Social Metrics included in Prediction Models on Software Engineering: A Mapping Study

Igor Scaliante Wiese, Filipe Roseiro Côgo,
Reginaldo Ré, Igor Steinmacher
Computer Science Department
Federal University of Technology - Parana (UTFPR)
{igor, filiper, reginaldo.re, igorfs}@utfpr.edu.br

Marco Aurélio Gerosa,
Department of Computer Science
University of São Paulo (USP)
gerosa@ime.usp.br

ABSTRACT

Context: Previous work that used prediction models on Software Engineering included few social metrics as predictors, even though many researchers argue that Software Engineering is a social activity. Even when social metrics were considered, they were classified as part of other dimensions, such as process, history, or change. Moreover, few papers report the individual effects of social metrics. Thus, it is not clear yet which social metrics are used in prediction models and what are the results of their use in different contexts. **Objective:** To identify, characterize, and classify social metrics included in prediction models reported in the literature. **Method:** We conducted a mapping study (MS) using a snowballing citation analysis. We built an initial seed list adapting strings of two previous systematic reviews on software prediction models. After that, we conducted backward and forward citation analysis using the initial seed list. Finally, we visited the profile of each distinct author identified in the previous steps and contacted each author that published more than 2 papers to ask for additional candidate studies. **Results:** We identified 48 primary studies and 51 social metrics. We organized the metrics into nine categories, which were divided into three groups - communication, project, and commit-related. We also mapped the applications of each group of metrics, indicating their positive or negative effects. **Conclusions:** This mapping may support researchers and practitioners to build their prediction models considering more social metrics.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics--*Process metrics*

General Terms

Measurement, Experimentation, Human Factors

Keywords

Mapping study, prediction models, social metrics, social network analysis.

1. INTRODUCTION

Software development is an inherently sociotechnical endeavor, especially, because of the collaboration and communication that take place among stakeholders [7, 33]. Repositories have introduced many social tools to facilitate the interaction among developers. This phenomenon is known as "social coding" and Github is an example of these repositories. Therefore, human factors considerably influence software development.

Previous studies in the literature considered few social metrics to build prediction models compared to the number of process metrics [9, 13, 18, 28, 31]. Hall *et al.* [13] highlights that there are few studies that consider developer information in prediction models and it is difficult to know the effectiveness of social metrics.

A possible explanation for this scenario can be related to the social metrics classification. Most part of the previous works considered social metrics as part of other dimension, and the performance of each predictor is frequently not discussed. For example, "number of developers" metric was used by researchers as part of different dimensions, such as change metrics [23], people metrics [25], developer information [24], team [12], developer activity [30], project level [35], and process metrics [10].

In this paper, we used the term "social metrics" to refer to any metric that measures aspects of the interactions between developers. For example, we consider as social metrics the number of comments, number of distinct authors that committed a file or metrics from the social networks extracted from these interactions. We also considered aspects related to developers' skill, like experience and ownership as part of social dimension. This definition is broader than the definition of *Social Dimension* proposed by Ibrahim *et al.* [14]. For them, the *Social Dimension* comprises metrics that capture the communication activity between developers and measures the impact of inter-personal relations.

We conducted a mapping study, following the snowballing method proposed by Webster and Watson [34], to identify, characterize, and classify the use of social metrics in prediction models. We chose this approach, because it offers a good coverage at the same time that it is less influenced by the amount of noise from digital libraries searches [16].

By conducting this mapping study, we aimed to answer two main questions: RQ1: Which social metrics were used in prediction models? and RQ2: Did the social metrics have positive effect when they were considered as predictors?

In summary, the main contributions of this study were:

- The conduction a secondary study to **summarize the current state of research on the use of social metrics in prediction models.** We described the effect of social metrics usage for each paper found in the literature. We described the applicability, prediction models techniques, and social metrics (Section 4).
- **A classification of social metrics** in three groups and nine categories. We classified twenty-one metrics as communication,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PROMISE '14, September 17 2014, Torino, Italy
Copyright 2014 ACM 978-1-4503-2898-2/14/09…$15.00.
http://dx.doi.org/10.1145/2639490.2639505.

eighteen metrics as project, and twelve metrics as commit-related. We also found forty-five social network analysis metrics on eight types of social networks (Section 4).

- **The comparison of the current study to six previous systematic literature reviews** showing differences and updating the survey of the state of the art in prediction models (Section 2).
- **The discovery of areas that require further attention,** presenting opportunities for new researches considering the use of social metrics in prediction models on software engineering (Section 6).

2. RELATED WORK

We identified six systematic literature reviews related to our study. Table 1 presents these studies reporting whether they mentioned social metrics.

Table 1. Previous systematic literature reviews related to prediction models

Study	*Focus*	*Results and mentions of social metrics*
Hall *et al.,*2012 [13]	Fault prediction models	The models that perform well use simple modelling techniques and combination of indepent variables. **Few papers considered social metrics.**
Radjenovic *et al*., 2013 [28]	Fault prediction metrics	Object-oriented and process metrics were reported to be more successful in finding faults compared to traditional size and complexity metrics. Process metrics seem to be better at predicting post-release faults compared to any static code metric. Like Hall et al.[13], they reported that developer information was used to build prediction models. **Just 8 papers reported the use of social metrics.**
Jureczko and Madeyski, 2011 [18]	A review of process metrics	Taxonomy of process metrics with five metrics. **The only social metric identified was number of distinct commiters.**
Catal, 2011 [9] and Catal and Diri, 2009 [8]	Current trend of fault prediction models	Few papers using process metrics was reported. **Social metrics were not discussed.**
Riaz *et al*., 2009 [31]	Maintainability prediction and metrics	Predictors reported were based on size, complexity, and coupling, and were gathered at source code level. **Social metrics were not discussed.**
Jorgensen and Shepperd [17]	Cost estimation	Provide a basis for the improvement of software estimation research. **Social metrics were not discussed.**
Azhar *et al.,* [2]	Web resource estimation	The aim of this paper is to present a SR of Web resource estimation in order to define the current state of the art. **Social metrics were not discussed.**

Jureczko and Madeyski [18] discussed the difference between product and process metrics. They described five process metrics: Number of Revisions, Number of Distinct Committers, Number of Modified Lines, Is New, and Number of Defects in Previous Revision. Number of distinct committers was presented as a "developer related-metric."

Hall *et al*. [13] updated Catal´s work investigating the context of model, the independent variables, and the modelling techniques used to build fault prediction models. As a result, they reported 36 studies. The models that perform well tended to be based on simple modeling techniques such as Naïve Bayes or Logistic Regression and have used combinations of independent variables. Feature selection has been applied to these combinations in order to discover the performance of each individual metric. They also reported many different types of independent variables. Considering process metrics, Hall *et al*. [13] mentioned that *few studies using developer information in models report conflicting results*. Our work complements Hall *et al.* systematic review, showing the effectiveness of social metrics in prediction models and focusing on the distinction of process metrics and social metrics.

Radjenovic *et al*. [28] presented a SLR considering 106 papers published between 1991 and 2011. The selected papers were classified according to metrics and context properties. They found that object-oriented metrics (49%) were used nearly twice as often when compared to traditional source code metrics (27%) or process metrics (24%). Chidamber and Kemerer's (CK) object-oriented metrics were most frequently used. Radjenovic *et al*. found 8 papers that used developer information to build prediction models. They emphasized that the usability of *developer information in fault prediction remains an important unanswered research question*.

Our mapping study differs from these related ones since we are focusing on social metrics used as predictors and we were not restricted to a specific goal of the prediction. Previous reviews have focused mainly in fault prediction. Nevertheless, we were able to find in our initial seed all the papers mentioned in these literature reviews that used at least one social metric.

3. METHODOLOGICAL APPROACH

This section presents the methodological approach to conduct this mapping study. First, we discuss each research question. After, we show the review method, inclusion/exclusion criteria, and seed validation analysis. Finally, we present a summarization of the data extraction.

3.1 Research Questions

We defined the following research questions:

RQ1: Which social metrics were used in prediction models? We wanted to investigate this question since the previous systematic literature reviews (Section 2) did not discuss which social metrics were considered as predictors to build prediction models and primary works use an inconsistent terminology for classifying social metrics and often do not report their individual result.

RQ2: Did the social metrics have positive effect when they were considered as predictor? By answering this question, we aimed to show the implications of social metrics to build prediction models on software engineering. We used three different analysis to show the effectiveness of social metrics. First, we identified papers describing evidences about the effects of social metrics. After that, we summarized the proposed classification, linking each group of metrics to the applicability of prediction models. Finally, we mapped in which application each group of social metrics were used so far. Our intention was to provided an overview of social metrics, papers, prediction techniques, and their applicability.

3.2 Review Method

A software engineering systematic mapping is defined as a method to build a classification scheme and structure a field of interest [26]. This kind of study follow the systematic review guidelines, but as a result are reported the frequency of publications for categories within

a proposed scheme. Thus, the coverage of the research field can be analyzed and different facets of the scheme can be combined to answer more specific research questions.

Systematic studies of the literature can be conducted following different guidelines or methods. Two methods are commonly used. Kitchenham and Charters [4] focus on systematic searches in databases using well-defined search strings to find relevant papers.

Webster and Watson [34] proposed the use of snowballing citation analysis as the main method to find relevant literature. Jalali and Wohlin [16] compared snowballing and search method to do systematic literature studies and did not find any remarkable differences between the results.

We applied the search method, following Kichenham and Charters [4] guideline. We used the string[1] extended from Hall *et al.* [13] and Radjenovic *et al.* [28] to find the initial list of candidate papers. Since we had many general terms, we were not able to lead with the amount of unrelated results returned using the search method.

Jalali and Wohlin [16] showed that snowballing might be more efficient when the keywords for searching include general terms. Webster and Watson [34] claim that this method avoid the amount of noise returned by exhaustive database searches. They also suggest that researchers should apply backward and forward snowballing citation analysis.

Following these recommendations, we used the snowballing to perform the systematic literature review. The Figure 1 shows the steps of our research. We conducted five main steps to select papers. In the first step, we used the adapted string from Hall *et al.* [13] and Radjenovic *et al.* [28] to produce the initial seed based on searches on ACM, IEEE, and Scopus.

In these two initial steps, we considered initially the title and abstracts. When they did not bring enough information to judge the criteria for inclusion and exclusion, the full text was also considered. We just excluded a paper from our seed list if it was clearly out of scope, considering the exclusion criteria.

We kept all potential primary studies for further analysis, because we knew that social metrics could be listed on different dimensions of software development and had different names. We used the inclusion/exclusion criteria to guide the first discussion between two researchers to define the agreement or disagreement for each selected paper. Using this candidate list, the same two authors inspected each paper applying the quality criteria listed on Table 2 to generate the seed list of papers.

Using the seed list, we performed backward analysis (step 2) by reviewing the reference lists to find new relevant paper and conducted a forward analysis (step 3) by identifying articles that cited papers of our selected paper list.

Once again, we used the inclusion/exclusion criteria to guide the discussion between two researchers to define the agreement or disagreement for each paper found on backward and forward steps. To include a paper on a backward or forward list two authors discussed about each paper using the quality criteria. We searched for new papers checking the DBLP profile of each author identified on the seed, backward, or forward list. As described above, we evaluate each paper found using the quality assessment checklist (Table 2). We follow the Dybå and Dingsoyr´s checklist [11] to evaluate if a paper would stay in our final list of papers.

Finally, to confirm the results, we used a validation step, suggested by Kitchenham and Brereton [19]. We sent e-mails to all the researchers that wrote more than two papers in our final list of papers. We used the answers to check possible flaws while performing the previous four steps.

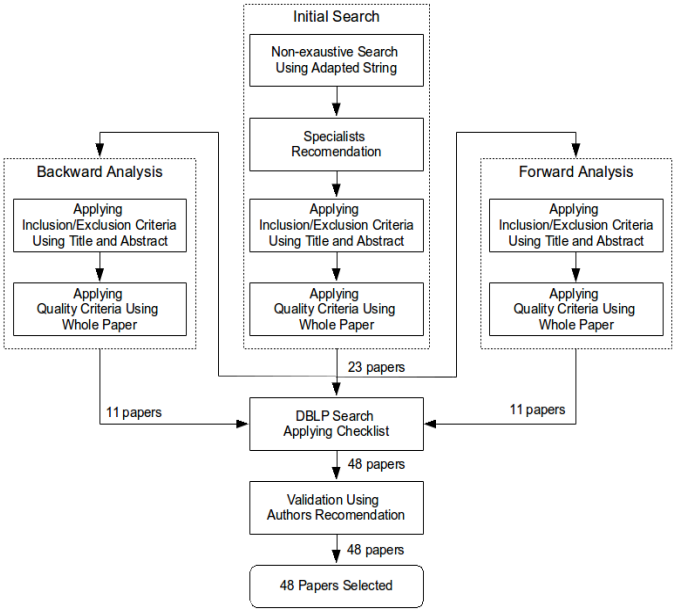

Figure 1. Snowballing process used in our study

3.3 Inclusion/Exclusion criteria

To analysis each paper, we used the inclusion and exclusion criteria to select candidate papers. We include all papers that used prediction models for Software Engineering aspects (e.g. fault, change proneness, bug, vulnerability, and effort) and used metrics not related to source-code code. We choose this generic criterion because many times authors mention about process, history, change, or organizational metrics without the term "social."

We were attentive not to exclude a relevant paper, considering that social metric could be part of these dimensions (group of metrics). Papers were included also based on the following inclusion criteria:

- Papers from journals, conferences, or workshops; and
- with at least one metric that could be classified as "social metric."

Papers were excluded from this study also based on the following exclusion criteria:

- Papers that explicitly mention only code metrics, objected oriented metrics, or static metrics
- Papers not written in English.
- Papers with full-text not accessible.
- Papers that were a preliminary or shorter version of the study published in other paper.

[1] The query string used was (software OR "open source" OR repository OR repositories) AND (fault OR effort OR defect OR quality OR error-prone OR error-proneness OR failure OR error OR prone) AND (metric OR metrics OR measurement OR measure OR measuring OR social OR socio-technical OR "communication network" OR "developer network" OR "developer information" OR "developer interaction" OR "human factor") AND (predict* OR estimat* OR classificat* OR regression)

Table 2 presents the eight questions to validate the quality of each selected paper. Papers needed to receive "YES" on the first five questions to remain in the final list. Questions 6, 7, and 8 did not exclude papers from the final list.

Table 2. Quality criteria to include a paper

Problem Statement
1. Is the objective of the research is explained and well-motivated?
Data Collection
2. Is there at least one metric that was computed considering some social aspect? (developer, communication, collaboration, experience, ownership, etc.)
3. Are the metrics appropriately described?
4. Is it possible to reproduce the social metric?
Research Design
5. Is there any type (classification or statistical) of prediction model used?
Data Analysis
6. Are there evaluation metrics applied on prediction models to evaluate the prediction results?
7. Are the results explained using feature selection algorithms or method to compare metrics to each other?
Conclusion
8. Does the paper discuss limitations or validity?

3.4 Seed validation analysis

To validate the list of papers obtained, we sent e-mails for all researchers that produced more than two primary studies. We sent 21 emails and received 10 answers. The specialists recommended 26 authors. Thereby, 11 were new authors and 15 had already been checked during the step four of our method. For each of these 11 new authors, we visited the DBLP page and did not find any paper to include in our final list of paper. All of them had works that considered social aspects, but they did not used prediction models.

3.5 Summarization of Data Extraction

This section summarizes the data extraction of each paper selected. Table 3 presents the summarization of 48 primary studies selected during each step of our method. We counted the number of papers published in conferences or workshops (#inc), the number of papers published on journals (#jou), and the number of papers rejected (#rej). Considering the number of rejected papers during the backward and forward citation analysis, we report the distinct number of papers rejected, instead of the total number of references found and rejected.

Table 3. Summary of number of primary studies included (#inc / #jou) and rejected (#rej)

Seed			Backward			Forward			DBLP		
#inc	#jou	#rej	#inc	# jou	#rej	#inc	#jou	#rej	#inc	#jou	#rej
19	4	48	6	5	419	9	2	212	1	2	34

Most part of the papers included in our final list were identified in this step. The backward and forward steps returned similar number of papers, but considering #rej, backward excluded many more papers. Just 3 papers were included after the visit to the DBLP page of each author, all the others relevant papers were already covered in the previous steps.

Considering the excluded papers, 5 of them were excluded from our seed list during the quality criteria analysis (Table 2). The backward and forward list had 1 paper removed by quality criteria. All papers from DBLP list were removed by exclusion and quality criteria.

Many papers were excluded because they presented social metrics, but they did not used prediction models. For example, we found papers that the main objective was to investigate the evolution of software communities, problems related to global software engineering, how to find mentors to help newcomers, or tools to visualize social interaction in software engineering.

We selected 48 primary studies in our final list of papers. These papers were produced by 103 distinct authors. The seed step identified 55 authors. The backward analysis identified 30 authors, 17 of them appeared in the seed step. The forward step included 38 distinct authors, 4 of them appeared in our seed author list, and 3 of them appeared in our backward author list. Considering the papers found in DBLP, we included more 6 authors. Five of them appeared in our seed author list, 5 on backward author list, and 2 of them in the forward author list.

Regarding venue of publication, 35 papers were published in conferences and workshops and 13 in journals. We found papers from the International Conference on Software Engineering (ICSE) with 10 papers between 2008 to 2013, International Symposium on the Foundations of Software Engineering (FSE) with 6 papers published in 2008, 2010, 2011, and 2012; and International Conference on Predictive Models in Software Engineering (PROMISE) with 5 papers published in 2007, 2010, 2011 (2), 2012. The main journals identified were the Empirical Software Engineering (SPRINGER), Transactions on Software Engineering (IEEE), Information and Software Technology (ELSEVIER) and Journal of Systems and Software (ELSEVIER). Just one journal paper was published in 2000; all others were published between 2008 to 2013.

4. SOCIAL METRICS IN PREDICTION MODELS ON SOFTWARE ENGINEERING

In this section, we discuss which social metrics were used in prediction models (Section 4.1) and how the literature classified the social metrics (Section 4.1.1). Aiming at answering the research questions, we present a classification schema that maps the use of social metrics in prediction models on software engineering (Section 4.1.1.1). We provide the complementary material used in our analysis as appendices. This material presents the metrics, effects, complete references of each paper selected and quality criteria analysis. These appendices are available at https://github.com/igorwiese/promise14.

4.1 RQ1: Which social metrics were used in prediction models?

We identified 48 primary studies and 51 social metrics. We noticed that depending on the design of the study, the same metric could be computed by different ways and aggregations. For example, the number of distinct developer was computed using the number of cumulative distinct developers considering the whole history of the project, a specific timeframe, previous release, or the same release that the analysis was performed. Considering the aggregations, we found the mean, maximum, minimum value, or entropy as ways to aggregate the value of one metric to specific timeframe analysis.

The number of distinct developers that committed a file was the metric that presented the highest number of different classifications (11), being process as the most common. This metric also was used with 18 different names and appeared in 32 different papers. This illustrates that there is no standard in terms of classification and terminology.

Considering the amount of times that each metric was cited we highlight that *amount of major contributor* to specific file (5 papers), *developer experience* (4 papers), *reputation of issue reporter* (4 papers), *amount of messages in a discussion* (4 papers), and *amount of words in a discussion* (4 papers) were frequently considered as a predictor.

To summarize which were these metrics, we proposed a classification for the use of social metrics in prediction models. We discuss and present the mapping of social metrics and papers in Section 4.1.2.

4.1.1 RQ1.1: How are the social metrics classified, considering the set of metrics used to build prediction models?

In this paper, we used the term "social metrics" to refer to any metric that measures aspects of the interactions between developers. Since these interactions happen in the context of a software process, previous work considered social metrics as part of process metrics [15] or other dimensions.

Process metrics reflect the changes over time, e.g. the number of code changes [18]. Recently, the term historical metrics is sometimes used instead of process metrics. D'Ambros *et al*. [10] define that process metrics are extracted from the versioning system, assuming that frequently changed files are part of process metrics.

Arisholm *et al*. [1] argue that process metrics require records of detailed information about developers work (e.g., changes and fault corrections, developer information, time of changes, whether a change passed certain test procedures, etc.). For example, they considered experience measures of each developer performing each change and the number of developers that have made changes to a file as process metrics.

D´Ambros *et al*. [10] classified approaches for predicting defects. Three papers selected in our final list of papers were included in an "*other approaches*" category by this author. Two of them [3, 36] are exclusively related to social metrics, and the third paper explored developer-module networks to predict defects [27].

We found two different interpretations to process metrics. Moser *et al*. [10] used a set of metrics based in "*file-centered*" process metrics, like number of revisions, number of times that the file was refactored, lines added and removed from each file, change set size, and age of each file. The unique social metric extracted from this study was number of authors. Rahman and Devanbu [29] presented more "*human-centered*" process metrics. The authors explored 14 process metrics, 9 of them having social metrics related. For example, non-social process metrics were number of lines added and removed from each file and number of commits made to a file. Metrics like number of *active developer* and *owner´s experience* can reflect the social side of process metrics.

On the other hand, instead of using process metrics, some authors considered different "*dimensions*" of software development. For example, Shihab *et al.* [32] considered time, size, code, file, purpose, and personnel aspects to predict risk changes. We selected *Developer Experience* from *Personnel Dimension* as social metric in our list. One of the most cited classification was "*organizational metrics*." This category grouped metrics related to *ownership*, *authorship experience,* and *roles* of developers. Normally, these metrics were used by studies that analyzed data provided by companies. A good example of this classification were found in Mockus and Nagappan *et al.* [22, 24].

We also found papers that considered different types of social networks. *Developer* and *Communication networks* were used by Meneely *et al.* [20], Bird *et. al* [7], Wolf *et al.* [36], and Biçer *et al.* [6]. In these studies, the social interactions were recovered when developers committed on the same file or commented on the same issue/work item. The social metrics in these cases were computed using social network analysis (SNA). We found that 45 different SNA metrics were used in these studies. These set of metrics address different properties of these networks. We found SNA metrics classified as "*Global and Local Measures*" that explores concepts like *centrality*, *ego*, and *structural holes* of the networks.

Finally, we found papers that discussed about software quality and social structures. Nicollas *et al.* [5] proposed dimensions to capture aspects from social interaction related to *Discussion Contents*, *Communication Dynamics,* and *Social Structures*. *Discussion Contents* are related to communication aspects like number of source code found in a discussion, or number of links listed in a discussion. *Communication Dynamics* are metrics that capture aspects from messages exchanged between developers, like reply time and number of messages. These metrics were also computed from developer mailing-list by Ibrahim *et al*. [14].

4.1.1.1 Classification proposed

Based on the previous discussion (Section 4.1) we proposed a classification schema to social metrics with three different groups: communication, project, and commit-related. We reported metrics used in the 48 papers selected in our systematic review. These metrics were classified twenty-one metrics in communication group, eighteen metrics in project group, and twelve metrics in commit group. We also found forty-five social network analysis metrics on eight types of social networks

Figure 2 presents the proposed classification. We used a mind-map schema to summarize the classification. First, we propose three groups. Each group is related to different groups of social metrics that represents the different interactions between developers during the software development.

The first group is the "*Communication*." The main sources to compute this set of metrics are issue trackers (like Github, JIRA, Bugzilla, etc) and mailing lists. Three different categories were described and 21 metrics were grouped. The first category represents the *Communication Dynamics*. This dynamic is related to the discussion activity, such as *experience* and *role* of participants involved in a discussion or *temporal aspects* of the messages. The metrics reporter reputation (P[6,21,46,48]), amount of messages (P[17,21,39,48]) and amount of words (P[17,21,39,48]) were the most frequently used metrics. The second category grouped metrics related to *Discussion Contents*. This category represents the interaction of developers during the exchange of messages and got information about the content of each message. For example, the amount of source code snippets written in a discussion. Just two papers P[17,21] used metrics of this group. The third category, called *Communication Structure* denotes the relationship among the developers during the communication tasks. We found 4 different types of social networks. The Developer Communication Network (P[7,13,17,50]) was the most cited network. We identify 45 SNA metrics used to analysis different types of networks, for example, developer communication network or task-assigned networks.

The second group, called "*Commit-related*" is related to the cooperative work carried by developers around the change of files. The main source to compute this set of metrics is the software repository. We also split this group in 3 main categories, grouping 12 metrics. *Ownership* grouped all metrics related to the developer contribution on commit time. We found 6 different metrics to compute the ownership. The two most cited metrics were amount of distinct developers that committed a file/module (cited by 26 distinct papers) and major contributor (P[20, 22,27,28,34]).

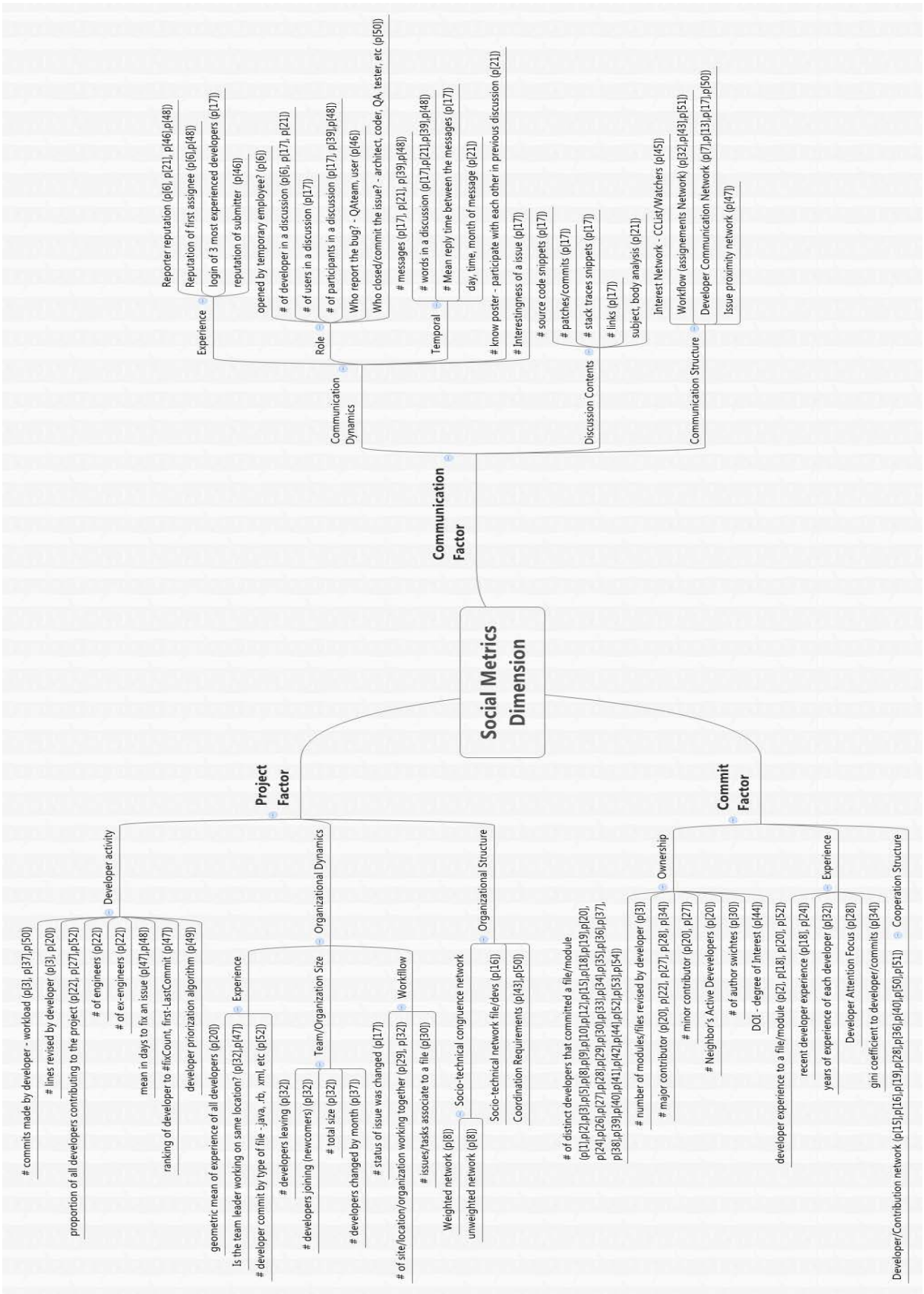

Figure 2. Mind Map with Social Metrics identified from distinct prediction models

The second category is related to the *Developer Experience*. We found 8 papers. They used 5 metrics to compute experience of each developer. The third category, called *Cooperation Structure* denotes the relationship among the developers when they commit the same file/module. The Developer/Contribution network was cited by 8 papers (P[15,16,19,28,36,40,50,51]).

The last group was called "*Project*." We grouped 18 metrics in 3 categories. The first category is related to the *developer activity*. The proportion of all developers contributing to the project (P[22,27,52]) was the most cited metric. The second category is *Organization Dynamic*. We grouped metrics related to the *experience*, *size of team/organization,* and metrics relate to the *workflow* of the project. We found 10 metrics used by 8 papers. The last category represents the *Organizational structure*, in which three different networks were identified: socio-technical congruence, socio-technical network between file and developers, and coordination requirements.

4.2 Implications to prediction models on software Engineering

To show the implications of social metrics to build prediction models, we discussed whether social metrics have positive, negative or neutral effect when they were considered as predictors.

4.2.1 RQ2: Did the social metrics have positive effect when they were considered as predictors?

During the quality criteria analysis (Table 2, question 7), we summarized which papers compared social metrics against other predictors or dimensions.

Table 4 shows the summary of social contribution to prediction models by the papers perspective. The column *effect* represents our interpretation for each result discussed by the original authors. We used the symbol "+" when the conclusion about the effectiveness of social metric was positive. The positive effect means that the results were stated satisfactory considering their evaluation approach. For example, one social metric has high accuracy or it was selected by one feature selection algorithm. The symbol "-" indicates the negative effect and "+ -" the neutral effect. The column *indications* report the summary of results presented by original author. If we are not able to identify the effect of the set of metrics used, the paper did not appeared on Table 4.

Table 4. Summary of social contribution

Paper	*Effect*	*Indications*
P[2]	+	**developer experience** was most important indicator of risky changes
P[3]	+	**Social metrics** improved the prediction performance when combined with other dimension. Strong correlation between fauls and developers
P[7]	+	**SNA metrics from communication network** reduced the cost required to verify the prediction results
P[8]	+	There is evidence that involving **many authors** in the same build also reduces the build success when using a weighted congruence conceptualization.
P[9]	+	The most significant predictors to predict failures are **number of developer's** and number of commits.
P[10]	+	**Number of active developers** gains significance when the prediction granularity was changed of file to package.
P[13]	+	**Developer communication** plays an important role in the quality of software integrations.
P[17]	-	The **contents of discussions** were considerably more relevant than **roles of participants**. Social information cannot explain a similar amount of variance than traditional models but complements the defect prediction models increasing their explanatory power.
P[18]	+	**Developer expertise**, expressed as the number of deltas a developer has made to the code, is a strong predictor of change quality
P[19]	+	Files are likely to be vulnerable when changed by **many developers** who have made many changes to other files.
P[21]	+	The **content**, the **length of a thread** and the **contribution activity** of a developer are the most important contribution factors.
P[24]	-	Slighter difference when **developer information** were add in prediction models.
P[27]	+	High levels of **ownership** are associated with less defects.
P[28]	+	When **developer attention focus** is higher, fewer defects were introduced
P[29]	-	The **number of developers** who had changed a module did not help predicting numbers of faults
P[33]	-	The **developer information** only slight improve the prediction results.
P[34]	+	Repository metrics (**ownership**) were able to get lower *pf rate* on the average from 32% to 23%. compared to static code attributes
P[36]	+ -	Code churn, commit changes, and **developer activity** can potentially reduce the vulnerability inspection effort compared to a random selection of files
P[37]	+	Found that bug fixing by **authors** who were active in the learning period helps to improve defect prediction quality.
P[39]	-	In conjunction with source code and change metrics, popularity metrics increase both the explanative and predictive power of existing defect prediction techniques.
P[40]	+ -	**Social network metrics** alone do not have significant impact on the prediction performance of the model. Although social metrics give us the lowest false alarm (pf) rates, the detection rates are also very low
P[41]	-	The **developer information** cannot improve significantly the prediction results
P[42]	-	The **developer information** cannot improve the prediction results.
P[43]	+	**Coordination requirements** were more likely to have defects in the files they worked on.
P[46]	+	**Submitter** and **ownership** are the top 2 most important features
P[48]	+	**Number of stakeholders** was good predictors of issue lead time
P[50]	+	Significant aspects of the relationship between **developer communication** and the units of work around which such communication takes place in a large-scale software project.
P[51]	+	Networks built using data from **Bug Re-assignment** and Co-Commit **developers' network** were relevant.
P[52]	+	**Organizational metrics** provide a better basis for predicting long-term code churn for individual files than code metrics
P[54]	+	The **number of distinct authors** that performed changes to a file show strong correlations with a file's defect count

We observed that 21 papers recommended the use of social metrics in prediction models against 6 papers that mentioned negative effect of social metrics. Two papers presented neutral results. Even when the results shown positive or negative effect, the conclusions were careful reported, indicating that more studies are necessary to explore the useful of social metrics in prediction models.

For example, we were not able to draw conclusions about the results of P[44]. However, authors noticed that future defect prediction models need to use more information from developers' and micro level interactions for effective defect prediction. Once this first analysis was focused on the papers, we provide on more fine-grained analysis focusing on our group of metrics proposed on Section 4.1.2 (Figure 2).

We want to highlight that the effects reported in Table 4 follow the authors indications. It is risky to draw conclusion on the positive, negative or neutral effect of social metrics from different papers, because each paper has its own context, machine learning method, experimental approach, performance measures, and so on, so they are not comparable.

4.2.2 Mapping prediction applicability, prediction techniques, social metrics and papers.

In this section, we mapped the groups of social metrics and prediction applicability. After, we performed the same analysis linking each prediction techniques for each group. Our aim was provide insights for help researchers to get an overview about the social metrics, prediction applicability and techniques used so far.

Table 5 presented the group of metrics and the applicability (purpose) for each study. The intersection between applicability and metrics indicated that at least one metric classified in our study was used as predictor. We noticed that the commit-related and project factors grouped the most part of the papers. The highest concentration of metrics were used to build models to predict fault/bug proneness and few groups were used to predict other applicability's. Considering the groups of metrics, *ownership*, *communication structure*, *developer activity* and *cooperation structure* were most selected by the studies. We also want to highlight that previous studies reported *centrality measures* computed using social network techniques for *communication* and *developer networks* as recurrent predictors.

Table 6 focused to present techniques to build prediction models. We observed that *ownership* metrics were most selected as predictor to classification and regression models. Considering the techniques, Naïve Bayes, J48, Decision Tree (classification models) and Logistic Regression (statistical model) were recurrently selected by researchers as techniques to build prediction models.

The mapping showed that the most groups of social metrics were used on spread way. They were considered in some specific applicability's.

5. Threats of Validity

Even though we have conducted snowballing process to select primary studies, we are aware that the selection of papers may not have captured all relevant studies during the first step (seed list). To treat this issue, we used two validation steps described in Section 4.3 and this validation was conduct by an author that did not participate in the first four steps (seed, backward, forward, and DBLP). We highlight that we received answers of 10 specialists and all suggested papers were found in some stage of our review method.

Another potential threat in snowballing is that we might find several papers from the same authors since their previous research is usually relevant and was cited. Thus, the results of snowballing approach might be biased by over presenting specific authors' research. To minimize this threat, for each of the 103 distinct authors identified during the seed, backward, and forward analysis, we visited the DBLP author's page, searching for new papers that could be included in our SLR. We found 3 additional papers during this step.

During the steps of selection and synthesis, some bias can be inserted. To reduce this issue, the selection process was conducted by two authors and constantly crosschecked. These two authors also jointly performed the synthesis. Conflicts were discussed until consensus was reached.

During the steps of selection and synthesis, some bias can be inserted. To reduce this issue, the selection process was conducted by two authors and constantly crosschecked. These two authors also jointly performed the synthesis. Conflicts were discussed until consensus was reached

The mind map reflects our interpretation about the papers and social metrics. As mentioned, we classified as social metrics, but many metrics were classified by other authors as process metrics. We encourage other researches refine our classification.

There are important factors influencing the use of social metrics, for example, whether open source or commercial software projects are studied, whether social metrics are combined with other prediction metrics. So, the effects of social metrics reported by each study are related to their context.

6. CONCLUSION

We conducted a systematic mapping to identify the use of social metrics in prediction models on the Software Engineering context. We found 48 papers that applied at least one metric that could be classified as social. We proposed a classification with 3 groups (communication, project, and commit-related) and nine categories. We grouped 21 metrics at communication, 18 metrics at project, and 12 metrics at commit group. We found 45 SNA metrics (centrality, global, ego, and structural hole measures) used to analyze social interactions.

Considering the prediction models built, the most frequent techniques were Naïve Bayes and Logistic Regression. More than one metric were used to evaluate the prediction results. Defect predictions on file level were the most common granularity and application. Only 14 studies considered more than 3 projects on their dataset´s. Menzies [21] recommended that it is necessary to discover and describe more information about the data collected from software development. He also recommend the use of other types of analysis to explore the data and its mining rules.

We observed that even when social metrics were considered, they were classified as part of other dimension, such as process, history, or change. Moreover, few papers reported the individual performance of social metrics as predictors.

Considering the results published so far, it could be risky to draw generalized conclusions about social metrics, since the studies employed many different techniques and investigated a limited number of software projects in different contexts. More studies are needed on this area, exploring the social metrics mapped on this study or proposing their own social metrics.

These studies should consider large scale and longitudinal analysis to investigate the effectiveness of social metrics to build prediction models. Since many previous works explored defect prediction on file level, to predict fault proneness, we encourage researchers to build predictive models to other related problems in software engineering, like tasks triage.

Finally, we claim that this work may support researchers and practitioners to build their prediction models, considering more social metrics, as well as the investigation of new hypothesis of influences of social aspects in software engineering activities.

7. ACKNOWLEDGMENTS

We thank Fundação Araucária, NAWEB and NAPSOL for the financial support. Marco G. receives individual grant from CNPq and FAPESP. Igor W. and Igor S. receive grants from CAPES (Process BEX 2039-13-3 and Process BEX 2038-13-7).

Table 5. Applicability of prediction models using Social Metrics

Social Metric Group	Social Metric Sub-Group	Applicability										
		Fault/Bug	Risky	Build Success	Vulnerability	Discussion Recomm.	Bug Triage	Bug Fixing Time	Re-opened Issues	Effort	Bug Severity	Churn File
Communication Dynamics (*)	Experience	P6,P17				P21		P46, P48				
	Role	P6,P17,P39 ,P50				P21		P46, P48				
	Temporal	P17, P39				P21		P48				
Discussion Contents (*)	-	P17				P21						
Communication Structure (*)	-	P7, P17, P32, P43, P47, P50,P51		P13			P45			P51	P51	
Ownership (**)	-	P3, P20, P22, P27, P28, P30, P34, P44										
Experience (**)	-	P20, P24, P28, P32, P34	P2, P18									P52
Cooperation Structure (**)	-	P15,P16,P28, P40, P50, P51			P19, P36					P51	P51	
Developer Activity (***)	-	P3, P20, P22, P27, P37, P50					P49	P48	P47			P52
Organizational Dynamics (***)	Experience	P20, P32							P47			P52
	Team/Organization Site	P32, P37										
	Workflow	P17, P29, P30, P32										
Organizational Structure (***)	-	P16,P43,P50		P8								

(*) communication factor, (**) commit factor, and (***) project factor

Ownership – # distinct developers that committed the file/module was cited 32 times (fault, risky, build, vulnerability)

Table 6. Prediction techniques using Social Metrics

Social Metric Group	Prediction technique														
	Classification									Statistical					
	NayB	J48	DT	BayN	SVM	RanF	RBFB	NN	KNN	LogR	LinR	BinR	MLR	PoiR	RTree	NLDT
Communication Dynamics (*)	P21, P46	P48	P21, P46	P46, P48	-	-	P46, P48	-	P46	P6, P17, P48	P48	-	P39, P48, P50	-	-	-
Discussion Contents(*)	P21	-	P21	-	-	-	-	-	-	P17	-	-	-	-	-	-
Communication Structure (*)	P7,P13	-	-	-	P45	-	-	-	-	P32, P47	-	-	P50	P15	-	-
Ownership (**)	P1,P20 P26, P34, P36, P40	P1,P2, P36, P37	P3,P21 ,P26,P 30,P44 ,P46, P52	P12,P1 9,P36	P20	P36	-	P52	-	P1, P2,P8 P9,P10, P15,P16, P18,P20, P22,P26, P32,P36, P38	P3, P30	P5,P15 ,P24, P28, P33, P41, P42	P27, P39, P50	P15, P29	P30	P35
Experience (**)	P20	P2	P52	-	P20	-	-	P52	-	P2 ,P20, P32	-	P24, P28	-	-	-	-
Cooperation Structure (**)	P36	-	-	P36	-	p36	-	-	-	P15, P16,P36	-	P15, P28	P50	P15	-	-
Developer Activity (***)	P20 ,P49	P37, P48	P3	P48	P20, P49	-	P48	P52	-	P3, P22,P47, P48	P3, P48	-	P27, P48, P50	-	-	-
Organizational Dynamics (***)	-	-	P30	-	P20	-	-	P52	-	P17,P20, P32,P47	P30	-	-	P29	P30	-
Organizational Structure (***)	-	-	-	-	-	-	-	-	-	P8,P16, P43	-	-	-	-	-	-

Naive Bayes (NayB), Decision Tree (DT), Bayes Network (BayN), Random Forest (RanF), RBF Bayes (RBFB), Neural Network (NN), Logistical Regression (LogR), Linear Regression (LinR), Binomial Regression (BinR), Multiple Linear Regression (MLR), Poison Regression (PoiR), Regression Tree (RTree), Non-linear Decision Tree (NLDT)

8. REFERENCES

[1] Arisholm, E., Briand, L.C. and Johannessen, E.B. 2010. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *Journal of Systems and Software*. 83, 1 (2010), 2–17.

[2] Azhar, D., Mendes, E. and Riddle, P. 2012. A systematic review of web resource estimation. *PROMISE* (2012), 49–58.

[3] Bacchelli, A., D'Ambros, M. and Lanza, M. 2010. Are Popular Classes More Defect Prone? *Proceedings of the 13th FASE* (2010), 59–73.

[4] Barbara Kitchenham, S.C. 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering Technical Report EBSE-2007-01. (2007).

[5] Bettenburg, N. and Hassan, A.E. 2013. Studying the impact of social interactions on software quality. *Empirical Softw. Engg.* 18, 2 (Apr. 2013), 375–431.

[6] Biçer, S., Bener, A.B. and Çaglayan, B. 2011. Defect prediction using social network analysis on issue repositories. *Proceedings of ICSSP 2011* (2011), 63–71.

[7] Bird, C., Nagappan, N., Gall, H., Murphy, B. and Devanbu, P. 2009. Putting It All Together: Using Socio-technical Networks to Predict Failures. *Proceedings of the 2009 20th International Symposium on Software Reliability Engineering* (2009), 109–119.

[8] Catal, C. 2011. Software fault prediction: A literature review and current trends. *Expert Systems with Applications*. 38, 4 (2011), 4626–4636.

[9] Catal, C. and Diri, B. 2009. A systematic review of software fault prediction studies. *Expert Systems with Applications*. 36, 4 (2009), 7346–7354.

[10] D'Ambros, M., Lanza, M. and Robbes, R. 2012. Evaluating defect prediction approaches: a benchmark and an extensive comparison. *Empirical Softw. Eng.* 17, 4-5 (Aug. 2012), 531–577.

[11] Dybå, T. and Dingsøyr, T. 2008. Strength of Evidence in Systematic Reviews in Software Engineering. *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (2008), 178–187.

[12] Graves, T.L., Karr, A.F., Marron, J.S. and Siy, H. 2000. Predicting fault incidence using software change history. *Software Engineering, IEEE Transactions on*. 26, 7 (Jul. 2000), 653–661.

[13] Hall, T., Beecham, S., Bowes, D., Gray, D. and Counsell, S. 2012. A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Trans. Software Eng.* 38, 6 (2012), 1276–1304.

[14] Ibrahim, W.M., Bettenburg, N., Shihab, E., Adams, B. and Hassan, A.E. 2010. Should I contribute to this discussion? *MSR '10: Proceedings* (2010), 181–191.

[15] Illes-Seifert, T. and Paech, B. 2010. Exploring the relationship of a file's history and its fault-proneness: An empirical method and its application to open source programs. *Information and Software Technology*. 52, 5 (2010), 539–558.

[16] Jalali, S. and Wohlin, C. 2012. Systematic Literature Studies: Database Searches vs. Backward Snowballing. *Proceedings of the ACM-IEEE ESEM* (2012), 29–38.

[17] Jorgensen, M. and Shepperd, M. 2007. A Systematic Review of Software Development Cost Estimation Studies. *Software Engineering, IEEE Transactions on*. 33, 1 (Jan. 2007), 33–53.

[18] Jureczko, M. and Madeyski, L. 2011. A review of process metrics in defect prediction studies. *Metody Informatyki Stosowanej*. 5 (2011), 133–145.

[19] Kitchenham, B. and Brereton, P. 2013. A systematic review of systematic review process research in software engineering. *Information and Software Technology*. 55, 12 (2013), 2049–2075.

[20] Meneely, A., Williams, L., Snipes, W. and Osborne, J. 2008. Predicting Failures with Developer Networks and Social Network Analysis. *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (2008), 13–23.

[21] Menzies, T. 2012. Predicting the Future of Predictive Modeling. *NSF Workshop: Planning Future Directions in AI &SE (AISE'12)* (Sep. 2012).

[22] Mockus, A. 2010. Organizational Volatility and Its Effects on Software Defects. *Proceedings of the Eighteenth ACM SIGSOFT FSE* (2010), 117–126.

[23] Moser, R., Pedrycz, W. and Succi, G. 2008. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. *Proceedings of the 30th ICSE* (2008), 181–190.

[24] Nagappan, N., Murphy, B. and Basili, V.R. 2008. The influence of organizational structure on software quality: an empirical case study. *ICSE* (2008), 521–530.

[25] Nagappan, N., Zeller, A., Zimmermann, T., Herzig, K. and Murphy, B. 2010. Change Bursts as Defect Predictors. *Proceedings of the ISSRE 2010* (2010), 309–318.

[26] Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M. 2008. Systematic Mapping Studies in Software Engineering. *Proceedings of the 12th EASE* (2008), 68–77.

[27] Pinzger, M., Nagappan, N. and Murphy, B. 2008. Can Developer-module Networks Predict Failures? *Proceedings of the 16th ACM SIGSOFT FSE* (2008), 2–12.

[28] Radjenović, D., Heričko, M., Torkar, R. and Živkovič, A. 2013. Software fault prediction metrics: A systematic literature review. *Information and Software Technology*. 55, 8 (2013), 1397–1418.

[29] Rahman, F. and Devanbu, P. 2013. How, and why, process metrics are better. *Proceedings of the 2013 International Conference on Software Engineering* (2013), 432–441.

[30] Ratzinger, J., Pinzger, M. and Gall, H. 2007. EQ-mine: Predicting Short-term Defects for Software Evolution. *Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering* (2007), 12–26.

[31] Riaz, M., Mendes, E. and Tempero, E. 2009. A systematic review of software maintainability prediction and metrics. *Empirical Software Engineering and Measurement* (Oct. 2009), 367–377.

[32] Shihab, E., Hassan, A.E., Adams, B. and Jiang, Z.M. 2012. An industrial study on the risk of software changes. *Proceedings of the ACM SIGSOFT 20th FSE* (2012), 62:1–62:11.

[33] Souza, C.R. de, Quirk, S., Trainer, E. and Redmiles, D.F. 2007. Supporting Collaborative Software Development Through the Visualization of Socio-technical Dependencies. *Proceedings of the 2007 GROUP* (2007), 147–156.

[34] Webster, J. and Watson, R.T. 2002. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Q.* 26, 2 (Jun. 2002), xiii–xxiii.

[35] Weyuker, E.J., Ostrand, T.J. and Bell, R.M. 2008. Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models. *Empirical Software Engineering*. 13, 5 (2008), 539–559.

[36] Wolf, T., Schroter, A., Damian, D. and Nguyen, T. 2009. Predicting build failures using social network analysis on developer communication. *Software Engineering, 2009. ICSE 2009.* (May. 2009), 1–11.