The Journal of Systems & Software 144 (2018) 216–239

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Early software defect prediction: A systematic map and review

Rana Özakıncı*, Ayça Tarhan

Software Engineering Research Group, Department of Computer Engineering, Hacettepe University, Ankara, Turkey

ARTICLE INFO

Keywords:
Early defect prediction
Software defect
Software quality
Prediction model
Systematic mapping
Systematic literature review

ABSTRACT

Context: Software defect prediction is a trending research topic, and a wide variety of the published papers focus on coding phase or after. A limited number of papers, however, includes the prior (early) phases of the software development lifecycle (SDLC).
Objective: The goal of this study is to obtain a general view of the characteristics and usefulness of Early Software Defect Prediction (ESDP) models reported in scientific literature.
Method: A systematic mapping and systematic literature review study has been conducted. We searched for the studies reported between 2000 and 2016. We reviewed 52 studies and analyzed the trend and demographics, maturity of state-of-research, in-depth characteristics, success and benefits of ESDP models.
Results: We found that categorical models that rely on requirement and design phase metrics, and few continuous models including metrics from requirements phase are very successful. We also found that most studies reported qualitative benefits of using ESDP models.
Conclusion: We have highlighted the most preferred prediction methods, metrics, datasets and performance evaluation methods, as well as the addressed SDLC phases. We expect the results will be useful for software teams by guiding them to use early predictors effectively in practice, and for researchers in directing their future efforts.

1. Introduction

Software systems are inherently complex and have a constantly growing structure. Providing software quality during and after software development is an indispensable task for people involved in software projects. Reliable software development within limited time, budget and resources makes this task even more difficult. Predictive models are used throughout the lifecycle of the software projects to evaluate software development risks (Smidts et al., 1998) and they can be helpful for identifying problem areas early and making any necessary adjustments (Cukic, 2005). Software defect prediction aims to predict the defect-prone sections of the software during its lifecycle phases, with the purpose of enabling development, test and management teams to form an opinion about the quality of the software. Defect prediction models allow software engineers to focus on defective code, hence improving software quality and making better use of resources (Hall et al., 2012).

Numerous defect prediction models have been developed and presented in the literature over the last 30 years (Song et al., 2011). In general, these studies apply some data processing methods and use various software metrics to build prediction models, frequently based on the later phases of the software development lifecycle (SDLC), namely coding, testing, or operational use. This may be considered a missed opportunity, however, to control and guarantee cost-effectiveness (Pandey and Goyal, 2013). Early detection of defects can lead to timely correction of these defects and delivery of maintainable software. For this reason, it may be useful to apply defect prediction in early phases such as requirements, analysis, or design.

Early software defect prediction is needed for early determination of software quality, cost overrun, and optimal development and testing strategy. 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 (IEEE 2017) and use this information to predict the number of defects in coding or testing phases (Kan, 2003). In projects employing incremental or agile development, early evaluation includes identifying defects in early releases to predict defectiveness in later ones (Aydin and Tarhan, 2014). Cross-project defect prediction may also enable early evaluation if its underlying requirements regarding defect data across the projects are met (Zimmermann et al., 2009). In any case, foreseeing the defective parts of the software may provide preventive actions such as additional reviews and more comprehensive testing, therefore it may ensure improving software process control and achieving high software quality (Pandey and Goyal, 2013).

A wide variety of terms is used to describe a software defect and its

* Corresponding author.
E-mail addresses: ranaozakinci@hacettepe.edu.tr (R. Özakıncı), atarhan@hacettepe.edu.tr (A. Tarhan).

https://doi.org/10.1016/j.jss.2018.06.025
Received 31 October 2017; Received in revised form 8 May 2018; Accepted 7 June 2018
Available online 19 June 2018
0164-1212/

derivatives (Society, 2008). Failure means that the system or its components cannot meet the desired functions within certain limits. Fault is defined as a defect that can lead to various failures in the code (Society, 2008). IEEE Standard Classification for Software Anomalies (IEEE 2009) provides a reasonable approach to the use of these terms. According to this standard; defect is an imperfection or deficiency in a work product where that work product does not meet its specifications; for example, some omissions and imperfections found during early in the SDLC (IEEE 2009).

As there are numerous defect prediction studies targeting late phases of software development and early evaluation is critical as discussed above, in this article we focus on early software defect prediction (ESDP) within the software development lifecycle. The term "early" in this context 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. In addition, in order to address the various kinds of anomalies in these early phases, the term "defect" is used throughout the work carried out and presented in this article.

With an aim to obtain a general view about the characteristics and usefulness of ESDP models reported in scientific literature, we identify and report in this article 52 studies published within the years 2000 and 2016. In order to retrieve and categorize the primary studies, a systematic mapping and a following systematic literature review are conducted. The essential characteristics for building early software defect prediction models, which include prediction methods, software metrics, and approaches for evaluating the performance of the prediction models, are elaborated.

To the best of our knowledge, there is no such study in the literature that evaluates the characteristics of ESDP studies according to their prediction model information. Therefore, this paper contributes to the literature in various ways, such as;

- 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,
- Providing a classification scheme and mapping, and
- Analysis of the prediction model design in the studies as well as their prediction performances, benefits or advantages of the early software defect prediction.

The rest of this article is organized as follows. Section 2 gives a summary of the secondary studies on software defect prediction and their findings. Section 3 includes the protocol (i.e. Sections 3.1–3.7) and the classification scheme (i.e. Section 3.8) of this systematic study. Section 4 provides the analysis results of ESDP studies by an overview of the trends and maturity of the studies in time, and by highlighting development phases, prediction methods, software metrics and performance evaluation methods adopted in the prediction models. Section 5 presents a summary of the findings with respect to the research questions, and our suggestions for future research and practice. Section 6 explains the validity threats of this study and discusses the actions taken to minimize them. Section 7 closes the article by overall conclusions.

2. Related work

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. Table 1 shows the list of these secondary studies, with the summary of the study, research method (Systematic Literature Review (SLR), Systematic Mapping (SM), or Literature Review), and the number of papers reviewed by that study. We should note that research methods of the secondary studies were classified based on the guidelines provided by Petersen et al. (2015) and Kitchenham and Charters (2007). If any guidelines were not followed in secondary studies, we classified them as literature review. Our current study, as given in the last row of the table, 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.

We also provide a comparison of the secondary studies based on their research focuses in Table 2. Our study addresses all the research focuses included in software defect prediction area so far. The categories for identifying research focuses of secondary studies can be explained as follows:

- Prediction Methods: Approaches or techniques used in defect prediction model
- Prediction Metrics: Software metrics or measures which are used as inputs to the defect prediction model
- Datasets: Quantitative or qualitative data of various software projects to which defect prediction model applies
- SDLC: Related software development lifecycle phases that originate the inputs for the defect prediction model
- Performance Evaluation Metrics: Measures used to evaluate the performance of the defect prediction model
- Performance: Quantitative results to be used for interpreting the success of the defect prediction model
- Context: Information about the context of the defect prediction model applied
- Highlights on Main Findings: Evaluation of the results or findings reported in primary studies

Most of the secondary studies searched and reviewed the literature on software defect prediction with a focus on prediction methods, metrics, datasets, and performance evaluation of the models; and none of these studies has examined the literature in relation to ESDP. We have therefore conducted and reported in this article a systematic mapping and review study that focuses on early software defect prediction. The first results from a systematic mapping of studies published between 2000 and 2015 and focusing on process metrics in a narrower scope were published as a conference paper by Ozakinci and Tarhan (2016). In that paper, we reviewed 18 primary studies in depth, which utilize process-based data, in order to discover software process metrics used in the ESDP models. The results showed that process-based software metrics (i.e. effort of review activities, requirement stability, process maturity and number of the defects found in review), play an important role in the early prediction, and that effort based process metrics collected from the review activities were found to be commonly used in the ESDP studies. This study is an extension over (Ozakinci and Tarhan, 2016). We made additions in several aspects, such as the number of studies reviewed, the focus of the conducted systematic study, and additional research questions. In this study, in addition to our previous study, we reviewed 11 more studies that were included from both the year interval (2000 and 2016) and the additional complementary searches we performed. Our focus is also broader than the former, with regard to our new mapping questions (MQs) and review questions (RQs), mainly MQ2 and RQ2. In other words, in this article, we systematically mapped and reviewed the literature by addressing a wider set of papers and adding more aspects in the analysis of ESDP studies. In addition, we have developed and finalized a classification scheme as another contribution to the literature, which is explained in detail in Section 3.8.

3. Goal and research method

This section provides the research protocol (i.e. Sections 3.1–3.7)

Table 1
Secondary studies on software defect prediction.

Ref.	Title	Overview	Research method	# of papers reviewed
(Catal and Diri, 2009)	A systematic review of software fault prediction studies	Catal and Diri reviewed software defect prediction papers with a focus on types of metrics, methods and datasets. The results show that the percentage of use of public databases and machine learning approaches increased significantly after 2005 when PROMISE repository was created.	SLR	74
(Catal, 2011)	Software fault prediction: A literature review and current trends	Catal investigated 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.	Literature Review	90
(Jureczko and Madeyski, 2011)	A review of process metrics in defect prediction studies	Jureczko and Madeyski presented a review of research studies that investigated process metrics in software defect prediction. 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.	Literature Review	*Not specified*
(Hall et al., 2012)	A systematic literature review on fault prediction performance in software engineering	Hall et al. investigated the performances of defect prediction models. The main objective was to assess context, independent variables and modeling techniques and their influences to the performance of the models. The main findings showed that models based on simple modeling techniques such as Naïve Bayes or Logistic Regression performed well. In addition, combinations of independent variables, and usage of feature selection techniques resulted in better performance.	SLR	A subset of 36 out of 208 studies initially included
(Radjenovic et al., 2013)	Software fault prediction metrics: A systematic literature review	Radjenovic et al. reviewed software metrics and their applicability in software defect prediction. They found that object-oriented (OO) metrics were used nearly twice as often compared to traditional source code metrics or process metrics. They also stated that OO and process metrics have been reported to be more successful in finding defects compared to traditional size and complexity metrics, while process metrics seem to be better at predicting post-release defects compared to any static code metrics.	SLR	106
(Malhotra, 2015)	A systematic review of machine learning techniques for software fault prediction	Malhotra analyzed the performance of the machine learning techniques for software defect prediction models and summarized the characteristics based on metrics reduction techniques, metrics, data sets 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.	SLR	64
(Murillo-Morera et al., 2015)	Software fault prediction: A systematic mapping study	Murillo-Morera et al. investigated the software metrics, prediction techniques based on data mining or machine learning and their performance. They found the most frequent combination of metrics, models and techniques were: Halstead, McCabe and LOC + Random Forest, Naive Bayes, Logistic Regression and Decision Tree.	SM	70
(Singh et al., 2015)	A systematic review on software defect prediction	Singh et al. investigated various prediction methods used in the area. According to the results, researchers have mainly used multivariate regression analysis, genetic algorithm, neural network, Bayesian network data mining techniques and machine learning approaches for software defect prediction. It is stated that NASA datasets are the most common data source and widely used in the area.	Literature Review	20
(Wahono, 2015)	A Systematic literature review of software defect prediction: Research trends, datasets, methods and frameworks	Wahono identified and analyzed the research trends, datasets, methods and frameworks used in software defect prediction studies. The results showed that about 77% of the research studies were related to classification methods, and 65% of the research studies used public datasets.	SLR	71
(Ozakinci and Tarhan, 2016)	The role of process in early software defect prediction: Methods, attributes and metrics	Ozakinci and Tarhan presented initial results from a systematic mapping of 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.	SM	41
This study	Early software defect prediction: A systematic map and review	In this study, we systematically mapped and reviewed 52 primary studies published between 2000 and 2016. We 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.	Systematic mapping and literature review*	52

**Research method applied in this study is described in Section 3.2.*

Table 2
Comparison based on research focuses of secondary studies.

Reference	Prediction methods	Prediction metrics	Datasets	SDLC	Performance evaluation metrics	Performance	Context	Highlights on main findings
(Catal and Diri, 2009)	X	X	X					
(Catal, 2011)	X	X	X		X			X
(Jureczko and Madeyski, 2011)		X*	X					X
(Hall et al., 2012)	X	X	X		X	X	X	X
(Radjenovic et al., 2013)	X	X	X	X**			X	
(Malhotra, 2015)	X***	X	X		X	X		
(Murillo-Morera et al., 2015)	X	X	X		X	X		
(Singh et al., 2015)	X		X					X
(Wahono, 2015)	X		X			X****		
(Ozakinci and Tarhan, 2016)	X	X*	X	X				
This study	X	X	X	X	X	X	X	X

*Process metrics, ** Pre/Post-release, *** Machine learning techniques, **** Partially given.

and the classification scheme (i.e. Section 3.8) of this systematic study. In particular, the research questions are firstly elicited (Section 3.1), then the research method is described (Section 3.2). The search strategy is explained in detail (Section 3.3) with the underlying search process. Afterwards, the criteria for both study selection (Section 3.4) and quality assessment (Section 3.5) are presented. We then explain the data extraction (Section 3.6) and data synthesis (Section 3.7) process applied to primary studies. Finally, classification scheme is provided (Section 3.8) with a detailed explanation.

3.1. Goal and research questions

In this study, with the goal of understanding the state-of-the-art and highlight potential benefits of early software defect prediction, we conducted systematic mapping and systematic literature review of software defect prediction studies regarding the models that use metrics data from the early phases of software development. Accordingly, we have identified the following research questions as mapping and review questions and investigated their responses by conducting a systematic mapping and a following systematic literature review.

- **MQ 1. What is the trend and demographics of the studies on ESDP?** Review the bibliometric studies in early software defect prediction area.
  - **MQ1.1 What is the annual number of the studies?**
  - **MQ1.2 Which papers have been cited the most by other papers?**
  - **MQ1.3 What is the distribution of the types of publications?**
  - **MQ1.4 Which are the main venues (journals, conferences, etc.) of the publications?**
- **MQ 2. How mature is the state-of-research on ESDP?** Review the contribution and research type of studies.
  - **MQ2.1 What is the contribution type of the studies?** Example contribution types include model/approach, empirical study etc.
  - **MQ2.2 What is the research type of the studies?** Example research types include solution proposal, validation research etc.
- **RQ 1. What are the characteristics of models on ESDP?**
  - **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 early software defect prediction models if reported.

3.2. Research method

Systematic mapping (SM) studies are used to structure a research area with the aim of providing a general overview of the topic (Petersen et al., 2015). A systematic mapping study allows the evidence about the topic to be plotted at a high level of granularity. Therefore, the findings can help to direct focus on a future systematic literature review by allowing a comprehensive research of the primary studies. The output of a systematic mapping study can be a map, containing a classification about the research area regarding the research questions. A systematic literature review (SLR) is conducted for identifying, evaluating and interpreting the evidence based on all existing research for a particular subject area (Kitchenham and Charters, 2007). SLRs can provide recommendations for identifying gaps or suggest areas for further evaluation (Petersen et al., 2015; Kitchenham and Charters, 2007; Kitchenham et al., 2009). This study is carried out as a systematic mapping and systematic literature review study, following the guideline and protocol proposed by both Petersen et al. (2015) and Kitchenham and Charters (2007). Note that Petersen (2011) and Idri and Abran (2015) also adopted the same methodology for conducting systematic mapping and review study. The protocol of our systematic study is shown in Fig. 1. A detailed explanation for performing each step is also given in the following sections, thereafter which this study will be referred as a systematic review.

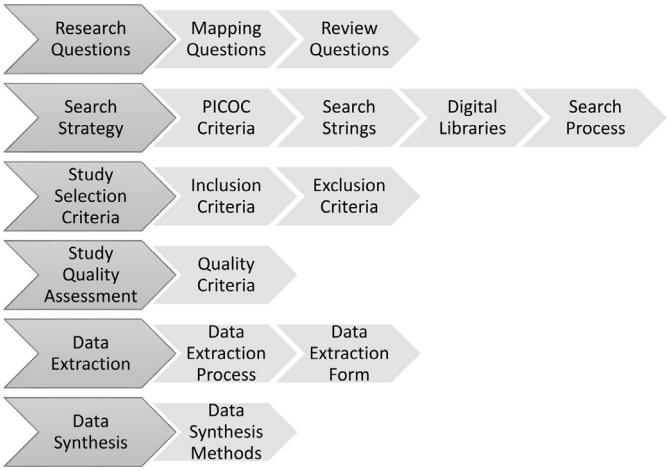


Fig. 1. Research protocol for systematic mapping and systematic literature review.

3.3. Search strategy

PICOC (Population, Intervention, Comparison, Outcomes, and Context) suggested by Kitchenham and Charters (2007) was developed in order to identify keywords and specify search strings:

- Population: Software
- Intervention: Early software defect prediction model
- Comparison: The characteristics of the early software defect prediction models
- Outcomes: The methods, attributes, metrics, performance evaluations
- Context: Academia (scientific literature)

Our concern was to retrieve the studies that focus on early software defect prediction as already mentioned in the introduction. Accordingly, the term "early" in selected studies would refer to the phases of requirements or design that originated metrics for building prediction models. Also, the studies that included data from later phases of coding or testing *in addition to* the early phases of requirements or design would also be considered in study selection in order to investigate their relative effect in building prediction models (i.e. performance evaluation as required by RQ2.2). However, the studies that included data only from the later phases such as coding, testing, or customer use (and not from requirements and design phases) would not be considered in study selection.

By considering the above objective and the existence of numerous studies in software defect prediction, and also after a number of initial searches, we decided on our search strategy. In spite of the fact that there are many terms used meaning the term "defect" (e.g. fault, error, and bug), only the "fault" keyword was further included, since the other two words do not reflect the meaning of the faults that occur in the early phases. We also included "reliability" keyword in the search string in order to be able to catch the studies that are named under reliability but focus and apply "defect" prediction actually.

The main body of final search strings and added keywords were as follows:

- Title: ((*"early" or "earlier"*) and
- Title – abstract – keywords:

("software defect" or "software fault" or "software reliability") and ("prediction" or "estimation" or "analysis"))

During the initial searches, we noticed that software defect prediction is a very broad topic and it is challenging to decide the best-fit search strategy. Therefore, after spending several days in screening potentially relevant studies via Google Scholar, we identified 10 control papers (five having "early" in their title and five not) with high average citations and evaluated the performance of candidate searches in retrieving these papers. Consequently, we selected the candidate search above because it could catch the control papers with the highest rate and would also enable economic use of our resources. In order to mitigate the potential weaknesses in our selected search string, we applied a complementary search strategy. In this regard, we applied backward snowballing (Wohlin, 2014), reviewed pools of recent secondary studies (e.g. Malhotra (2015), Wahono (2015)), and performed a secondary search in Google Scholar as the meta-search engine.

We ran searches by using the selected search string for the studies reported between the years 2000 and 2016 in the following digital libraries of the scientific literature (in alphabetical order): ACM, ScienceDirect, Scopus, SpringerLink, Web of Science, and Wiley. Table 3 shows the searches ran in the digital libraries, and the number of studies initially retrieved. We should note that few electronic libraries (e.g., SpringerLink) did not allow to use the search string as-is. For these sites, we separated the search into several sub-searches preserving the initial search context.

Table 3
Number of studies initially retrieved from digital libraries.

Digital library	Initially retrieved
ACM	5
ScienceDirect	3
Scopus	61
SpringerLink	28
Web of Science	26
Wiley	4
TOTAL	**127**

Initially, 127 studies were retrieved from the searches and 38 of them were eliminated as duplicates. On the resulting 89 unique candidates, we applied study selection criteria presented in Section 3.4. We first applied criteria C1 by reading study titles and/or abstracts and excluded 14 studies based on this criterion. When we had difficulty in having a decision about a study, we browsed through its potentially relevant sections. By this way, we excluded 27 more studies. Then, we performed full-text reading on the rest of the studies (48) in order to decide inclusion/exclusion based on criteria C2, and excluded 15 papers out of 48. After this step, we also applied quality assessment based on our quality assessment criteria presented in Section 3.5 and excluded five studies. Consequently, we initially included 28 studies in the pool.

Within the complementary search strategy, we checked the pools of recently published systematic secondary studies (e.g. Malhotra (2015), Wahono (2015)) and also run secondary searches in Google Scholar as the meta-search engine in case we missed some primary studies. Two studies were retrieved from the pools of recent secondary studies. The secondary search strings were selected from the initial candidates which had underperformed in retrieving the control papers in comparison to our selected search string. In this step, we initially retrieved 316 studies and selected 35 of them through title and abstract. Then we performed full-text reading and selected three studies to be included in our pool. The complementary searches ran in Google Scholar are defined below:

(1) (``software requirements" OR ``requirements document" OR ``requirements specification" OR ``requirements model") AND ("software defect" or "software fault" or "software reliability") and ("prediction" or "estimation" or "analysis")
(2) (``user story" OR ``user scenario" OR ``use case") AND ("software defect" or "software fault" or "software reliability") and ("prediction" or "estimation" or "analysis")

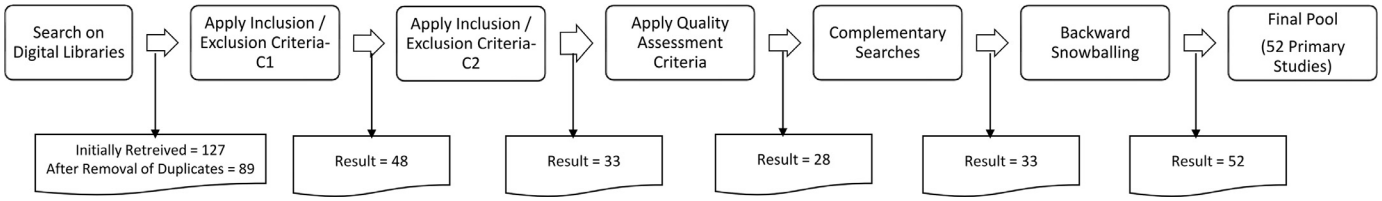


Fig. 2. Study selection process.

(3) (``software architecture" OR ``software design" OR ``design document" OR ``design specification" OR ``design model") AND (“software defect” or “software fault” or “software reliability”) and (“prediction” or “estimation” or “analysis”)

Through the 33 resulting studies included in the pool, backward snowballing approach (Wohlin, 2014) was performed. While performing snowballing, we first checked for publication years (between 2000 and 2016). Second, we read the titles of the papers, if it may be tentatively included, then we checked for publication venues and authors. Afterwards, we looked through the study which was being examined, and found out where and how the study was referenced within the text. If the purpose and context matched our criteria, we included that study as a strong candidate for further examination. We retrieved 66 studies from the start set for snowballing, and iteratively excluded 45 of them based on inclusion/exclusion criteria. Lastly, we added 21 remaining studies to the pool, all included via backward snowballing. As a result, our final pool included a total of 52 studies.

It is important to note that, both inclusion/exclusion and quality assessment criteria were considered while applying the snowballing procedure and making further reviews. Furthermore, we regularly checked for duplicates and identified clusters of papers, thus excluded a couple of primary studies because of the reason that they were included by follow-up studies published by the same authors. For example, journal article [S49] derived from a previous conference paper (Kläs et al., 2008), and therefore conference paper was excluded from the set of primary studies.

Fig. 2 shows the search process and the number of the papers obtained in each step. To ensure transparency, we have made the entire repository of the primary studies available online at (Ozakinci and Tarhan, 2017).

3.4. Study selection criteria

We carefully defined the inclusion and exclusion criteria to ensure including all the relevant studies but not those which were out of scope. To be included in this research, a study must be reported in a paper published in English as a journal article, conference proceedings, or book chapter. Workshop papers were also included in conference proceedings category. The criteria for studies to be included in this systematic review are presented in Table 4. The motivation for the criteria was related to the goal of this systematic review study, i.e., to retrieve defect prediction studies with the specific aim of describing the characteristics of prediction models and investigating the effect of using early-phase metrics in the performance of these models.

In order to explain the criteria in detail, we need to give a few examples of those excluded and the rationale behind. Table 5 introduces several studies with the causes of elimination.

3.5. Study quality assessment

The approach for identifying studies suitable for this systematic review study is motivated by Kitchenham and Charters (2007) notion of a quality check. Quality assessment is specifically focused on determining the papers that report sufficient information to compare to other studies in terms of answering defined research questions. To be able to do this, a basic set of information (prediction method, datasets, and metrics) must have been reported in papers. Without these details, it will be very difficult to accurately extract the required information for answering the review questions (e.g. in RQ1, we are focusing on the characteristics of early defect prediction models in order to see how they were constructed). Therefore, we have developed and applied a set of criteria in order to ensure that sufficient information is reported in ESDP studies. Quality assessment criteria were considered during the full-text reading of the primary studies in the pool. All primary studies were examined in detail by the first author of this paper according to the defined criteria, and selective reviews were conducted by the second author. The quality assessment criteria are defined below.

- Can it be extracted from the study which SDLC phase(s) originate metrics as input to the prediction?
- Does the study clearly define which prediction method(s) or approach(es) were used?
- Does the study give metrics used to build the prediction model?

3.6. Data extraction

Data extraction process was conducted according to a data extraction form which was filled in for each primary study in the pool. The form was split into some sections for categorizing the information, typically in accordance with our research questions (i.e. MQs and RQs). The first section is about the basic information of the papers such as study ID, title, authors, summary, and whether a paper is found from search process or gathered from snowballing process (i.e. source in search strategy). The second section presents information about trends and demographics of the papers, such as publication year, publication type, citation number, average citation. In the third section, contribution and research facets are classified. The fourth section describes the design characteristics for ESDP models. The data of this theme includes datasets used in the study, the SDLC phases that originate metrics for the prediction model, the software entities and related attributes for each entity, software metrics and prediction methods used for building the model, as well as the contextual parameters for the study, if reported. Moreover, in the fifth section, performance evaluation

Table 4
Study inclusion and exclusion criteria.

ID	Inclusion criteria	Exclusion criteria
C1	studies that focus on software “defect” prediction	studies that focus on reliability growth of software product as an external quality attribute, or on estimating cost
C2	studies that include metrics from “at least” one early phase (requirement and/or design) for defect prediction	studies that focus “only” on the coding and/or testing phases of SDLC, or on customer-use.

Table 5
A selection of excluded studies with causes of elimination.

Ref.	Study	Cause of elimination
(Fenton and Ohlsson, 2000)	Quantitative analysis of faults and failures in a complex software system	This study is not focusing on predicting software defects, rather is analyzing a range of software engineering hypotheses such as the use of early fault data to predict later fault and failure data. Therefore, it was excluded in accordance to C1.
(Singh et al., 2016)	Early prediction of software reliability: a case study with a nuclear power plant system	This study focuses on predicting software reliability rather than defect prediction, and was excluded in accordance to C1.
(Nagappan et al., 2006)	Using historical in-process and product metrics for early estimation of software failures	This study has an aim as "to make estimates of post-release failures early in the software development cycle, during the implementation and testing phases." Therefore it focuses only on coding and later phases, and was excluded in accordance to C2.
(Hu et al., 2006)	Early software reliability prediction with extended ANN model	This study is specifically focusing on early software testing phase, and was excluded in accordance to C2.
(Nagappan and Ball, 2005)	Static analysis tools as early indicators of pre-release defect density	This study is using only coding phase-based metrics for prediction, and was excluded in accordance to C2.

methods/metrics, performance results of the models, main findings, and benefits of the models are investigated. It is important to note that benefits or advantages/disadvantages of the ESDP were mostly presented qualitatively in the studies. Therefore, this section has the free text information about the benefits where reported. All sections and extracted data names can be seen in Table 6. Data extraction form was developed in iterations and we recorded the information in spreadsheets as the data was extracted from the studies.

3.7. Data synthesis

Synthesis of the extracted data is necessary in order to consolidate the evidence of the included primary studies. Since the extracted data was both quantitative and qualitative, three different data synthesis approaches presented by Dixon-Woods et al. (2005) were used. Firstly, content analysis was used to categorize the data and identify frequencies of these categories. The tabulation of the categories simplified the evaluation of the evidence gathered. We extracted the data based on the data extraction form and consequently provided a classification scheme in order to categorize data in a general form. Some categories used in this study were already identified in the literature. We refer to them in the explanation of the classification scheme in Section 3.8 (e.g., for performance evaluation methods/metrics we refer to Hall et al. (2012)). Secondly, a narrative synthesis was used in Section 5 to recount and describe the principal findings of this study with an interpretive approach. Various kinds of evidence could be discussed side by side by means of narrative summaries. Lastly, the reciprocal translational analysis was performed which aims to define the main themes in each study and translate them into the context of other studies. We applied reciprocal analysis in order to synthesize the main benefits (RQ2.3) reported in primary studies by selecting the themes with the best explanatory power. It is worth to note that, meta-analysis was not applied in this study because of the variety of designs in the defect prediction models and performance evaluation approaches in the selected studies.

3.8. Classification scheme

The main classification scheme of the study was prepared based on key-wording described in (Petersen et al., 2008). Keywording is a two-step process that simplifies the developing of the classification scheme by ensuring that the categories were derived systematically and considering all studies into account. We performed key-wording as in the following steps: First, all papers were identified and reviewed by specifying whether they would be included in the study or not, regarding its compliance with the study selection criteria. Then, selected papers were analyzed deeply in order to map the properties of each paper with determined categories. To gather all information about these properties, full-text of each study was read and elicited information was recorded within a tabular form. Table 7 shows the classification scheme. In the table, 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. Finally, 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 with regard to prediction method category will be marked in this case. The explanation for each property and related possible values is given next.

- "Publication type" refers to the type of venues such as Journal or Conference. It is the most frequently reported dimension in systematic mapping studies (Petersen et al., 2015).
- "Contribution type" and "Research type" facets are taken as explained in (Petersen et al., 2008). According to this source, contribution type facet may refer to determining the type of intervention being studied, such as a method, model, tool, metric or process. We also added the type 'empirical (case) study only', in which the focus of the paper is only an empirical study, rather than contributing to any other (e.g., model/approach, method/technique, tool, metric, or process). Research type facet reflects the approach for conducting the research paper. Research facet of a paper might be solution proposal, weak or strong empirical study.
- "Dataset type" refers to the access the data used in the study is whether public or private. If public datasets were used in the study, it could be more easily repeated. If private data sets were used, the study could not be repeated easily by other researchers and therefore the results could be interpreted as suspicious. Thus, we categorized the studies according to data set type as follows:
  - Private: Neither dataset, defect data or source code is available. It is therefore not definite if the study is reproducible. It is worth to note that if the study did not state the availability of data set, it was categorized as private in our systematic review.
  - Public: The metrics and the defect data are publicly available (e.g. PROMISE Data Repository). The study is deemed to be reproducible.
- "SDLC phase" states the software development lifecycle 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. We categorized the phases as Requirement, Design, Coding, and Testing. Together with the phase information, we would like to report the software development method used in the studies; however, only a few papers [S1, S5, S13, S37] expressed clearly the development method used.
- According to Fenton and Bieman (2014), 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 (Fenton and Bieman, 2014) and measurable product and process attributes of Florac et al. (1997), we include "Software Entity" and "Attributes Associated with Each Entity" to describe the type of the

Table 6
Fields in data extraction form.

Section number	Section name	Column name	Research question
1	Basic information	Study ID, Title, Authors	
2	Trends and Demographics	Publication Year (MQ1.1), Citation Number (MQ1.2), Average Citation (MQ1.2), Publication Type (MQ1.3)	MQ1
3	Contribution and Research Facet	Contribution Type (MQ2.1), Research Type (MQ2.2)	MQ2
4	Model Characteristics	Dataset Type (RQ1.1), SDLC Phase (RQ1.2), Software Entity (RQ1.3), Attributes Associated with Each Entity (RQ1.4), Prediction Methods (RQ1.6), Contextual Parameters (RQ1.7)	RQ1
5	Model Performance and Benefits	Performance Evaluation Methods/Metrics (RQ2.1), Main Findings/Benefits (RQ2.3)	RQ2

entities and their related attributes, respectively. Software entities can be defined as follows:
  - ○ Processes: Software-related activities.
  - ○ Products: Artifacts, deliverables, or documents that result from a process activity.
  - ○ Resources: Entities required by a process activity.
- For each software entity, we included related internal and external attributes into the classification scheme. Internal attributes can be measured purely in terms of the product, process, or resource itself. External attributes can be measured only with respect to how the product, process, or resource relates to its environment, namely considering its behavior. We determined that 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, we added those attributes and metrics to the classification scheme, progressively.
- "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 data sets used in the prediction studies. We adopted some of the contextual characteristics from Zimmermann et al. (2009) and Petersen and Wohlin (2009). Examples of contextual parameters include domain, programming language, and size of the software, development methodology used in the project lifecycle etc.
- "Performance Evaluation Methods and Measures" are necessary for assessing the success of the prediction model. According to the classification of Hall et al. (2012), defect prediction studies may report their results via categorical or continuous dependent variables. Categorical studies report their results in terms of predicting whether the software unit is likely to be defect prone or not defect prone. Performance metrics of this kind of reporting can be precision, recall, accuracy, f-measure, and ROC. Continuous studies report their results in terms of the number of defects predicted in a software unit. Measures reported by continuous studies may be based on error calculations.
- "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.

Table 7
Classification scheme.

Research question	Property	Possible values	(M)ultiple/ (S)ingle
MQ1.1	Publication Year	Years between 2000 and 2016	S
MQ1.2	Citation Number	Number of citation of the study	S
MQ1.3	Publication Type	Journal Article (J), Conference Paper (C), Book Chapter (B)	S
MQ1.4	Venue Name	Venue which the study was published	S
MQ2.1	Contribution Type	Proposing Model / approach, Method / technique, Tool, Metric, Process, Empirical (Case) study only	M
MQ2.2	Research Type	Solution Proposal (Only Simple Example), Weak empirical study (Validation Research), Strong empirical study (Evaluation Research)	S
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 10.	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, NA	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 results.	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 12.	M

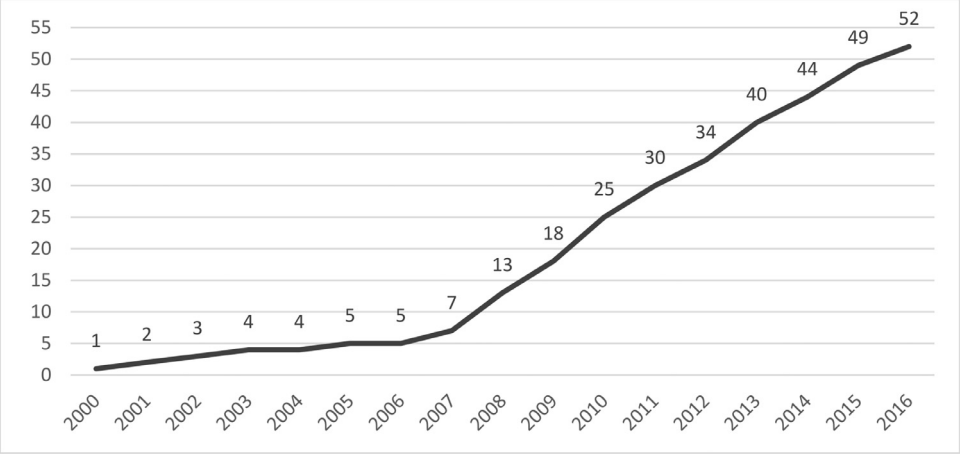


Fig. 3. Cumulative number of early studies per year.

4. Results

Results of our study are presented from Sections 4.1 to 4.4.

4.1. MQ 1 – What is the trend and demographics of the studies on ESDP?

4.1.1. MQ 1.1 - What is the annual number of the studies?

The number of the early software defect prediction studies varied from year to year. Cumulative number of early studies per year is shown in Fig. 3. There has been an interest in early software defect prediction research since 2005 (Catal and Diri, 2009; Wahono, 2015), and the cumulative distribution has an increasing trend so far. The most active years were between 2008 – 2011 and 2013 – 2015, with an average ratio of five studies per year.

4.1.2. MQ 1.2 - Which papers have been cited the most by other papers?

In order to identify the most cited papers, we included the publication year of the study in the calculation of the number of citations, since the year range was very wide. It is important to note that the citation numbers were recently gathered from Google Scholar on April 23, 2018.

The number of citations and the average annual citations per publication year are given in Fig. 4(a) and (b), respectively. In order to visualize the average number of citations, following formula was used:

$$Average\ Citation\ of\ the\ paper\ = \frac{Total\ citation\ of\ the\ paper}{2018 - Publication\ year\ of\ the\ paper}$$

Top three papers with the most total and average citations were [S51, S16, S52] and [S51, S16, S7], respectively.

4.1.3. MQ 1.3 - What is the distribution of the types of publications?

The distribution of the publication types is shown in Fig. 5. More than half of the papers (58%) were published as journal articles, which provides insight about the quality of these studies. Also, some studies (34%) were published as conference proceedings, while few of them (8%) were presented as book chapters. Moreover, in order to see the change of interest in publication types, the cumulative number of studies for each publication type over the years is presented in Fig. 6. Here, we also see that journal articles have had an accelerating trend in the last few years, which is a good indicator for the maturation of the area.

4.1.4. MQ 1.4 - Which are the main venues (journals, conferences, etc.) of the publications?

The ranking of the venues with at least two papers is shown in Table 8. The main venue with three papers was International Journal of System Assurance Engineering and Management. In addition, one conference and one book, which were focused on software reliability, included two papers each on ESDP. Considering these numbers, it can be inferred that there was no common venue for publishing of early software defect prediction studies.

4.2. MQ 2 – How mature is the state-of-research on ESDP?

4.2.1. MQ 2.1 - What is the contribution type of the studies?

Fig. 7 shows the total number of the papers by contribution facet types and the references. The most contributed facets were new model/approach proposal and an empirical study, with percentages of 42% and 40%, respectively. In addition, two papers contributed new metrics for early defect prediction and no study has contributed a new process or tool, which indicate rooms for future proposals.

4.2.2. MQ 2.2 - What is the research type of the studies?

Fig. 8 shows the types of the studies by research facet. The most popular research type was validation research and 75% of the studies did weak empirical study under this category, by reporting the performance evaluation results of the defect prediction models. In solution proposal studies (15%), the performance evaluation of the prediction model was not reported but only a simple example of the model was given. In evaluation research studies, hypothesis testing was conducted and evaluation results were reported strongly. Unfortunately, the rate of evaluation research was few (10%), which indicates the immaturity of the contributions made. It is worth noting that, researches should attach importance to conduct strong empirical studies while proposing new model/approaches in the context of ESDP.

4.3. RQ 1– What are the characteristics of models on ESDP?

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

We have categorized the studies according to their types of datasets, and give the distribution of the types in Fig. 9. 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 (Menzies et al., 2016), (2) qualitative and quantitative data about 31 projects that were published in [S37], 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 Fig. 10. It was obtained from the number of dataset types used in the studies by summing them over the years.

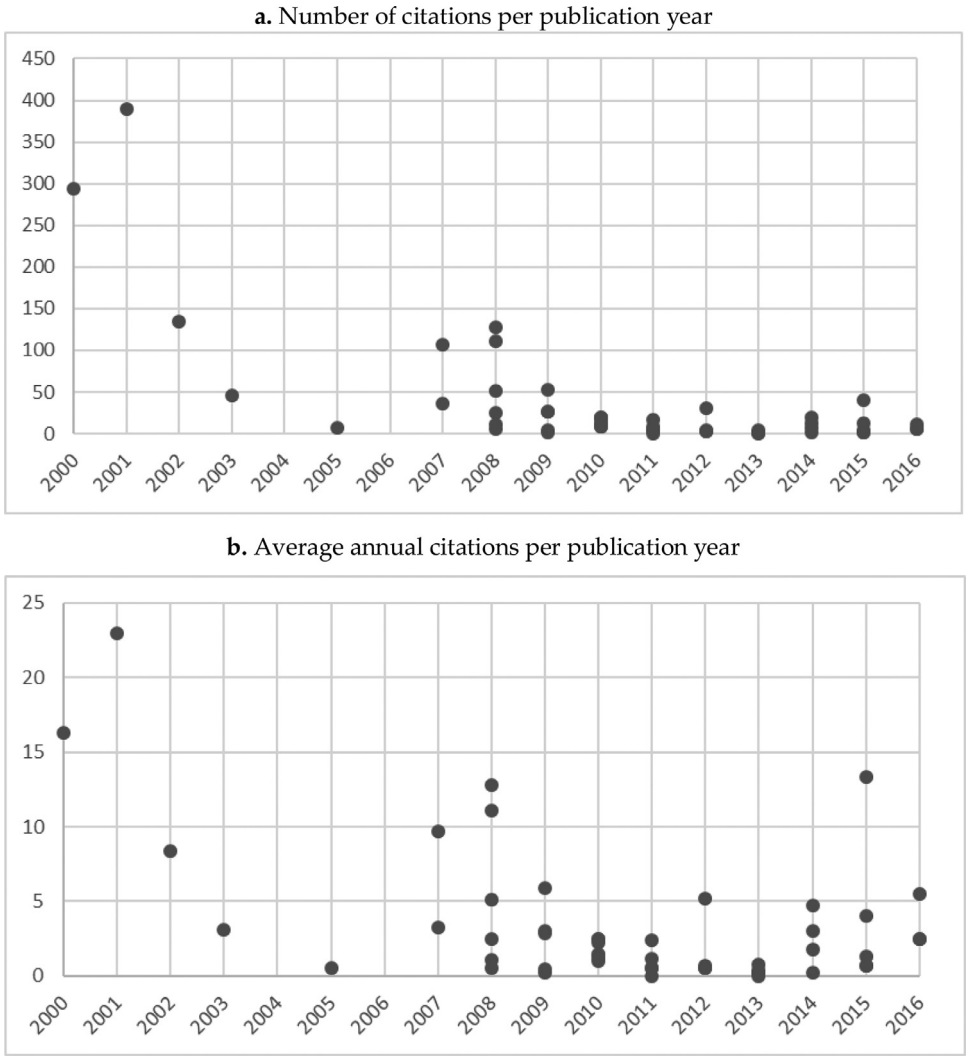


Fig. 4. Total and average numbers of citations per publication year.

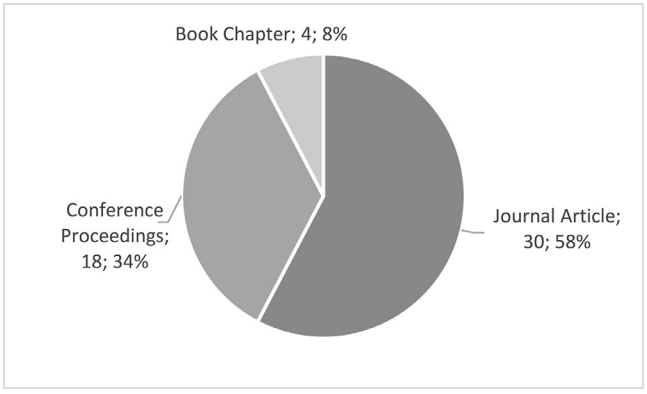


Fig. 5. Distribution of publication types.

4.3.2. 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 Fig. 11. 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.

The cumulative percentages of the SDLC phases associated with early prediction studies can be seen in Fig. 12. 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%.

4.3.3. 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 software metrics, where only two studies used metrics from process and resource entities together. Fourteen studies used software metrics that were related to all entities. The individual distribution of the entities among all studies is shown in Fig. 13.

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

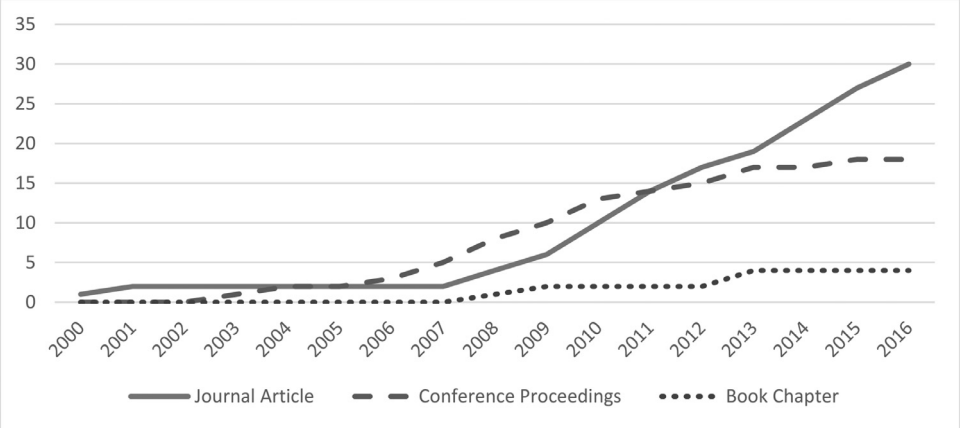


Fig. 6. Cumulative number of publication types per year.

Table 8
Venues ranked by number of papers.

Venue	Number of studies	References
J: International Journal of System Assurance Engineering and Management	3	S14, S30, S43
J: International Journal of Computer Applications	2	S32, S48
J: Empirical Software Engineering	2	S37, S49
J: World Academy of Science, Engineering and Technology	2	S11, S12
C: International Symposium on Software Reliability Engineering	2	S1, S33
B: Early Software Reliability Prediction	2	S24, S34

entities used in studies can be seen in Fig. 14. We see 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.

4.3.4. 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 Fenton and Bieman (2014) and Florac et al. (1997) as shown in Table 9. Accordingly, product structure, size, process effort and human resource characteristics were the most included attributes in the prediction models.

4.3.5. 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 Fenton and Bieman (2014 and Florac et al. (1997) as shown in Table 10. 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.

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

Fig. 15 shows the distribution of the prediction methods used for early defect prediction in the studies. We found 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

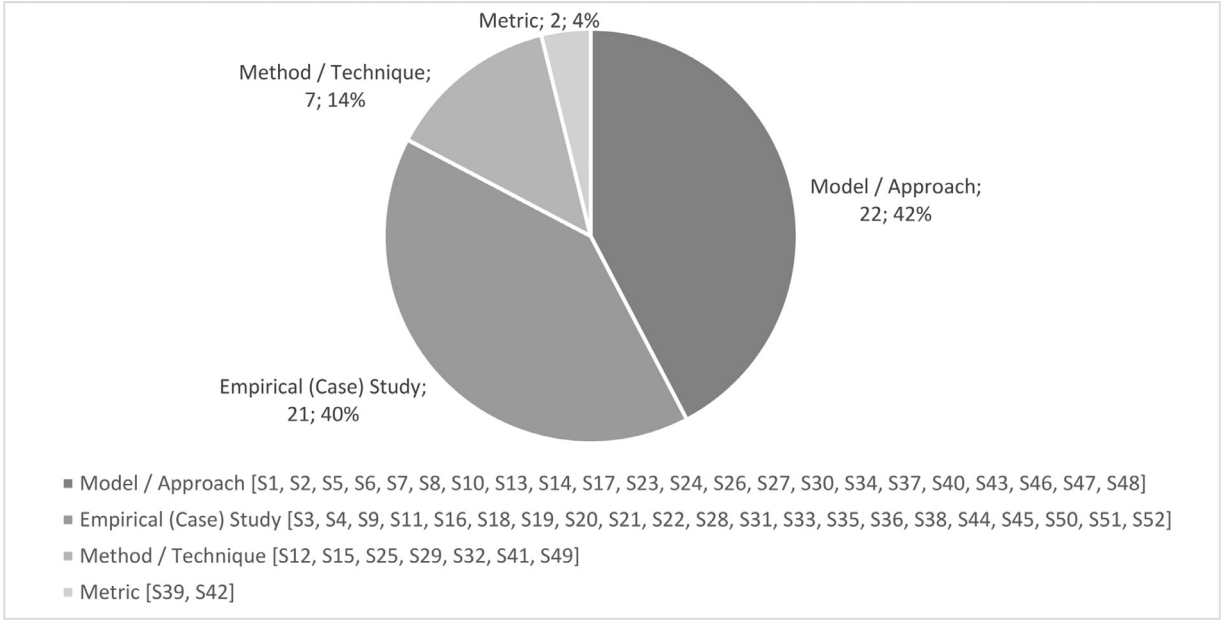


Fig. 7. Contribution facet of the papers.

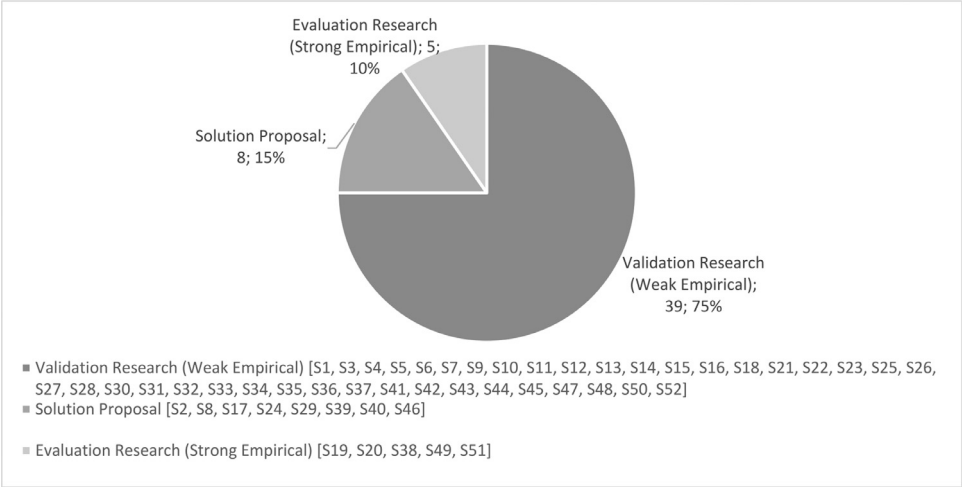


Fig. 8. Research facet of the papers.

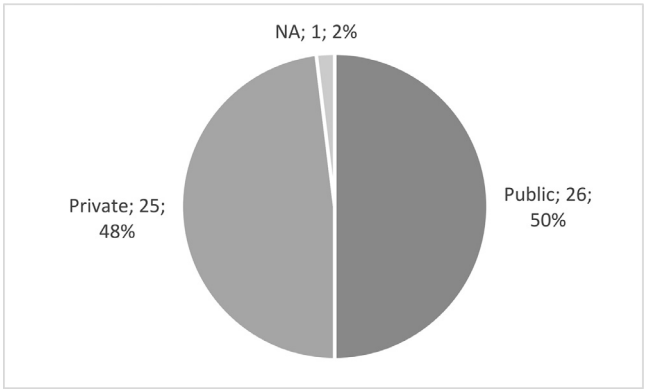


Fig. 9. Distribution of dataset types.

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%.

4.3.7. RQ1.7 - What are the contextual parameters as reported in the prediction models?

We have gathered contextual parameters according to some references, such as Zimmermann et al. (2009) and Petersen and Wohlin (2009). We have 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, we could infer its contextual parameters since the dataset is public to access. Besides, we know the contextual parameters about the NASA MDP dataset 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 11, which include business domain, product size (as KLOC), programming language, development methodology, and effort.

Aside from these public datasets, we extracted the contextual parameters reported in 18 studies out of 25 studies that used private dataset. Fig. 16 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

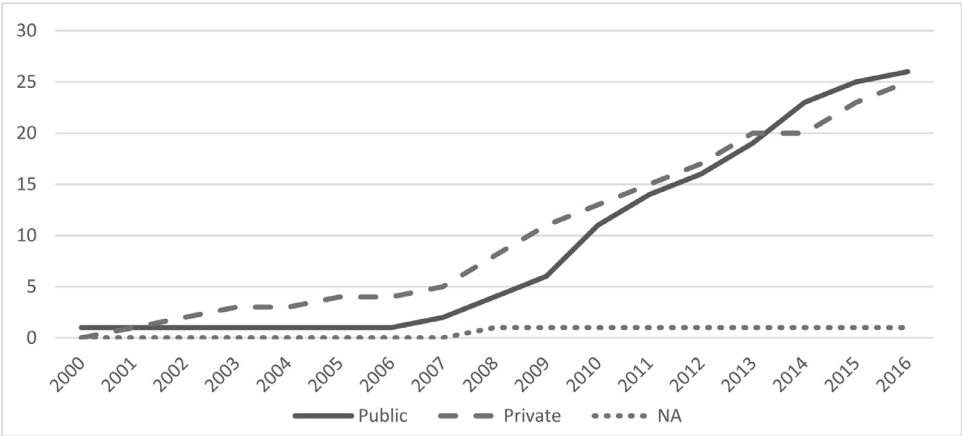


Fig. 10. Cumulative number of datasets per year.

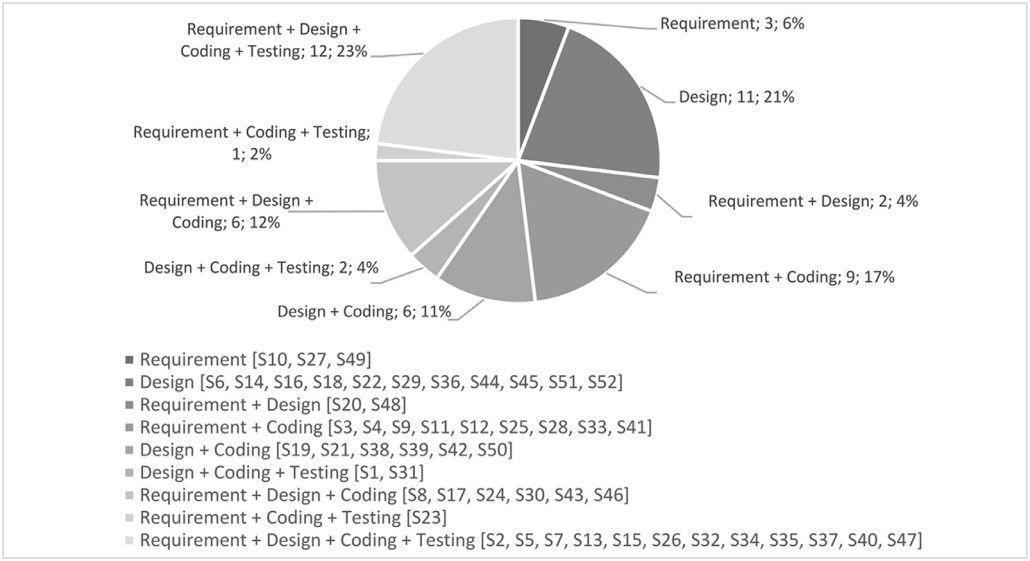


Fig. 11. Individual distribution of SDLC phases.

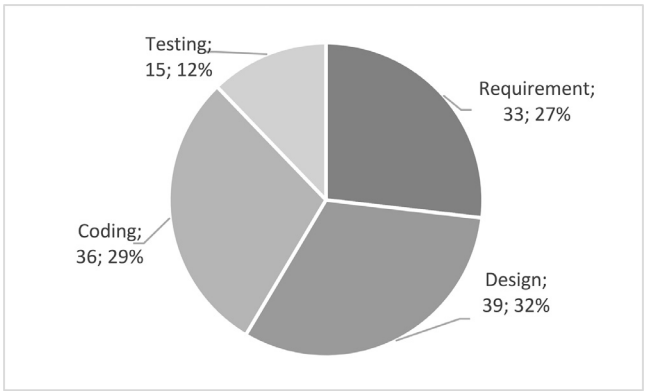


Fig. 12. Cumulative distribution of the SDLC phases.

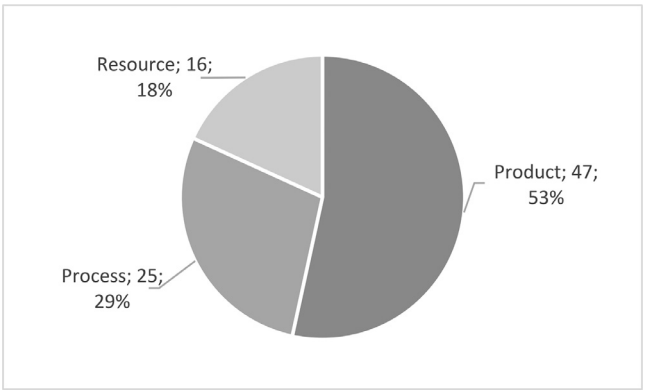


Fig. 14. Cumulative distribution of software entities.

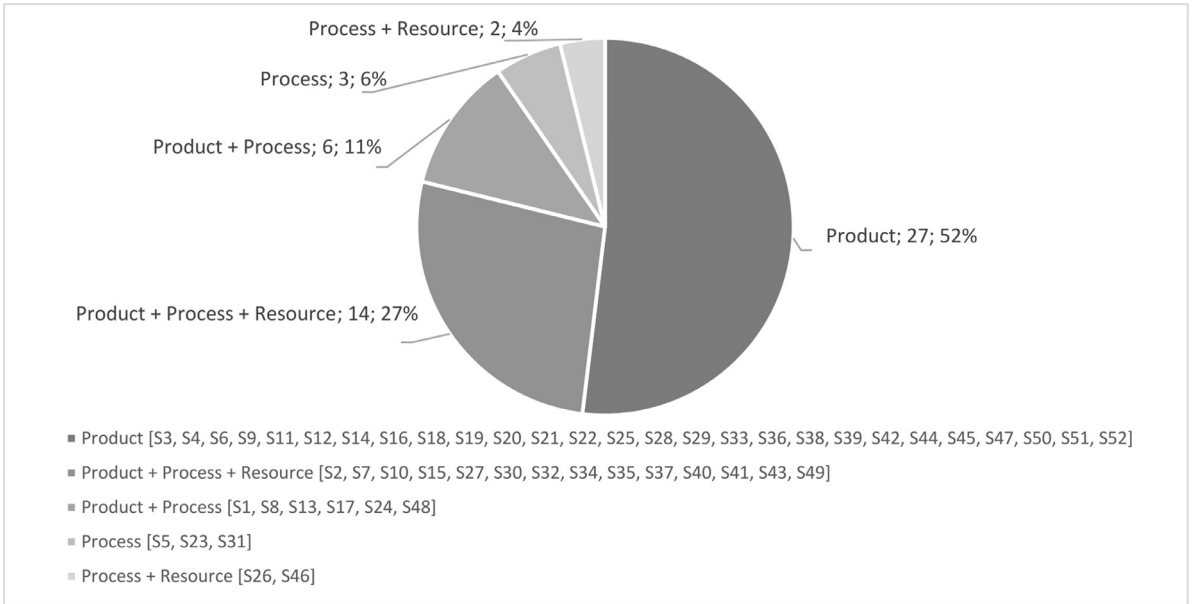


Fig. 13. Individual distribution of software entities.

Table 9
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 as person-day.	S1, S2, S5, S37, S23, S31, S26, S8, S32, S40, S35, S13, S46, S24, S34, S43, S30, S7, S10, S48	20
	Time	Represents the measures related to the time for applying a process activity.	S15, S31, S41	3
	Stability	Identifies 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

development methodologies adopted during the lifecycle 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.

4.4. RQ 2 – Are models of ESDP successful and beneficial?

4.4.1. 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. In Figs. 17–19, we provide the distributions related to performance evaluation methods and measures reported in primary studies. We can see 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.

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.

4.4.2. 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 (Hall et al., 2012). 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. Fig. 20 shows the results based on AUC values; while Fig. 21 shows the results based on precision, recall, and f-measure values as provided by the studies. 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.

For the continuous studies, the results were reported in a variety of measures, which makes it difficult to convert the data presented in these studies 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

Table 10
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
		Code metrics: 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.)	S52, S21, S17, S47, S42, S44, S30, S39, S22, S7, S48, S19	12
		Halstead Metrics (total number of operators, operands etc.)		
		Object-oriented Metrics (Complexity, Length, Coupling, Cohesion, Modularity, Reuse)	S51, S50, S29, S18, S14, S6, S16, S36, S19	9
		Design metrics from UML (e.g. Chidamber and Kemerer (1994))		
		Data flow complexity (DC), cyclomatic complexity (CC)	S8, S24, S43	3
		Requirements complexity (RC), 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 (DRE)	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 (RCR)	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 detected defects from review	S1, S37, S15, S35, S13	5
		Requirement fault density (RFD), design defect density (DDD), fault days number (FDN), code defect density (CDD),	S15, S17, S8, S27, S24, S10, S48,	7
	Adequacy	Analysis, design, review quality	S37, S40, S35, S43	4
		Quality of documented test cases (QDT)	S35, S7, S41	3
		Defined process followed (DPF)	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

(BMMRE), and Mean Absolute Error (MAE).

We provide MMRE results with regard to phase information in Fig. 22, 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, we see 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]. We also see that 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.

Moreover, R^2 values were also preferred among continuous studies. We provide those results with regard to the phase information in Fig. 23. 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 $R^2 = 0.989$). We also see that 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.

4.4.3. 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

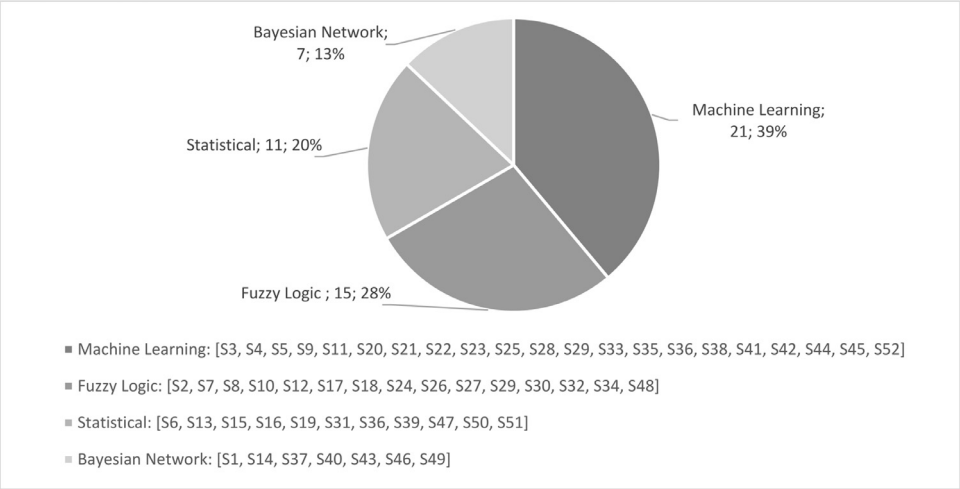


Fig. 15. Distribution of prediction methods.

Table 11
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 (located in PROMISE repo) (Menzies et al., 2016)	14	X	X	X		
Data published in Fenton et al. [S37]	10	X	X	X	X	X
Data published in Cartwright and Shepperd [S16]	2	X	X	X	X	

[S37], it was indicated that an obvious benefit of a Bayesian Network model was its arrangement of a range of decision support and risk assessment capabilities, 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 usefulness of the model was evaluated using measurement data (e.g. artifact size, number of defects found) collected for five historical projects. Knowledge of seven domain experts was elicited 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 12 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. (2005). 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.

In addition to the benefits listed in the table, we would like to report some issues below, which we consider significant and important for highlighting other contributions of the ESDP models.

Two studies [S20, S48] that used only early phase data (i.e. requirement and design phase) reported the criticality of requirements or design stages, and also the combination of the two metric groups for early prediction. In [S20], a hybrid model was proposed that used both requirement and design phase data. The results showed that the hybrid model led to reasonably effective predictions (with an average AUC value of 0.84). In [S48], requirements and object-oriented product measures were used for early reliability prediction and the results obtained were quite promising. It was stated that regardless of the data from within project, ESDP models could be built at design phase by using other design level data of other projects, since cross project defect prediction was confirmed with an AUC value bigger than 0.5 [S22].

Moreover, some studies reported that prediction models would be more accurate and therefore more helpful if they were built by combining metric data of two phases (i.e. hybrid models), such as data from requirement and coding phases [S3, S4, S9, S11, S12, S25, S27, S28, S33], or data from design and coding phases [S21, S38, S39, S42, S50] together. The main benefit of using the hybrid model was claimed that it gives more accurate results than the individual models and thus can help to improve the efficiency of the product, as well as reduce the testing efforts, as defect-prone areas will be already known. In addition, in [S50] it was stated that models, which were built after the implementation of software, were more accurate (with 34%) compared to earlier models. This result, however, was due to the metric that measures the size of the class modification, which was available only after the implementation phase.

5. Summary of overall findings and observations

Below we summarize the results of the MQ and RQs, and discuss the findings and our observations for the research community and practitioners.

MQ 1 - Mapping by trends and demographics

It is good to note that, after year 2007, the number of studies has had an increasing, cumulative trend. One reason can be due to the public access to the datasets such as NASA MDP (Menzies et al., 2016)

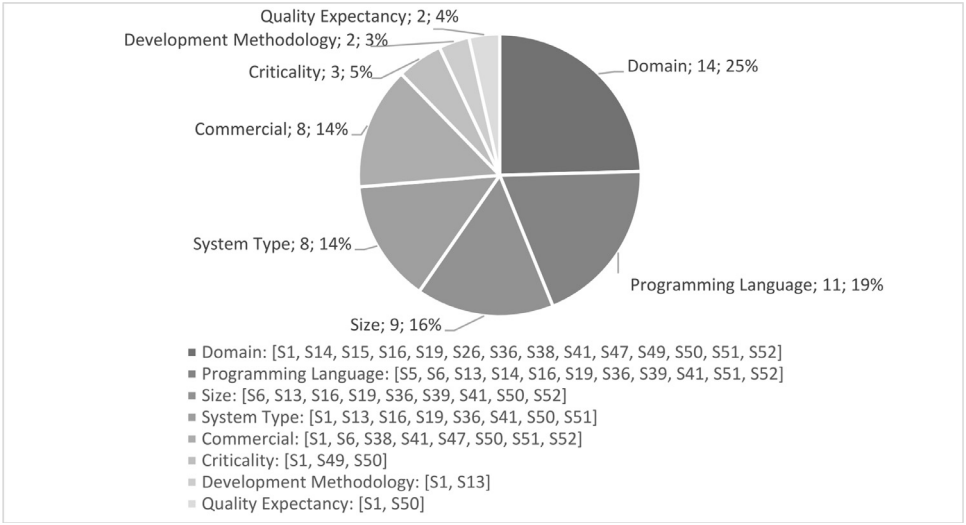


Fig. 16. Categories of contextual parameters reported in 18 primary studies.

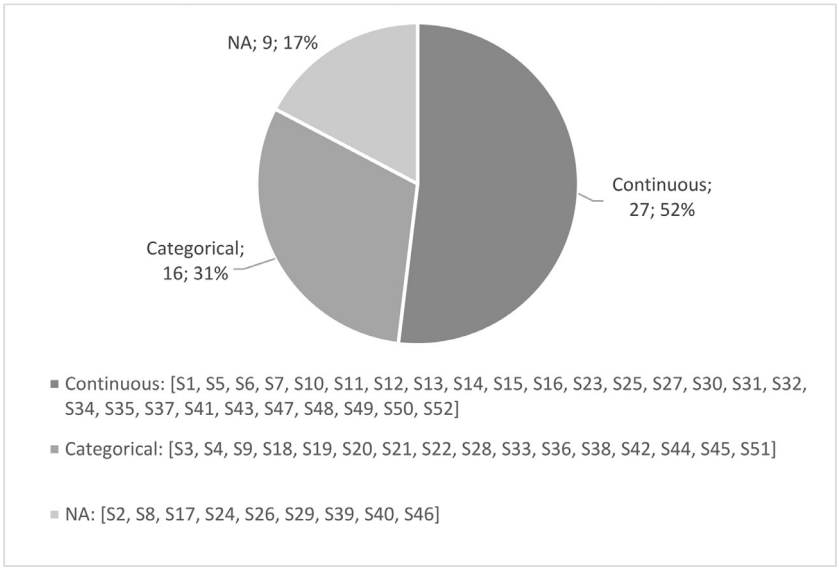


Fig. 17. Distribution of the prediction performance methods.

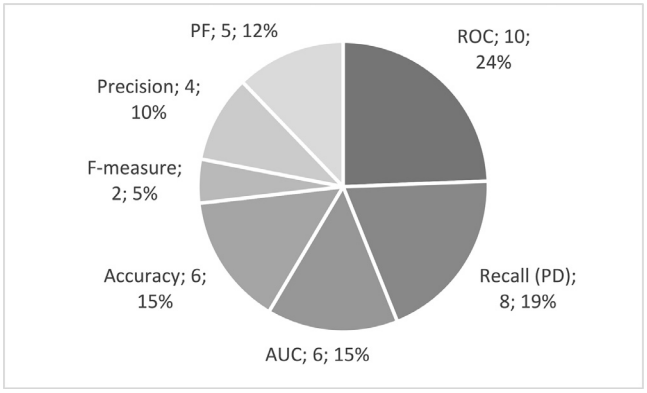


Fig. 18. Distribution of the performance evaluation measures in categorical ESDP models.

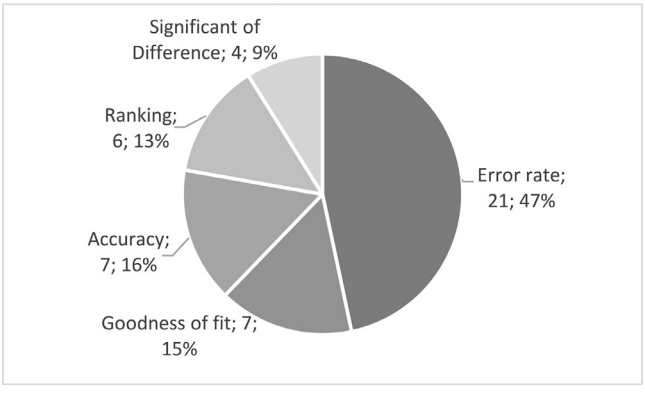


Fig. 19. Distribution of the performance evaluation measures in continuous ESDP models.

and data published in [S37]. In addition, the attention has increased more in journal articles rather than conference proceedings, which is another indicator for the maturation of the area. Top three most cited studies with regard to their total and average citations per year are [S51, S16, S52] and [S51, S16, S7], respectively.

MQ 2 - Mapping by contribution and research facets

Contribution types of the studies showed that most of them propose new models and approaches, such as new prediction models based on

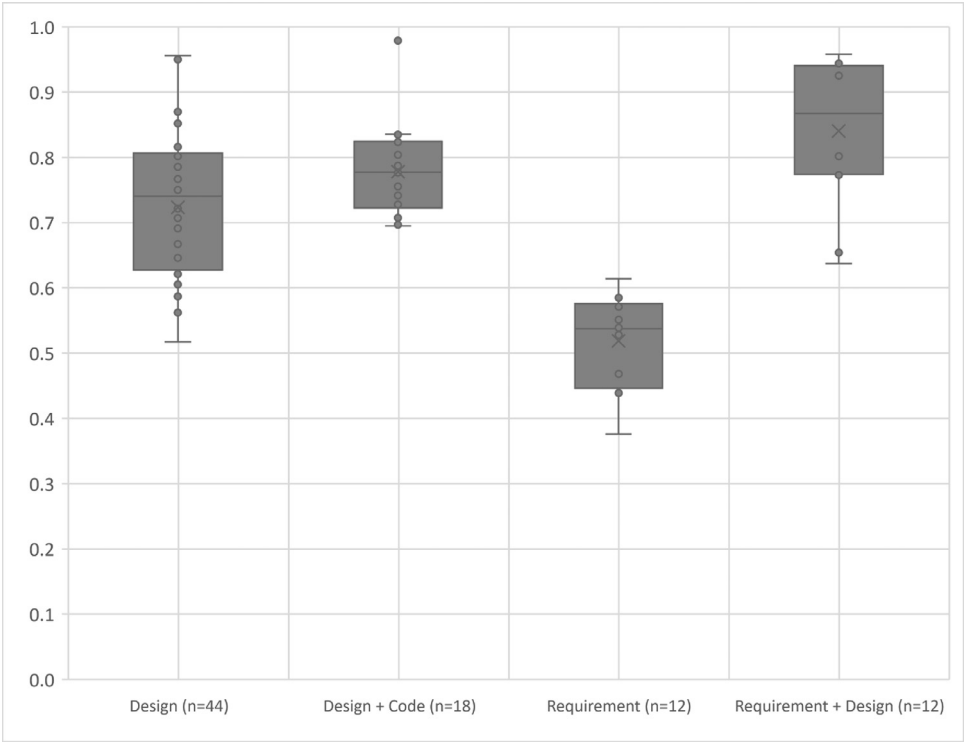


Fig. 20. Performance results with regard to phase in categorical studies as reported in AUC values.

artificial neural networks (ANN) and fuzzy inference systems (FIS), or phase-wise approaches. Contributions as empirical (case) studies are crucial to demonstrate the usefulness of ESDP in practice.

In the sense of research types, most studies are under the facet of "validation research (weak empirical studies)", which can be interpreted as there is a need for stronger empirical studies with hypothesis testing.

When we evaluate the findings of contribution and research types together, we can say that model/approach proposals are not being

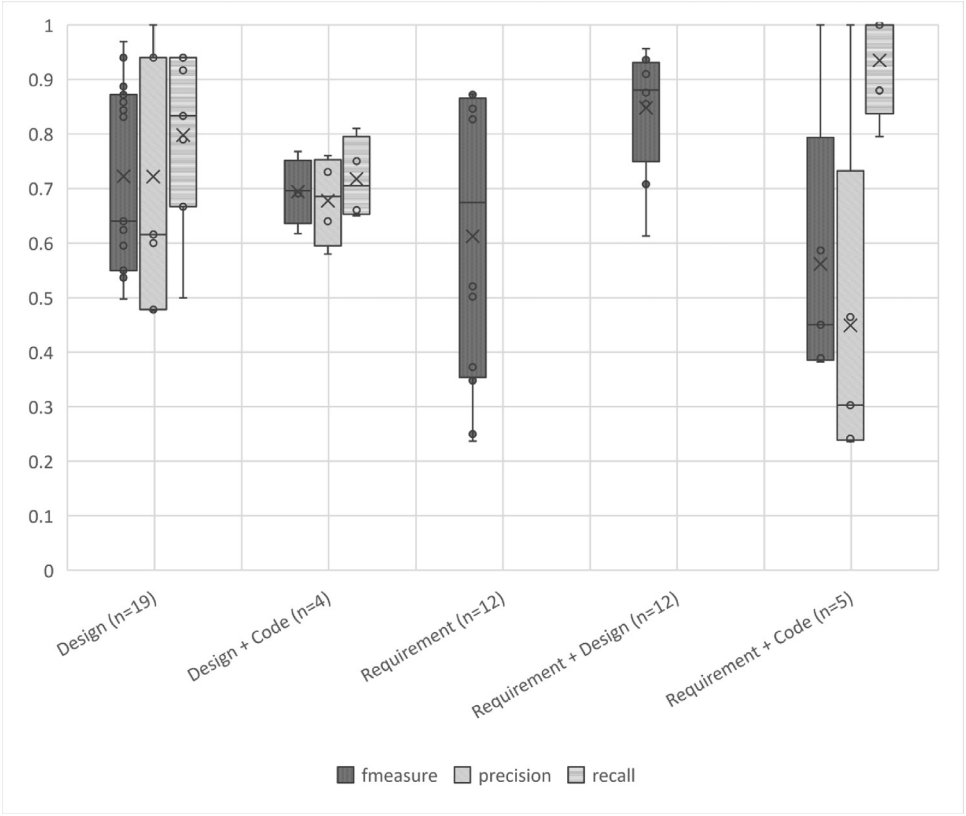


Fig. 21. Performance results with regard to phase in categorical studies as reported in f-measure, precision and recall.

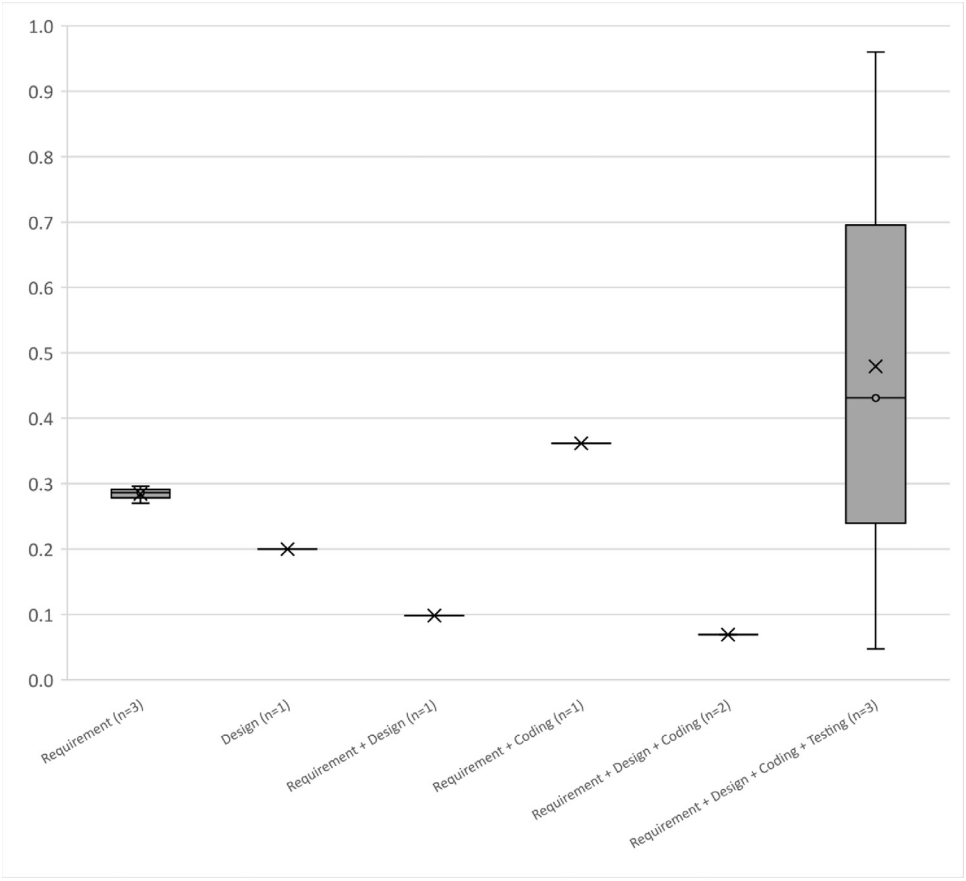


Fig. 22. Performance results with regard to phase in continuous studies as reported in MMRE.

validated strongly. These findings highlight the need and opportunity to perform evaluation research in addition to new proposals and empirical validation studies in the area.

RQ 1 - Review by characteristics of models

Increased interest in public data sets are 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. We can say that 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

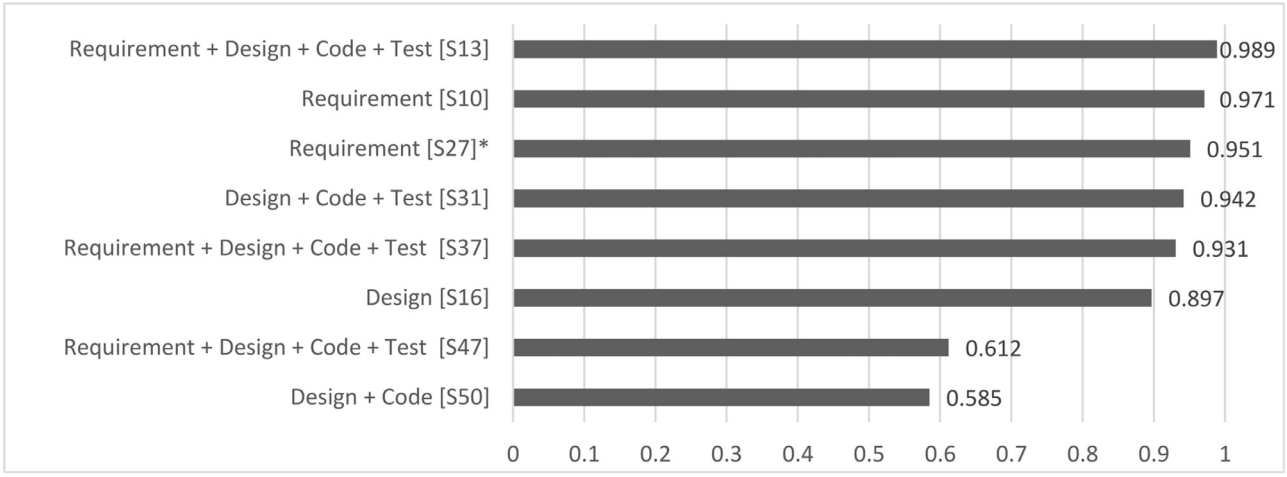


Fig. 23. Goodness-of-fit (R^2) values reported in continuous studies.

Table 12
Reported benefits of early software defect prediction.

Benefit ID	Benefits focus	Reported benefits	Primary studies	Number 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 lifecycle 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

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.

We can say 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.

RQ 2 - 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 for effective decision-support and trade-off analysis 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 lifecycle 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.

6. Potential threats to validity

The main threats to the validity of our study has been discussed under four categories based on a standard de-facto checklist adopted from Wohlin et al. (2012). We also discuss the steps that we took to mitigate or minimize the threats.

Internal validity

Threats to internal validity are influences that can affect the independent variable with respect to causality, without the researcher's knowledge (Wohlin et al., 2012).

The researcher bias is the main internal validity threat of this study since a single researcher conducted the inclusion/ exclusion of the papers and data extraction process. To reduce this threat, some actions have been taken. Firstly, the review protocol of the research was prepared with both authors 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 given in Table 1. 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 second author 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.

As for the search process, we did not include the terms such as “quality”, “error”, “bug” or “failure” in our search terms because this would lead to the examination of a wide range of out of scope papers, as we observed in the trial searches we initially made. We might have missed some papers that use these terms as a synonym for “defect” . However, we have added “fault” term to our search to mitigate this threat. Our work has some limitations due to its search string and selection criteria. One limitation is to search for the words “early” or “earlier” rather than phase-based words such as “requirements” or “design” . Another limitation is to search for the words “early” or “earlier” in titles of studies rather than title-abstract-keywords of the studies. These selections were part of the search strategy and decided after running several initial searches. Though we applied a complementary search strategy to cope with these limitations, we might have missed few primary studies by these selections. Another limitation is using of the phrases like "software defect" instead of using Boolean operators to combine the words like "software" AND "defect". We might also have missed some primary papers due to this limitation.

The most important threat in this study may be our 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. Though our 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.

In order to reduce the abovementioned threats related to the selected search string, we performed backward snowballing through the primary studies to elicit additional ones, and defined and applied inclusion/exclusion criteria specific to our study's purpose. We also included the references of the recently published systematic secondary studies, e.g., [18, 21] in the backward snowballing procedure. Furthermore, we run secondary search strings including phase-based words, e.g., requirements and design on Google Scholar as the meta-search engine, in case we have missed some primary studies due to our primary search string.

Construct validity

Construct validity is concerned with the relation between the study structure and the actual reflection of the research (Wohlin et al., 2012). Threats related to this type of validity might be suitability of research questions and classification scheme used for data extraction. We designed the MQs and RQs to address our research goal. Research questions were answered based on a classification scheme, which was designed based on some standards adapted from Petersen et al. (2008). Also, we finalized the scheme through several iterations until we extracted all related information with the research questions.

Conclusion validity

This type of validity is concerned with the relationship between the data collected from primary studies and the results/conclusions (Wohlin et al., 2012). 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 raw data (Ozakinci and Tarhan, 2017). In addition, we presented in Section 4 our analysis results as we generated directly from the data, and discussed in Section 5 the explicit findings and our observations in relation to these results.

External validity

The external validity is concerned with generalization of our study's conclusions (Wohlin et al., 2012). Our findings in this systematic review study are mainly valid within the specific area under study (i.e. early software defect prediction originated in requirements or design stages) and therefore cannot be generalized beyond this context.

7. Conclusions

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 software development lifecycle. Software prediction models can help in the early detection of software defect proneness. Although it is not easy 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 systematic review study, we investigated evidence on the trends and maturity of research as well as the success and usefulness of early software defect prediction. According to our findings; model/approach proposals and empirical case studies are the most contributed types, and weak empirical validation was the most reported research type. Few studies reporting evaluation research indicate the need for conducting stronger empirical validation for future contributions. 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.

By considering the findings that we obtained in this systematic review study, 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.

In this paper, we have elaborated on the studies that predict the software defects in the early lifecycle phases as requirements or design. Our work contributes to the scope of defect prediction area by demonstrating the up-to-date picture of the literature and the distinctive features of the early defect predictors. We expect that our work will be exemplary for future analysis of the effects, benefits, and contributions of early-phase defect predictors, and motivate new studies on ESDP area.

Appendix- Pool of primary studies

[S1] Amasaki, S., Takagi, Y., Mizuno, O., & Kikuno, T. (2003). A Bayesian belief network for assessing the likelihood of fault content. In 14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003. (pp. 215–226). http://doi.org/10.1109/ISSRE.2003.1251044.

[S2] Indu Sharma and Ms. ParveenBano, “A Combined Approach of Software Metrics and Software Fault Analysis to Estimate Software Reliability,” IOSR J. Comput. Eng., vol. 11, no. 6, pp. 01–14, 2013.

[S3] Sandhu, P. S., Kaur, M., & Kaur, A. (2010). A Density Based Clustering approach for early detection of fault prone modules. In 2010 International Conference on Electronics and Information Engineering (pp. V2-525-V2-530). IEEE. http://doi.org/10.1109/ICEIE.2010.5559753.

[S4] Kaur, A. (2011). A Framework for Analyzing Software Quality using Hierarchical Clustering. International Journal on Computer Science and Engineering (IJCSE), 3(2), 854–861.

[S5] V. Vashisht, M. Lal, and G. S. Sureshchandar, “A Framework for Software Defect Prediction Using Neural Networks,” J. Softw. Eng. Appl., vol. 8, pp. 384–394, 2015.

[S6] Bharathi, R., & Selvarani, R. (2015). A framework for the estimation of OO software reliability using design complexity metrics. In 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15) (pp. 1–7). IEEE. https://doi.org/10.1109/ITACT.2015.7492648.

[S7] Bahadur Yadav, H., & Kumar Yadav, D. (2015). A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Information and Software Technology, 44–57. https://doi.org/10.1016/j.infsof.2015.03.001.

[S8] Pandey, A., & Goyal, N. (2009). A fuzzy model for early software fault prediction using process maturity and software metrics. International Journal of Electronics …, 1(2), 239–245. Retrieved from http://www.csjournals.com/IJEE/PDF. 1–2/21_AJEET KUMAR PANDEY_N. K.GOYAL.pdf.

[S9] Sandhu, P. S., Goel, R., Brar, A. S., Kaur, J., & Anand, S. (2010). A model for early prediction of faults in software systems. In 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE) (pp. 281–285). http://doi.org/10.1109/ICCAE.2010.5451695.

[S10] Chatterjee, S., & Maji, B. (2016). A new fuzzy rule based algorithm for estimating software faults in early phase of development. Soft Computing, 20(10), 4023–4035. http://doi.org/10.1007/s00500-015-1738-x.

[S11] Parvinder S. Sandhu, Sunil Khullar, Satpreet Singh, Simranjit K. Bains, Manpreet Kaur, G. S. (2010). A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm. World Academy of Science, Engineering and Technology, (48), 648–653.

[S12] R. S. Sidhu, S. Khullar, P. S. Sandhu, R. P. S. Bedi, and K. Kaur, “A Subtractive Clustering Based Approach for Early Prediction of Fault Proneness in Software Modules,” World Acad. Sci. Eng. Technol., vol. 4, no. 7, pp. 1165–1169, 2010.

[S13] Dhiauddin, M., Suffian, M., & Ibrahim, S. (2013). A Systematic Approach to Predict System Testing Defects using Prior Phases Metrics for V-Model. Open International Journal of Informatics (OIJI), 1, 1–17.

[S14] Mohanta, S., Vinod, G., & Mall, R. (2011). A technique for early prediction of software reliability based on design metrics. International Journal of System Assurance Engineering and Management, 2(4), 261–281. http://doi.org/10.1007/s13198-011-0078-1.

[S15] Hong, Y., Baik, J., Ko, I. Y., & Choi, H. J. (2008). A value-added predictive defect type distribution model based on project characteristics. In Proceedings - 7th IEEE/ACIS International

Conference on Computer and Information Science, IEEE/ACIS ICIS 2008, In conjunction with 2nd IEEE/ACIS Int. Workshop on e-Activity, IEEE/ACIS IWEA 2008 (pp. 469–474). http://doi.org/10.1109/ICIS.2008.36.

[S16] M. Cartwright and M. Shepperd, “An empirical investigation of an object-oriented software system,” IEEE Trans. Softw. Eng., vol. 26, no. 8, pp. 786–796, 2000.

[S17] Kumar, K. S., & Misra, R. B. (2008). An Enhanced Model for Early Software Reliability Prediction Using Software Engineering Metrics. In 2008 Second International Conference on Secure System Integration and Reliability Improvement (pp. 177–178). http://doi.org/10.1109/SSIRI.2008.32.

[S18] Rana, Z. A., Awais, M. M., & Shamail, S. (2009). An FIS for early detection of defect prone modules. In Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence (Vol. 5755 LNAI, pp. 144–153). http://doi.org/10.1007/978-3-642-04020-7_16.

[S19] A. Nugroho, M. R. V Chaudron, and E. Arisholm, “Assessing UML design metrics for predicting fault-prone classes in a Java system,” in Proceedings - International Conference on Software Engineering, 2010, pp. 21–30.

[S20] Ma, Y., Zhu, S., Qin, K., & Luo, G. (2014). Combining the requirement information for software defect estimation in design time. Information Processing Letters (Vol. 114). https://doi.org/10.1016/j.ipl.2014.03.012.

[S21] Jiang, Y., Cuki, B., Menzies, T., & Bartlow, N. (2008). Comparing Design and Code Metrics for Software Quality Prediction. Proceedings of the 4th International Workshop on Predictor Models in Software Engineering, 12, 11–18. http://doi.org/10.1145/1370788.1370793.

[S22] Singh, P., & Verma, S. (2015). Cross Project Software Fault Prediction at Design Phase. International Journal of Computer, Electrical, Automation, Control and Information Engineering, 9(3), 800–805. https://doi.org/10.5370/JEET.2014.9.4.742.

[S23] V. Vashisht, M. Lal, and G. Sureshchandar, “Defect Prediction Framework Using Neural Networks for Software Enhancement Projects,” Br. J. Math. Comput. Sci., vol. 16, no. 5, pp. 1–12, Jan. 2016.

[S24] Pandey, A. K., & Goyal, N. K. (2013). Early Fault Prediction Using Software Metrics and Process Maturity. In Early Software Reliability Prediction (Vol. 303, pp. 35–57). http://doi.org/10.1007/978-81-322-1176-1.

[S25] R. Ratra, N. S. Randhawa, P. Kaur, and G. Singh, “Early Prediction of Fault Prone Modules using Clustering Based vs. Neural Network Approach in Software Systems,” Int. J. Electron. Commun. Technol., vol. 2, no. 4, pp. 47–50, 2011.

[S26] MOHAN, K. K., VERMA, A. K., & SRIVIDYA, A. (2009). Early Qualitative Software Reliability Prediction and Risk Management in Process Centric Development Through a Soft Computing Technique. International Journal of Reliability, Quality and Safety Engineering, 16(6), 521–532. http://doi.org/10.1142/S0218539309003551.

[S27] Yadav, D. K., Charurvedi, S. K., & Mishra, R. B. (2012). Early Software Defects Prediction Using Fuzzy Logic. International Journal of Performability Engineering, 8(4), 399–408.

[S28] Kaur, a., Sandhu, P. S., & Bra, A. S. (2009). Early Software Fault Prediction Using Real Time Defect Data. In 2009 Second International Conference on Machine Vision (pp. 242–245). http://doi.org/10.1109/ICMV.2009.54.

[S29] Yang, B., Yao, L., & Huang, H.-Z. (2007). Early Software Quality Prediction Based on a Fuzzy Neural Network Model. In Third International Conference on Natural Computation (ICNC 2007) (Vol. 1, pp. 760–764). IEEE. http://doi.org/10.1109/ICNC.2007.347.

[S30] Yadav, H. B., & Yadav, D. K. (2014). Early software reliability analysis using reliability relevant software metrics. International Journal of System Assurance Engineering and Management. http://doi.org/10.1007/s13198-014-0325-3.

[S31] Yamada, S.: Early-stage software product quality prediction based on process measurement data. In: Misra, K.B. (ed.) Handbook of Performability Engineering, pp. 1227–1237. Springer, London (2008).

[S32] Pandey, A. K., & Goyal, N. K. (2010). Fault Prediction Model by Fuzzy Profile Development of Reliability Relevant Software Metrics. International Journal of Computer Applications, 11(6), 34–41. http://doi.org/10.5120/1584 – 2124.

[S33] Jiang, Y., Cukic, B., & Menzies, T. (2007). Fault Prediction using Early Lifecycle Data. In 18th IEEE International Symposium on Software Reliability Engineering (pp. 237–246). http://doi.org/10.1109/ISSRE.2007.24.

[S34] Pandey, A. K., & Goyal, N. K. (2013). Multistage Model for Residual Fault Prediction. In Early Software Reliability Prediction (Vol. 303, pp. 117–130). https://doi.org/10.1007/978-81-322-1176-1.

[S35] Sandhu, P. S., Lata, S., & Grewal, D. K. (2012). Neural Network Approach for Software Defect Prediction Based on Quantitative and Qualitative Factors. International Journal of Computer Theory and Engineering, 4(2), 298–303.

[S36] D. L. Gupta and A. K. Malviya, “Observations on Fault Proneness Prediction Models of Object-Oriented System to Improve Software Quality,” Int. J. Adv. Res. Comput. Sci., vol. 2, pp. 57–65, 2011.

[S37] "Fenton, N., Neil, M., Marsh, W., Hearty, P., Radlinski, L., & Krause, P. (2008). On the effectiveness of early life cycle defect prediction with Bayesian nets. Empirical Software Engineering, 13(5), 499–537. http://doi.org/10.1007/s10664-008-9072-x.

[S38] Zimmermann, T., & Nagappan, N. (2009). Predicting defects with program dependencies. In 2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009 (pp. 435–438). http://doi.org/10.1109/ESEM.2009.5316024.

[S39] Sehgal, R., & Mehrotra, D. (2015). Predicting faults before testing phase using Halstead's metrics. International Journal of Software Engineering and Its Applications, 9(7), 135–142. http://doi.org/10.14257/ijseia.2015.9.7.14.

[S40] Ba, J., & Wu, S. (2012). ProPRED: A probabilistic model for the prediction of residual defects. In Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (pp. 247–251). http://doi.org/10.1109/MESA.2012.6275569.

[S41] S. Bibi, G. Tsoumakas, I. Stamelos, and I. Vlahavas, “Regression via Classification applied on software defect estimation,” Expert Syst. Appl., vol. 34, no. 3, pp. 2091–2101, 2008.

[S42] Jiang, Y., Lin, J., Cukic, B., Lin, S., & Hu, Z. (2013). Replacing Code Metrics in Software Fault Prediction with Early Life Cycle Metrics. In Third International Conference on Information Science and Technology (pp. 516–523). http://doi.org/10.1109/SCC.2014.108.

[S43] Kumar, C., & Yadav, D. K. (2014). Software defects estimation using metrics of early phases of software development life cycle. International Journal of System Assurance Engineering and Management, 1–9. http://doi.org/10.1007/s13198-014-0326-2.

[S44] Singh, P., Verma, S., & Vyas, O. P. (2014). Software Fault Prediction at Design Phase. Journal of Electrical Engineering and Technology, 9(5), 742–748. http://doi.org/10.5370/JEET.2014.9.4.742.

[S45] H. Euyseok, “Software Fault-proneness Prediction using Random Forest,” Int. J. Smart Home, vol. 6, no. 4, pp. 147–152, 2012.

[S46] Kumar, C., & Yadav, D. K. (2013). Software Quality Modeling using Metrics of Early Artifacts. In Confluence 2013: The Next Generation Information Technology Summit (4th International Conference) (pp. 7–11). http://doi.org/10.1049/cp.2013.2285.

[S47] Lee, W., Lee, J. K., & Baik, J. (2011). Software Reliability Prediction for Open Source Software Adoption Systems Based on Early Lifecycle Measurements. In 2011 IEEE 35th Annual Computer Software and Applications Conference (pp. 366–371). http://doi.org/10.1109/COMPSAC.2011.55.

[S48] Wajahat, S., Rizvi, A., Khan, R. A., & Singh, V. K. (2016). Software Reliability Prediction using Fuzzy Inference System: Early

Stage Perspective. International Journal of Computer Applications, 145(10), 975–8887.

[S49] Kläs, M., Nakao, H., Elberzhager, F., & Münch, J. (2010). Support planning and controlling of early quality assurance by combining expert judgment and defect data- A case study. Empirical Software Engineering, 15(4), 423–454. http://doi.org/10.1007/s10664-009-9112-1.

[S50] Tomaszewski, P., Håkansson, J., Lundberg, L., & Grahn, H. (2005). The Accuracy of Early Fault Prediction in Modified Code. In Proceedings of the Fifth Conference on Software Engineering Research and Practice in Sweden (SERPS) (pp. 57–63). http://doi.org/10.1109/ECBS.2006.68.

[S51] Emam, K. El, Melo, W., & Machado, J. C. (2001). The Prediction of Faulty Classes Using Object-Oriented Design Metrics. Journal of Systems and Software (JSS), 56(1), 63–75.

[S52] Khoshgoftaar, T. M., & Seliya, N. (2002). Tree-based software quality estimation models for fault prediction. In Proceedings Eighth IEEE Symposium on Software Metrics (pp. 203–214). http://doi.org/10.1109/METRIC.2002.1011339.

References

Aydin, A, Tarhan, A, 2014. Investigating defect prediction models for iterative software development when phase data is not recorded lessons learned. In: Proceedings of the Ninth International Conference Evaluation of Novel Approaches to Software Engineering, pp. 1–11.

Catal, C, Diri, B, 2009. A systematic review of software fault prediction studies. Expert Syst. Appl. 36, 7346–7354. http://dx.doi.org/10.1016/j.eswa.2008.10.027.

Catal, C, 2011. Software fault prediction: a literature review and current trends. Expert Syst. Appl. 38, 4626–4636. http://dx.doi.org/10.1016/j.eswa.2010.10.024.

Chidamber, SR, Kemerer, CF, 1994. A metrics suite for object oriented design. IEEE Trans. Softw. Eng. 20, 476–493. http://dx.doi.org/10.1109/32.295895.

Cukic, B, Hayes, JH, 2005. The virtues of assessing software reliability early. IEEE Softw. 22, 50–53. http://dx.doi.org/10.1109/MS.2005.79.

Dixon-Woods, M, Agarwal, S, Jones, D, et al., 2005. Synthesising qualitative and quantitative evidence: a review of possible methods. J. Heal Serv. Res. Policy 10, 45–53. http://dx.doi.org/10.1258/1355819052801804.

Fenton, N, Bieman, J, 2014. Software Metrics: A Rigorous and Practical Approach, Third Edition (3rd ed.). CRC Press, Inc., Boca Raton, FL, USA.

Fenton, NE, Ohlsson, N, 2000. Quantitative analysis of faults and failures in a complex software system. IEEE Trans. Softw. Eng. 26, 797–814.

Florac WA, Park RE, Carleton A, 1997. Practical software measurement: measuring for process management and improvement, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-96-HB-003, April 1997.

Hall, T, Beecham, S, Bowes, D, et al., 2012. A systematic literature review on fault prediction performance in software engineering. IEEE Trans. Softw. Eng. 38, 1276–1304. http://dx.doi.org/10.1109/TSE.2011.103.

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

IEEE Recommended Practice on Software Reliability, in *IEEE Std 1633-2016 (Revision of IEEE Std 1633-2008)*, vol., no., pp. 1–261, Jan. 18 2017 doi: 10.1109/IEEESTD.2017.7827907.

IEEE (2009) IEEE standard classification for software anomalies (IEEE 1044 - 2009). IEEE Std. 1044: 1 – 4.

IEEE Standard for System, Software, and Hardware Verification and Validation, in *IEEE Std 1012-2016 (Revision of IEEE Std 1012-2012/ Incorporates IEEE Std 1012-2016/ Cor1-2017)*, pp. 1–260, Sept. 29 2017, doi: 10.1109/IEEESTD.2017.8055462.

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

Kan, SH, 2003. Metrics and Models in Software Quality Engineering, second ed. Addison Wesley.

Kitchenham, B, Charters, S, 2007. Guidelines for performing systematic literature reviews in software engineering. Engineering 2, 1051. http://dx.doi.org/10.1145/1134285.1134500.

Kitchenham, B, Pearl Brereton, O, Budgen, D, et al., 2009. Systematic literature reviews in software engineering - a systematic literature review. Inf. Softw. Technol. 51, 7–15. http://dx.doi.org/10.1016/j.infsof.2008.09.009.

Kläs, M, Nakao, H, Elberzhager, F, Münch, J, 2008. Predicting defect content and quality assurance effectiveness by combining expert judgment and defect data - A case study. In: Proceedings of the International Symposium on Software Reliability Engineering, pp. 17–26. http://dx.doi.org/10.1109/ISSRE.2008.43.

Malhotra, R, 2015. A systematic review of machine learning techniques for software fault prediction. Appl. Soft Comput. 27, 504–518.

Menzies, T, Krishna, R, Pryor, D, 2016. The promise repository of empirical software engineering data. http://openscience.us/repo.

Murillo-Morera, J, Quesada-López, C, Jenkins, M, 2015. Software fault prediction: a systematic mapping study. In: Proceedings of the XVIII Iberoamerican Conference on Software Engineering.

Nagappan, N, Ball, T, 2005. Static analysis tools as early indicators of pre-release defect density. In: Proceedings of the IEEE Twenty Seventh International Conference Software Engineering, pp. 580–586.

Nagappan, N, Ball, T, Murphy, B, 2006. Using historical in-process and product metrics for early estimation of software failures. In: Proceedings of the Seventeenth International Symposium on Software Reliability Engineering, pp. 62–74.

Ozakinci, R, Tarhan, A, 2016. The role of process in early software defect prediction: methods, attributes and metrics. Commun. Comput. Inf. Sci. 609, 287–300. http://dx.doi.org/10.1007/978-3-319-38980-6_21.

Ozakinci, R, Tarhan, A, 2017. Online paper repository for the systematic mapping and review on “Early Software Defect Prediction. https://tinyurl.com/y86nz4fp

Hu, Qp., Dai, Y, Xie, M, Ng, Sh., 2006. Early software reliability prediction with extended ANN model. In: Proceedings of the IEEE Thirtieth Annual International Computer Software Applied Conference, pp. 234–239.

Pandey AK, Goyal NK (2013) Early software reliability prediction. 303:105–115. doi: 10.1007/978-81-322-1176-1.

Petersen, K, Wohlin, C, 2009. Context in industrial software engineering research. In: Proceedings of the Third International Empirical Software Engineering and Measurement, pp. 401–404.

Petersen, K, Feldt, R, Mujtaba, S, Mattsson, M, 2008. Systematic mapping studies in software engineering. In: Proceedings of the Twelfth International Conference Evaluation and Assessment in Software Engineering, pp. 68–77.

Petersen, K, Vakkalanka, S, Kuzniarz, L, 2015. Guidelines for conducting systematic mapping studies in software engineering: an update. Inf. Softw. Technol. 64, 1–18. http://dx.doi.org/10.1016/j.infsof.2015.03.007.

Petersen, K, 2011. Measuring and predicting software productivity: a systematic map and review. Inf. Softw. Technol. 53, 317–343. http://dx.doi.org/10.1016/j.infsof.2010.12.001.

Radjenovic, D, Hericko, M, Torkar, R, et al., 2013. Software fault prediction metrics: a systematic literature review. Inf. Softw. Technol. 55, 1397–1418. http://dx.doi.org/10.1016/j.infsof.2013.02.009.

Singh, PK, Agarwal, D, Gupta, A, 2015. A systematic review on software defect prediction. In: Proceedings of the Second International Conference on Computing for Sustainable Global Development, pp. 1793–1797.

Singh, LK, Vinod, G, Tripathi, AK, 2016. Early prediction of software reliability: a case study with a nuclear power plant system. Comput. (Long Beach Calif) 49, 52–58. http://dx.doi.org/10.1109/MC.2016.15.

Smidts, C, Stutzke, M, Stoddard, RW, 1998. Software reliability modeling: an approach to early reliability prediction. IEEE Trans. Reliab. 47, 268–278. http://dx.doi.org/10.1109/24.740500.

Song, Q, Jia, Z, Shepperd, M, et al., 2011. A general software defect-proneness prediction framework. IEEE Trans. Softw. Eng. 37, 356–370. http://dx.doi.org/10.1109/TSE.2010.90.

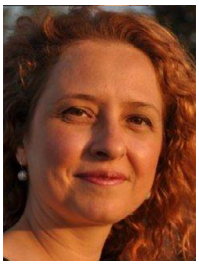
Wahono, RS, 2015. A systematic literature review of software defect prediction : research trends, datasets. Methods Frameworks. J. Softw. Eng. 1, 1–16.

Wohlin, C, Runeson, P, Höst, M, et al., 2012. Experimentation in software engineering. Exp. Softw. Eng. http://dx.doi.org/10.1007/978-3-642-29044-2.

Wohlin, C, 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE), pp. 1–10. http://dx.doi.org/10.1145/2601248.2601268.

Zimmermann, T, Nagappan, N, Gall, H, et al., 2009. Cross-project defect prediction. In: Proceedings of the Seventh European Software Engineering Conference and Symposium on Foundations of Software Engineering, http://dx.doi.org/10.1145/1595696.1595713.

Rana Özakıncı works as a senior software engineer at Turkish Notaries Union in Ankara. She received her B.Sc. in computer engineering in 2011 from the Department of Computer Engineering at Hacettepe University, where she is currently studying to receive her Ph.D. degree with a focus on software engineering and defect prediction.



Ayca Tarhan works as a researcher and practitioner in the area of software engineering for fifteen years. She is experienced in model-based assessment and improvement of software processes. She pursues her studies with a focus on software quality, software development methodologies, software measurement, business processes, and process management. She completed her PhD in Information Systems at Middle East Technical University, and currently works as an Assistant Professor in Computer Engineering Department of Hacettepe University, Ankara.