bununla ilgili analizler sağlıklı yurutulurse kök nedeni bulup geliştirme gereksinim test süreçlerine girdi saglardik.
11	Hataları daha erken tespit etme

Firmanızda yazılım hata tahmini yapılıyor olsaydı sizce ne gibi zorlukları olurdu?

1	Tahmin modelini oluşturmak zor olurdu. Projelerin benzer nitelikte olduğunu belirleyen kriterlerin tanımlanması zor olabilirdi. Matris yapıda bir firmada çalıştığım için yazılım ve test mühendisliği ekiplerinin iş yükü artabilirdi.
2	Yeterli verinin oluşması, personel sirkülasyonunun modele etkisi, değişen altyapı vb.
3	Deadline'ları yakalamak için çok hızlı geliştirme yapılıyor, bu süreçte hata tahmini uygulamak bizi yavaşlatabilirdi.
4	Tahminleme için gerekli veriyi toplama
5	Projelere ekstra maliyet getirirdi. Bu durum özellikle Kamu projelerinde firmamıza dezavantaj oluştururdu.
6	Bu zamana kadar herhangi bir tahmin yapılmadığı için (ilk kez yapılacağı için) bilinmezlikler olurdu. Gereken verileri toplamak için zaman ve kaynak gerekebilirdi.
7	Kullanılan yönteme bağlı olarak yazılımcılar tek düze kod geliştirmeye yönlenmek zorunda kalabilirler.
8	Hata tahmini yapmak için hangi metriklerinin kullanılacağı ve bunların nasıl doğru bir şekilde toplanacağı en zorlayıcı sorunlar olurdu
9	Doğru bir tahminlere yapmak zor. Genelde insanlar bu modellerin başarısına inanıyorlar. İkna edilmeleri lazım. Hata tahmini için çok fazla değişken mevcut. Bir projenin kimi diğeriyle karşılaştırabilirsiniz, çünkü projelerin doğası farkli-kullanici sayısı, kullanım sıklığı vs. Eğer doğru tahminlere yapılırsa normalden sapma olmadığı sürece kullanılmaz. Hata tahmini normalden saparsa da alarm vermek yerine önce kalite olarak manuel doğruluğu değerlendirilmeli sonra ekiple çalışma yapılmalıdır. Belki o sürümde ekstra bir durum olmuştur. Acil durum sinyali vermektense doğruluğu inandırıcılığı oluşana kadar ölçülmelidir.
10	Veri toplamak

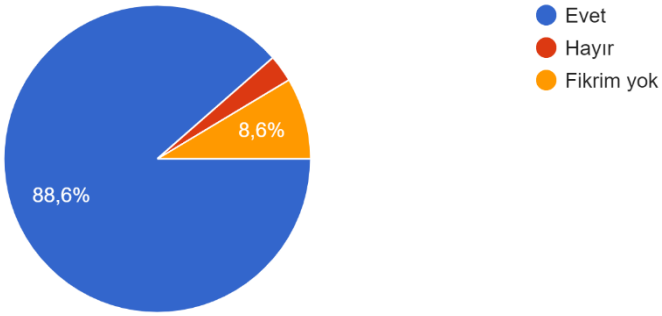
Sizce, yazılım geliştirmenin ilk aşamalarından itibaren yazılım hata tahmini sürecinin nasıl işletileceği konusunda bir kılavuz olsaydı, faydalı olur muydu?

35 yanıt



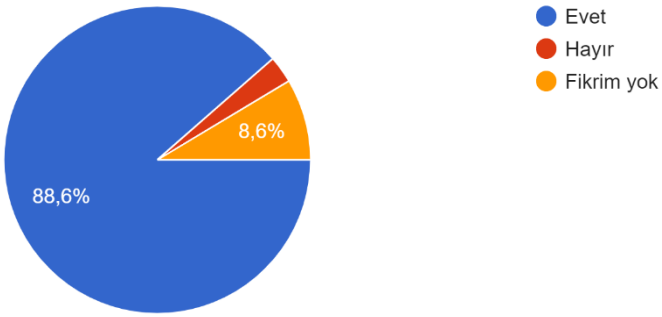
Hata tahmin yönteminin seçilmesi için kılavuzluğa ihtiyaç var mı?

35 yanıt



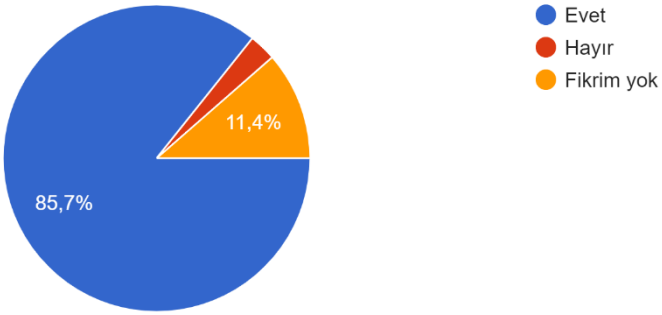
Hata tahmin modelinin girdi ve çıktılarının belirlenmesi için kılavuzluğa ihtiyaç var mı?

35 yanıt



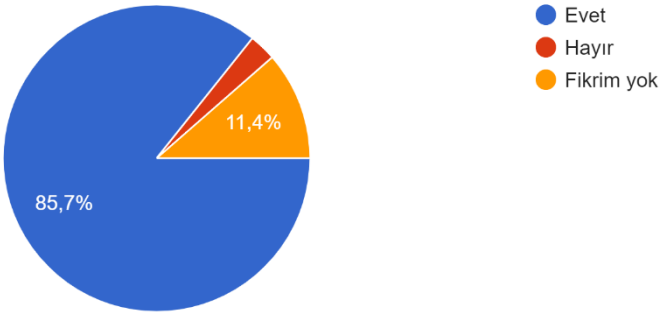
Hata tahmin modelinin oluşturulması için kılavuzluğa ihtiyaç var mı?

35 yanıt



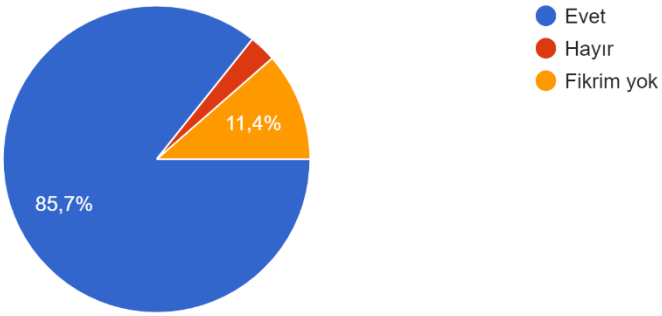
Hata tahmininin nasıl yapılacağı konusunda kılavuzluğa ihtiyaç var mı?

35 yanıt



Hata tahmin başarımının nasıl değerlendirileceği konusunda kılavuzluğa ihtiyaç var mı?

35 yanıt



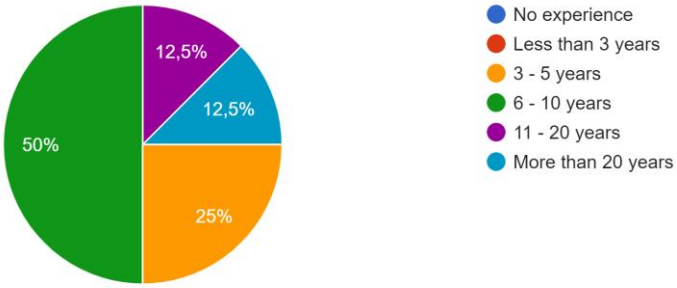
Sizce yazılım geliştirme sürecinin ilk aşamalarından itibaren yazılım hata tahmini için önerilecek bir kılavuzda yukarıdaki hususlara ek hangi konular ele alınabilir?

1	Farklı nitelikteki projeler için farklı modeller önerebilir. Ayrıca modelin yanısıra modelin içeriğinin de hangi roller tarafından işletileceğini de içerebilir.
2	Sektörel bazdaki farklılıklar. Örneğin savunma sanayine özel yazılımlardaki yöntem ile bankacılık ve finans sektöründeki yazılım modellerinde farklılık var mı?
3	Özellikle yazılım hata tahmini öncesinde kullanılabilecek verilerin ve yöntemlerin belirlenmesi amacıyla, o projenin dinamiklerine uygun bir yönlendirme işe yarayacaktır. Eski projelerden topladığımız verileri ortaklaştırabileceğimiz ve sonrası için kullanabileceğimiz bir yapı işimize yarayabilir. Kullandığımız istatistiksel yöntemler bazen yeterli olmuyor, ek olarak farklı metrikleri (ekip dinamiği, paydaşların öncelikleri, projelerdeki belirsizlikler vb) de işleyebileceğimiz bir yöntem seçimi işimize yarayacaktır.
4	Yukarıdaki bilgilerin yeterli olduğunu düşünüyorum.
5	Farklı modeller arasından hangi tür projelerde hangi modelin kullanılacağına dair bir model seçim kılavuzu faydalı olabilir.
6	En basit uygulanabilecek ve en iyi sonucu verebilecek bir tahmin modelini en kısa sürede nasıl oluşturabilirim sorusunun cevabı çok faydalı olurdu
7	Elimizde bulunan verilerle kullanabileceğimiz en ideal model hangisi, bizim için en kritik olan bu
8	Hangi sektörlerde geliştirilen yazılımlarda bu süreç daha faydalı olmaktadır. Geliştirilmek istenener her uygulama için bu yöntem sürdürülebilir mi? Varsa yanlış uygulandığı takdir de ne tür sorunlara neden olmaktadır?
9	Bu tarz tahmin süreçlerinin ilgili kuruma olan faydaları ve bu süreçlerin ek yük değil avantaj sağlayabileceğinin vurgulandığı konular ele alınabilir.
10	Metrikler açıkça ifade edilmeli ve ekip lideri veya Pylere inandırıcılığı sağlanması için örnek senaryolar üretilmelidir. Ayrıca dinamikleri sürekli değişen küçük projelerde uygulanmamali. Daha kurumsal büyük projelerde uygulanabilir.

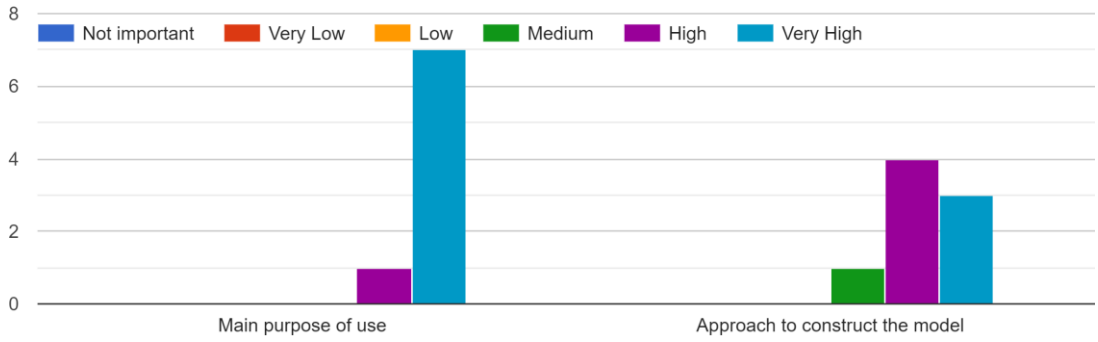
APPENDIX 3 – Results of “Expert Opinion Study on Identifying and Ranking the Criteria”

How many years of experience do you have in the area of SDP?

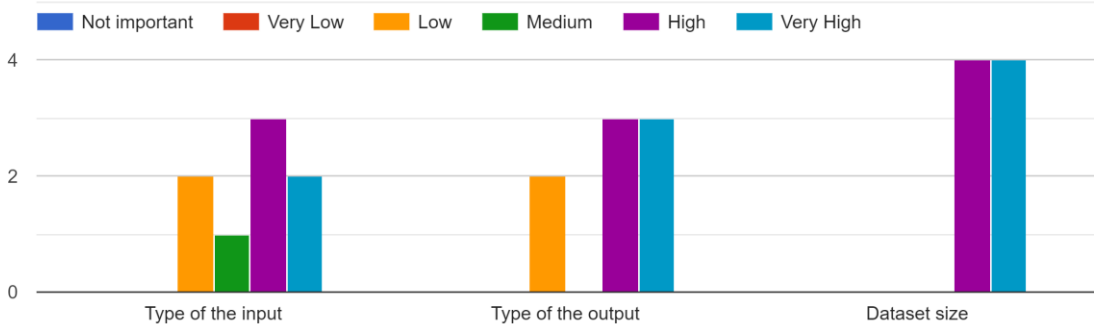
8 yanıt



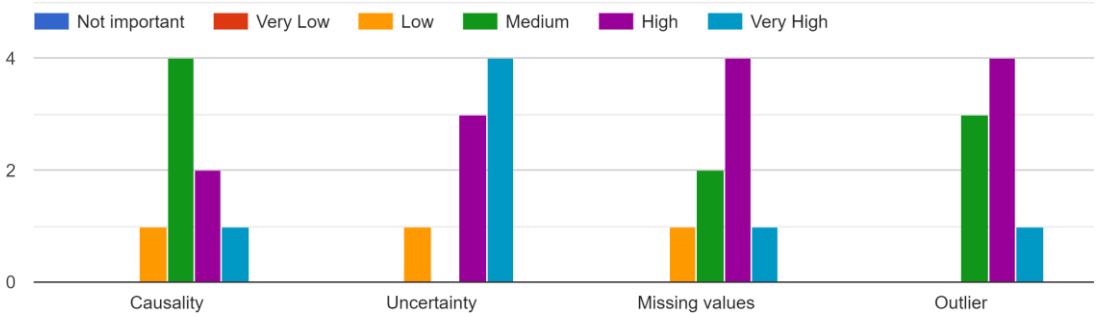
What is the importance of the below criteria about "model construction" with respect to SDP method selection?



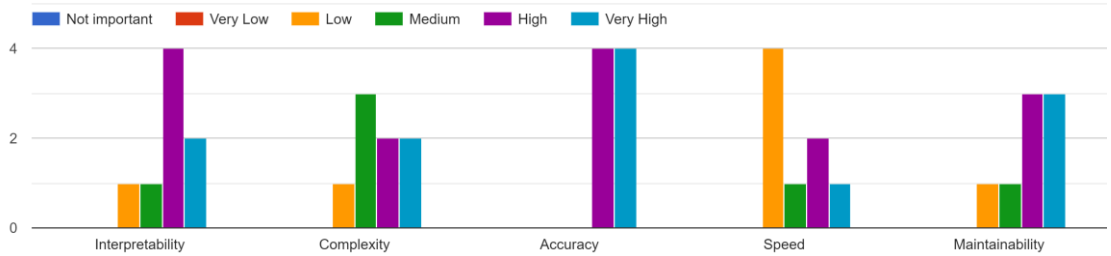
What is the importance of the below criteria about "data characteristics" with respect to SDP method selection?



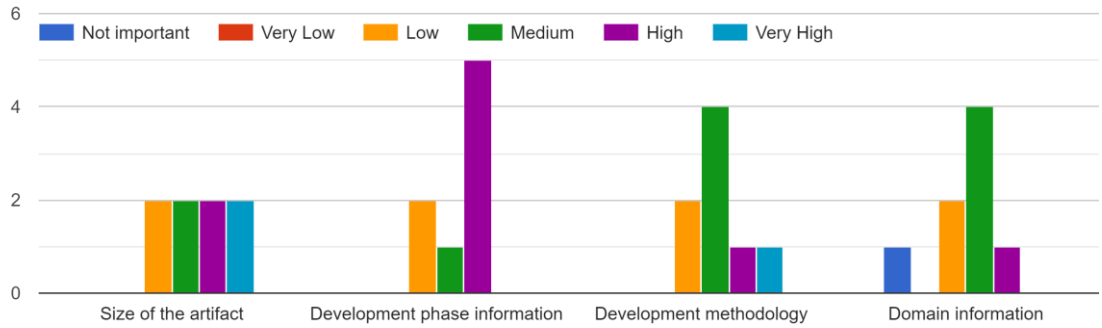
What is the importance of the below criteria about "data quality" with respect to SDP method selection?



What is the importance of the below criteria about "method characteristics" with respect to SDP method selection?

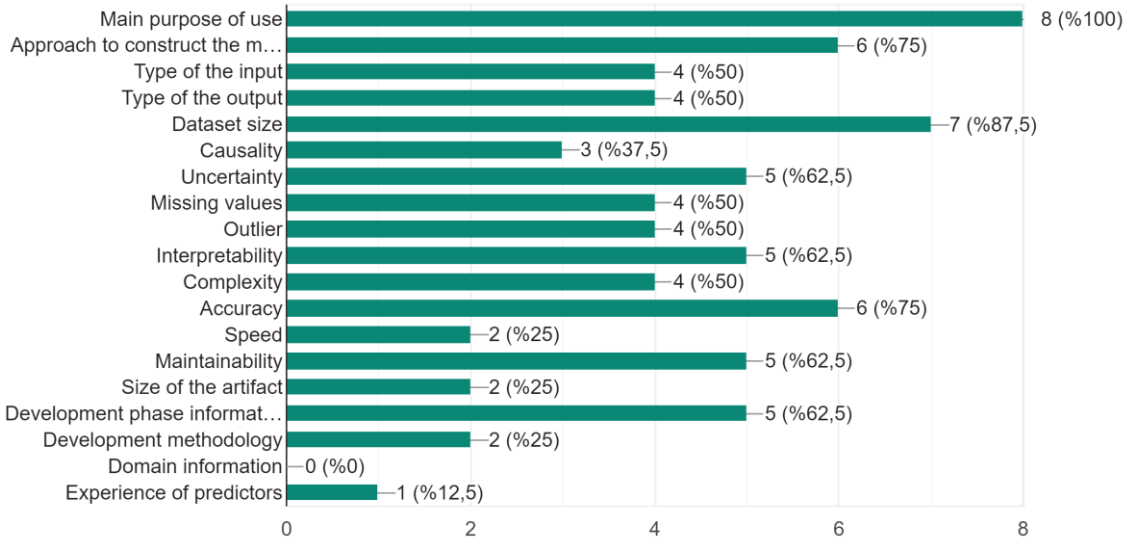


What is the importance of the below criteria about "project context" with respect to SDP method selection?



Select the criteria that you think as important for SDP in the "early phases" of the software development.

8 yanıt



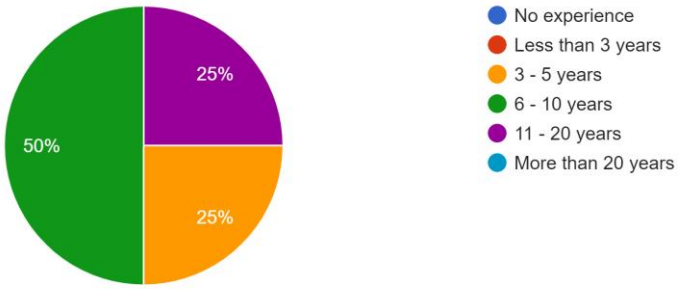
APPENDIX 4 – Results of "Expert Opinion Study for the Evaluation of Alternatives against Criteria"

How many years of experience do you have in the area of SDP?

4 yanıt

How many years of experience do you have in building / using of the prediction methods?

4 yanıt



Expert#1	I am familiar with this method	No experience	I have expertise in this method	No experience	I have expertise in this method	I have expertise in this method	I am familiar with this method	I am familiar with this method
	ANN	BBN	DT	FIS	LinR	LogR	NB	SVM
Dataset size	Large	Large	Small	Medium	Medium	Medium	Small	Large
Causality	Average	Average	Average	Low	Low	Average	Average	Average
Uncertainty	Average	Average	Average	Low	Average	Average	High	Average
Missing Values	High	High	High	Average	High	High	High	High
Outlier	High	Average	Very Low	Average	Average	High	High	High
Interpretability	Very Low	Average	Very High	Average	Very High	Very High	Very High	Average
Complexity	High	High	Very Low	High	Very Low	Very Low	Very Low	High
Performance	Average	High	High	Average	Average	High	High	High
Speed	Low	Average	Very High	Average	Very High	Very High	Very High	Average
Maintainability	Low	Average	High	Low	High	High	High	Average
Size	No	Yes	Yes	No	No	No	Yes	Yes
Development methodology	No	No	No	Yes	No	No	No	No
Development phase	Yes	Yes	No	No		No	No	No
Domain	Yes	Yes	Yes	Yes	No	No	No	No

Expert#2	I have expertise in this method	No experience	No experience	I have expertise in this method	No experience	No experience	I am familiar with this method	I am familiar with this method
	ANN	BBN	DT	FIS	LinR	LogR	NB	SVM
Dataset size	Large			No data required			Medium	Medium
Causality	Average			Very High				
Uncertainty	High			Low				
Missing Values	Average			Low				
Outlier	High			Low				
Interpretability	Very Low		High	Very High				
Complexity	Very High			Very Low			Low	Average
Performance	Very High			High			High	Average
Speed	Low			High			High	
Maintainability	Average			Very High			High	
Size	Yes			No			Yes	Yes
Development methodology	Yes			No			Yes	Yes
Development phase	Yes			No			Yes	Yes
Domain	No			No			No	No

Expert#3	I am familiar with this method	I have expertise in this method	I am familiar with this method	No experience	I have expertise in this method	I have expertise in this method	I have expertise in this method	I am familiar with this method
	ANN	BBN	DT	FIS	LinR	LogR	NB	SVM
Dataset size	Very Large	No data required	No data required	Medium	Medium	Medium	Medium	Large
Causality	Very Low	Very High	Very Low	Very Low	Very Low	Very Low	Very Low	Very Low
Uncertainty	Low	Very High	Low	Average	Low	Low	Very High	Low
Missing Values	Average	Very High	Average	Average	Low	Low	Very High	Low
Outlier	High	High	High		Low	Low	Average	Average
Interpretability	Very Low	Very High	Very High		High	High	High	Very Low
Complexity	High	Average	Low		Very Low	Very Low	Very Low	Low
Performance	High	High	High	Very Low	High	High	High	Average
Speed	High	Average	Very High	Average	Very High	Very High	Very High	High
Maintainability	Very Low	Very High	Very Low	Very Low	Average	Average	Average	Very Low
Size	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Development methodology	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Development phase	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Domain	No	Yes	No	Yes	No	No	No	No

Expert#4	I am familiar with this method	I am familiar with this method	I am familiar with this method	I am familiar with this method	I am familiar with this method	I am familiar with this method	I am familiar with this method	I am familiar with this method
	ANN	BBN	DT	FIS	LinR	LogR	NB	SVM
Dataset size	Large	No data required	Medium	No data required	Small	Small	Small	Large
Causality	Average	Very High	Average	Very High	Low	Low	Very Low	Low
Uncertainty	Average	Very High	Average	Average	Low	Low	High	Average
Missing Values	Average	Average	Low	High	Low	Low	High	Average
Outlier	High	High	Low	Average	Average	Average	Low	Low
Interpretability	Very Low	High	Very High	Very High	High	High	Average	Low
Complexity	Very High	High	Low	Low	Very Low	Very Low	Average	High
Performance	Very High	Average	High	Average	High	High	High	Average
Speed	Very Low	Very High	Very High	Very Low	Average	Average	Very High	Average
Maintainability	High	Average	Low	High	High	High	Average	Low
Size	No	Yes	Yes	Yes	No	No	No	Yes
Development methodology	No	Yes	Yes	Yes	No	No	No	No
Development phase	No	Yes	No	Yes	Yes	Yes	No	No
Domain	No	Yes	No	Yes	No	No	No	No

APPENDIX 5 – Related Publications – Journal Articles

R. Özakıncı, A. Tarhan, Early software defect prediction: A systematic map and review, J. Syst. Softw. 144 (2018) 216–239. https://doi.org/10.1016/j.jss.2018.06.025.

R. Özakıncı, A. Tarhan, A Decision Analysis Approach for Selecting Software Defect Prediction Method in the Early Phases (2022), Software Quality Journal, *(In Press)*

APPENDIX 6 – Related Publications – Conference Papers

R. Özakıncı, A. Tarhan, The role of process in early software defect prediction: Methods, attributes and metrics, in: Commun. Comput. Inf. Sci., 2016: pp. 287–300. https://doi.org/10.1007/978-3-319-38980-6_21.

R. Özakıncı, A. Tarhan, Yazılım Geliştirmede Erken Aşamalarda Toplanan Verinin Hata Tahmini Performansına Etkisi, in: CEUR Workshop Proc., 2016: pp. 532–543. http://ceur-ws.org/Vol-1721/UYMS16_paper_120.pdf.

R. Özakıncı, A. Tarhan, An Evaluation Approach for Selecting Suitable Defect Prediction Method at Early Phases, in: Proc. SEAA 2019 - 45th Euromicro Conf. Softw. Eng. Adv. Appl., 2019: pp. 199–203. https://doi.org/10.1109/SEAA.2019.00040.

D. Erhan, A.K. Tarhan, R. Özakıncı, Selecting Suitable Software Effort Estimation Method, in: 30th International Workshop on Software Measurement (IWSM), 15th International Conference on Software Process and Product Measurement (MENSURA), 2020. https://www.iwsm-mensura.org/wp-content/uploads/2020/10/paper11.pdf

APPENDIX 7 – Dissertation Originality Report