HAL open science

A Framework for Defect Prediction in Specific Software Project Contexts

Dindin Wahyudin, Rudolf Ramler, Stefan Biffl

▶ To cite this version:

Dindin Wahyudin, Rudolf Ramler, Stefan Biffl. A Framework for Defect Prediction in Specific Software Project Contexts. 3rd Central and East European Conference on Software Engineering Techniques (CEESET), Oct 2008, Brno, Czech Republic. pp.261-274, 10.1007/978-3-642-22386-0_20 . hal-01572547

HAL Id: hal-01572547

https://inria.hal.science/hal-01572547v1

Submitted on 7 Aug 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Framework for Defect Prediction in Specific Software Project Contexts

Dindin Wahyudin[1], Rudolf Ramler[2], and Stefan Biffl[1]

[1] Institute for Software Technology and Interactive Systems,
Vienna University of Technology,
Favoritenstr. 9/188, A-1040 Vienna, Austria

{dindin, stefan.biffl}@ifs.tuwien.ac.at

[2] Software Competence Center Hagenberg
Softwarepark 21, A-4232 Hagenberg, Austria

rudolf.ramler@scch.at

Abstract. Software defect prediction has drawn the attention of many researchers in empirical software engineering and software maintenance due to its importance in providing quality estimates and to identify the needs for improvement from project management perspective. However, most defect prediction studies seem valid primarily in a particular context and little concern is given on how to find out which prediction model is well suited for a given project context. In this paper we present a framework for conducting software defect prediction as aid for the project manager in the context of a particular project or organization. The framework has been aligned with practitioners' requirements and is supported by our findings from a systematical literature review on software defect prediction. We provide a guide to the body of existing studies on defect prediction by mapping the results of the systematic literature review to the framework.

Keywords: Software Defect Prediction, Systematical Literature Review, Metric-based Defect Prediction

1. Introduction

Software defect prediction has caught considerable attention from researchers as well as practitioners due to the increasing importance of software products as backbone for reliable industry systems. The rationale for identifying defective components of a software system prior to applying analytical quality assurance (QA) measures like inspection or testing has been summarized by Nagappan et al.: "During software production, software quality assurance consumes a considerable effort. To raise the effectiveness and efficiency of this effort, it is wise to direct it to those which need it most. We therefore need to identify those pieces of software which are the most likely to fail – and therefore require most of our attention." [17]A wide range of studies provide evidence about successful prediction of defects and various scenarios on how to

exploit defect prediction have been proposed, for example, focusing testing and QA activities, making informed release decisions, mitigating risks, allocating resources in maintenance planning, and supporting process improvement efforts.

These studies also provide valuable advice and share lessons learned important for those who want to adopt defect prediction in practice. Currently there are many approaches to perform defect prediction [9] and respective validation methods [4, 20]. However, Koru et al. [8] advise that in practice, the most appropriate prediction method has to be selected for the current project context and the type of defect pattern to be predicted. Thereby, a good defect prediction model has to be constructed using a set of predictor variables that represents the actual measures of the software product and process [14, 15, 26]. Furthermore, several measures to evaluate the quality of a prediction are recommended, e.g. [13], and calibrating the prediction model to align false alarm rates with prediction goals and business scenarios is recommended [12].

Despite the many findings and the comprehensive information provided by the existing studies, there still is a wide gap between published research results and their adoption in real-world projects. Studies sharing insights about the application of defect prediction in practice are rare. Li et al. [10] discuss experiences and results from initiating defect prediction at ABB Inc. for product test prioritization and maintenance resource planning. Ostrand et al. [21] describe automating algorithms for the identification of fault-prone files to support the application of defect prediction in a wide range of projects. These studies show that in many cases, research results on defect prediction cannot directly be translated to practice. Adaptation and interpretation in the context of a particular project or organization is required. Furthermore, many studies focus on specific research questions. While these studies provide a valuable contribution to defect prediction research, this contribution remains an isolated piece of a bigger picture without following the entire track of research.

The objective of this paper is to provide a guide to the body of existing studies on defect prediction to facilitate the use of systematic defect prediction in the context of a particular project or organization. Thus, common requirements for defect prediction in practice are outlined in Section 2, from these requirements a generic framework for conducting defect prediction in practice is derived in Section 3. From 12 published studies on defect prediction, findings have been distilled in a systematic literature review described in Section 4. The results are presented within the structure of the proposed framework in Section 5. Section 6 summarizes the paper and discusses questions for future work.

2. Requirements for Defect Prediction a Software Project

A number of empirical studies provide evidence of successful prediction of defects using data from real-world projects conducted in an industrial or open-source context. However, practitioners are confronted with additional requirements when they try to replicate the success of these studies within the context of their specific projects and organizations. We compiled a list of typical requirements encountered when conducting defect prediction in practice from both, the existing body of literature and our own experience on predicting defects, e.g.,[26].

- **Aligning defect prediction with project and business goals.** Empirical studies tend to focus on prevalent research questions. Practitioners, however, have to align defect prediction with the goals of their specific project. Concentrating

testing on defect-prone components or planning the effort for maintenance activities are examples for such goals. Defining the goals first is therefore an important requirement as an appropriate budget has to be allocated for defect prediction and, moreover, the investment has to be justified according to estimated savings and benefits.

- **Creating a project-specific prediction model.** Prediction models are constructed from a project's historical data. A prediction model, thus, models the context of a particular project. As a consequence, predictors obtained from one project are usually not applicable to other projects. Nagappan et al. [18], for example, showed that predictors are accurate only when obtained from the same or similar projects and that there is no single set of metrics that is applicable to all projects. These findings were supported by Koru and Liu [8] when analyzing the PROMISE repository containing data about projects conducted at different sites. "Normally, defect prediction models will change from one development environment to another according to specific defect patterns." [8]
- **Evaluating the feasibility in the project or organizational context.** Despite the success reported by many studies, the prediction of defects in a particular project may not be possible. Typical reasons are the poor quality of the available data [6] or the effort required to extract and collect the necessary data [23]. Most published studies report solely successful cases of defect prediction. Only few studies point toward limitations, for example, Li et al. [10] comment on the poor accuracy in predicting field defects for one of the studied products. The feasibility of predicting defects has to be estimated early to confirm that the defined goals will be met.
- **Striving for fast results.** Even when the feasibility is positively evaluated, defect prediction is required to produce results fast. Defect prediction is relatively new in the software development arena, and practitioners face a high level of uncertainty concerning the return on the investment in defect prediction. Thus, when results cannot be obtained within one or a few iterations the chance defect prediction will be applied in a real-world project is low. The general concerns of practitioners have also been described by Ostrand et al. [21]: "In our experience, practitioners won't even consider using a new technology without evidence that it has worked on a substantial number of real systems of varying types. It is very unlikely that practitioners will be convinced that a new tool is worth learning and evaluating merely on the basis of its demonstration on toy systems or on systems much smaller than the ones they normally develop and maintain."
- **Dealing with incomplete, insufficient data.** Extraction and integration of data from corporate databases and repositories is a costly, time-consuming endeavor and, eventually, does not assure the data is of appropriate quality. Thus, Li et al. [10] observe that "dealing with missing/incomplete information is important to practitioners because information is often not available in real-world settings and conducting analysis without important categories of predictors (e.g. deployment and usage predictors for field defect predictions) jeopardizes the validity and accuracy of results".
- **Predicting under uncertainty.** Fenton and Neil [2] remind that "Project managers make decisions about software quality using best guesses; it seems to us that will always be the case and the best that researchers can do is 1) recognize this fact and 2) improve the 'guessing' process. We, therefore, need to model the subjectivity and uncertainty that is pervasive in software development."

Uncertainty exists besides limitations resulting from incomplete, insufficient data. It arises often about how the data has to be interpreted, which reflects the peculiarities of a project such as individual project regulations, discontinuities in workflows and processes or specific use of tools. Practitioners therefore rely on expert judgment and have to make assumptions. These assumptions should be made explicit and – as a positive side-effect – the prediction model should provide information to verify these assumptions.

- **Outsourcing of model creation.** Ostrand et al. [21] found that "it is very time consuming to do the required data extraction and analysis needed to build the models, and few projects have the luxury of extra personnel to do these tasks or the extra time in their schedules that will be needed. In addition, statistical expertise was needed to actually build the models, and that is rare to find on most development projects". As a consequence, it should be possible to organize data extraction and model creation separately so it can be outsourced or – if tool support permits – automated.
- **Reusing and validating the existing model for upcoming releases.** To optimize the return on the investment in model creation, the model has to be reused for upcoming releases with minimal additional effort. However, over time, the project's context and the defect patterns can change. As a consequence, prediction results for a new release derived from a model created and verified with historical data have to be validated. Practitioners need a measure of reliability when they make decisions based on prediction results. Furthermore, Koru and Liu [8] point out that "as new measurement and defect data become available, you can include them in the data sets and rebuild the prediction model." As adjusting or rebuilding the model requires additional effort, the validation results should serve as an indicator when adjusting or rebuilding becomes necessary.

3. Software Defect Prediction Framework

In this section we described a framework for software defect prediction which consists of three phases – (A) preparation, (B) model creation and (C) model usage – as well as seven steps (see Figure 1.) This framework is in line with the requirements outlined in the previous section and has been derived from our experience and literature on software defect prediction and software estimation.

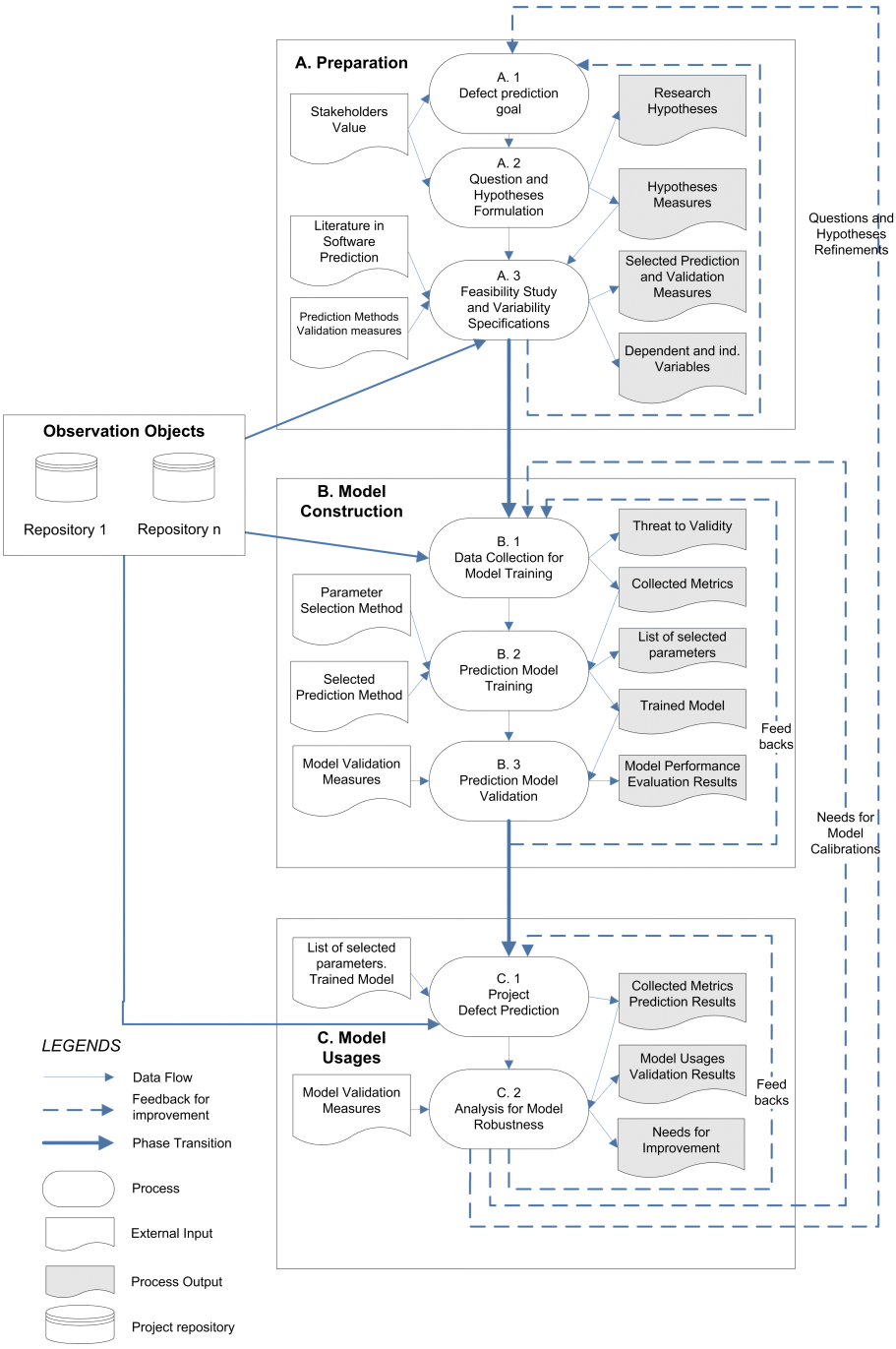

Figure 1: Framework for Software Defect Prediction

3.1. Phase A – Preparation

As first phase in conducting a defect prediction, one should start by preparing the necessary preconditions prior to model construction. The intention of the preparation phase is to create a clear focus of what results should be provided by the prediction, to appropriately design the prediction approach, and to have quick analysis whether such design will accomplish the expected results within project and organizational context. Following the Goal Question Metrics (GQM) model proposed by Basili et al. [1], we structure the first phase with following steps:

A.1 **Define defect prediction goal,** which represents the objective of defect prediction with respect to a particular stakeholder perspective and the current project context.

A.2 **Specify questions and hypotheses**. Questions are derived from the defect prediction goals. They are used to identify relevant models of the objects of study and, then, to more precisely define the expected achievement of a specific goal. The questions can be reframed as hypotheses about the observed situation or defect pattern. We recommend specifying hypotheses that are easily measurable to enable the falsification or acceptance of the hypotheses for a sound assessment of the prediction results.

A.3 **Quick feasibility study and variables specification.** A quick feasibility study is essential to assess whether the initial goals of the prediction can be achieved using the available data from the observation objects. A negative assessment indicates the initial goals are not feasible and shows the need for adjusting the goals and questions. After conducting a feasibility study, the set of metrics that should be collected and estimated in the prediction model is collected. These metrics act as independent variables and dependent variables in the prediction model to be constructed in the next phase.

3.2. Phase B – Model Construction

Constructing the prediction model is the core phase in defect prediction, Here, based on the variables and the defect prediction method specified in the previous phase, data collection, model training, and model evaluation are performed.

B.1 **Data collection for model training**. As part of a close investigation of the available data sources, the period of observation and relevant project repositories and databases are specified. Based on the previously selected variables the data is collected from the observation objects. Invalid and missing data is thereby filtered or refined. For making a sound prediction, potential threats to validity are recorded.

B.2 **Prediction model training**. Parameter selection is used to identify the parameters with a significant impact on the dependent variables. These parameters are used in training the model, usually applying standard statistical or machine learning tools.

B.3 **Prediction model validation.** The trained model needs to be validated for its performance, i.e., accuracy, recall and precision. Unsatisfying results should trigger a feedback loop back to the step data collection, as it will not make sense to proceed with a low-performance model that, e.g., has a high number of false positives or errors.

3.3. Phase C – Model Usages

A major concern from a practitioner's point of view is that many studies reported a trained defect prediction model which show a good performance by means of cross validation with historical data [2]. Only limited studies reported the robustness of the model with different observations. This, however, is a necessity in practical usages for predicting the quality for a certain time period in the future.

C.1 **Project defect prediction.** In this step the model trained in the previous phase is actually used, i.e. the model is parameterized with observations from new releases to predict defects in these releases.

C.2 **Analysis for prediction model robustness**. Based on the results of step C.1, the robustness of the model is analyzed. Thereby, the reliability of the current prediction results are estimated to determine how to apply the prediction results in the project, e.g., to safely rely on them or to be careful. If the analysis indicates low reliability, a feedback loop back to re-creating or calibrating the model should be triggered as well as suggestions for refinement of the prediction hypotheses should be provided.

4. Review of the Body of Literature on Defect Prediction

Numerous empirical studies on software defect prediction have been published in journals and conference proceedings. In order to provide a systematic guide to the existing body of literature, relevant studies have been searched and selected following the approach for a systematic literature review proposed by Kitchenham et al. [5]. By following this approach we identified 12 studies on defect prediction providing findings applicable within the framework outlined above.

A systematic literature review is defined as "a form of secondary study that uses a well-defined methodology to identify, analyze and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable" [5]. Staples and Niazi [24] summarize the characteristics of a systematic literature review: (a) a systematic review protocol defined in advance of conducting the review, (b) a documented search strategy, (c) explicit inclusion and exclusion criteria to select relevant studies from the search results, (d) quality assessment mechanisms to evaluate each study, (e) review and cross-checking processes to control researcher bias.

A key element of a systematic literature review is the review protocol, which documents all other elements constituting the systematic literature review. They in-

clude the research questions, the search process, the inclusions and exclusion criteria, and the quality assessment mechanisms.

- **Research Questions.** The research questions summarize the questions frequently addressed in empirical studies. These questions contribute essential findings from research to the application of defect prediction in practice and are mapped to the phases of the framework. According to the framework, we emphasize three research questions to guide the systematical literature review process:

  RQ1. *How do successful studies in defect prediction design the prediction process prior to model construction?*

  RQ2. *How do successful studies in defect prediction construct the prediction model from collected data?*

  RQ3. *How can external validation of the prediction model be provided for future predictions?*

- **Search Process.** The search process describes the process to identify the list of candidate studies. Following search process advocated by Barbara Kitchenham et al. [7], the search process was organized into two separate phases. The initial search phase identified candidate primary studies based on searches of electronic digital libraries from IEEE, ACM, Elsevier, Springer, and Wiley. Search strings have been composed from search terms such as defect, error, fault, bug, prediction, and estimation. The secondary search phase is to review the references in each of the primary studies identified in the first phase looking for more candidate primary sources which repeated until no further relevant papers can be found.
- **Inclusion and Exclusion Criteria.** The criteria for including a primary study comprised any study that compared software defect predictions which enables metric-based approaches based on analysis of project data. We excluded studies where data collected from a small number of observations (less than 5 observations). We also excluded studies where models constructed only based on historical data of defects with no other metrics as predictor variables. The third exclusion criterion is that we only consider studies that performed internal validation and external validation of constructed prediction model.

  Formal inclusion criteria are that papers have to be peer reviewed and document empirical research. Regarding the contents, inclusion requires that the study addresses at least one of the defined research questions.
- **Quality Assessment Mechanism.** This systematic literature review has been based on a documented and reviewed protocol established in advance of the review. Furthermore, in this study two researchers were involved in conducting the systematic literature review and cross validation of the results. For example, one researcher queried a digital library and extracted candidate studies while the second researcher verified the search terms, search results, and the list of identified candidate studies. Thereby we minimized researcher bias and assured the validity of the findings of the review.

5. Extraction of Findings and Discussion

This section maps the findings from the systematic literature review to the phases and tasks of the framework for defect prediction. The findings summarize the contributions extracted from the studies with respect to the research questions 1 to 3 used to drive our systematic literature review.

Table 1 lists information about how current research defines the goals of defect prediction studies, questions and hypotheses, as well as how variables are specified to describe each question.

Table 1 Study Related Factors- Preparation Phase

Study	**Preparation Steps**		
	A.1	A.2	A.3
	Goal definition	Research questions	Variables Specification
Moser et al [14]	Goal is implicitly described	Questions proposed with respective null hypotheses	Implicit variables specifications to predict module defect proneness
Li et al [9]	Goal is implicitly described	Explicit research question with no hypotheses	Explicit variables specification to predict defect intensity of a release
Zimmermann et al [28]	Goal is implicitly described	Implicit research question with no hypotheses	Implicit variables specifications to predict module defect proneness
Koru et al [8]	Implicit goal description	Implicit research question with no hypotheses	Implicit variables specifications to predict module defect proneness
Nagappan et al [16]	Implicit goal description	Explicit research hypotheses	Explicit variables specification
Li et al [11]	Goal is implicitly described	Explicit research question with no hypotheses	Explicit variables specification to predict defect intensity of a release
Weyuker et al [27]	Explicit goal description in later section	Implicit Research questions with hypotheses	Implicit variable specification to predict file defect proneness
Menzies et al [13]	Implict goal description	Implicit research question, hypotheses described later in the paper	Explicit variables specification for module defect proneness
Graves et al [3]	Goal is implicitly described	Implicit research questions with no hypotheses	Explicit variables specification for module defect proneness
Sunghun et al [25]	Implicit goal description	Explicit research hypotheses	Explicit variables specification of file defect proneness
Pai et al [22]	Implicit goal description	Explicit research question with no hypotheses	Explicit variables specification for number of defect per class and class defect proneness
Olague et al [19]	Ecxplicit goal statement	Explicit research hypotheses to describe proposed goal	Implicit variable specification to predict class defect proneness

Note that *Explicit* mean the study describes the following terms (goal, hypotheses, etc) clearly as a separate part from surrounding texts and adhere to our term defini-

tions in the framework. *Implicit* mean we need to extract the information from the text to identify a term definition. An *N/A* reveals that there is no information that contains the definition of an expected term in the study.

Most of the studies do not explicitly describe the goal of the study and there is no single study which identifies the target stakeholders of the results with their values expectations. 7 out of 12 studies explicitly stated the research questions and/or respective hypotheses, which provide guidance for the remaining empirical study process. Most of the studies specified the variables as part of the prediction model construction prior to data collection. Thus, we assert that the first phase in our framework which consists of goal definition, research questions and hypotheses formulation, and variable specifications is a common practice in conducting defect prediction with different levels of detail and presentation.

Table 2 outlines the collected data regarding common practices to construct prediction models. Most of these studies used variable selection prior to model construction. Methods such as Spearman bivariate correlation analysis and linear regression with selected methods (backward, stepwise, remove) are considered as common methods for variable selection.

The selection of prediction methods is based on what kind of defect pattern to be predicted, i.e., classification techniques such as logistic regression can be used to predict file defect-proneness but will obtain poor performance to predict file defect rates. Similar to prediction method selection, one should also choose appropriate internal validation methods and model performance measures.

Table 2 Study Related Factors- Model Construction

Study	**Model Construction Steps**			
	B.1	B.2	B.3	
	Variable Selection	Prediction Methods	Internal Validation	Model Performance Measures
Moser et al [14]	product and process metrics with no prior selection	Naive Bayes, Logistic regression and J48 with	10 Fold cross validation and Performance measure:	Number of False positive and Recall
Li et al [9]	Product and process metrics with prior variable selection	16 modeling methods	N/A	Average relative error
Zimmermann et al [28]	Product metrics with selection by Spearman bivariate correlation analysis	Naive Bayes, Logistic regression and J48	10 Fold cross validation.	Performance measures: Accuracy, recall and precision
Koru et al [8]	Product (Design) metrics with no prior selection	J48	10 Fold cross validation	Performance measures Recall, Precision and F-Measure
Nagappan et al [16]	Process (code churn) metrics with selection by Spearman correlation	Multiple regression, Step-wise regression and Principal Component Analysis (PCA)	Coefficient of determination analysis, F-test	Discriminant analysis
Li et al [11]	Product and process	16 modeling	N/A	Average relative

Li et al [11]	Product and process metrics with no prior selection	16 modeling methods	N/A	Average relative error
Weyuker et al [27]	Product and Process (developer) metrics	Negative binomial regression	N/A	Correctly identified files
Menzies et al [13]	Product (static code) metrics	Naive Bayes with log transform, J48, OneR	10 Fold cross validation	Accuracy, Number of false positive, Receiver operator curves
Graves et al [3]	Product (changes code) metrics	General linear models	N/A	Error measure
Sunghun et al [25]	Process (change) metrics with no variable selection	FixCache prediction method	Cross validation for all data set	Accuracy
Pai et al [22]	Product metrics with variable selection by correlation analysis and backward linear regression	Multiple linear regression, Logistic regression, Bayesian network model	10 Fold cross validation	False positive rate, precision , specificity, sensitivity
Olague et al [19]	Product (Object Oriented) Metrics with selection by Spearman bivariate	Univariate and Multivariate binary logistic regression	Hold out method	Percentage of correctly classified classes

Table 3 Study Related Factors- Model Usages

Study	**Model Usages Steps**	
	C.1	C.2
	External validation	Robustness Analysis
Moser et al [14]	Cross validation with different releases with low performance results	N/A
Li et al [9]	Constructed model were used to predict a certain period of defect growth per release	Proposed framework was used for commercial context [11]
Zimmermann et al [28]	Cross validation of trained prediction model in different releases and levels of observation	N/A
Koru et al [8]	Cross validation of trained prediction model with different class of data	Depicts the need for model calibration or refinement
Nagappan et al [16]	State briefly with no data	N/A
Li et al [11]	Cross validation with different releases	N/A
Weyuker [27]	N/A	N/A
Menzies [13]	N/A	N/A
Graves [3]	N/A	N/A
Sunghun[25]	N/A	N/A
Pai et al [22]	N/A	N/A
Olague et al [19]	N/A	N/A

The first two phases in our framework have been identified as commonly performed by researchers in defect prediction. However, for the third phase Model Usages (see Table 3), we found only two studies providing appropriate results of the two involved steps. This finding confirms the critique from Norman and Fenton [2] that most of the existing studies on defect prediction do not provide empirical proof whether the model can be generalized for different observations.

There are several reasons why many studies did not report the external validation and robustness analysis of constructed prediction model such as the availability of new observation data [22] and external validation results which signify poor performance of the model [14] for which many of the authors do not wish to report. However from practitioners' perspective such conditions should be addressed properly by data collection process refinement and model calibrations until the model can be proven for its usefulness for prediction in particular context.

6. Conclusion and Further Work

Whilst a large number of studies address defect prediction, little support is provided about the application of defect prediction for practitioners. In this paper we proposed a framework for conducting software defect prediction as an aid for the practitioner establishing defect prediction in the context of a particular project or organization and as a guide to the body of existing studies on defect prediction. The framework has been aligned with practitioners' requirements and supported by our findings from a systematical literature review on software defect prediction.

The systematic literature review also served as an initial empirical evaluation of the proposed framework by showing the co-existence of the key elements of the framework in existing research on software defect prediction. The mapping of findings from empirical studies to the phases and steps of the framework show that the existing literatures can be easily classified using the framework and verifies that each of the steps is attainable.

Nevertheless, we also found several issues relevant for applying defect prediction in practice, which are currently not adequately addressed by existing research. Related future work is encouraged in order to make software defect prediction a commonly accepted and valuable aid in practice:

- Existing studies on defect prediction neglect the fact that information is often missing or incomplete in real world settings. Practitioners therefore require methods to deal with missing or incomplete information. Li et al. reported: "We find that by acknowledging incomplete information and collecting data that capture similar ideas as the missing information, we are able to produce more accurate and valid models and motivate better data collection." [11]
- Defect predictions remain a risky endeavor for practitioners as long as upfront investments for data collection and model construction are high and a return on these investments has to be expected late or never. Most projects and organizations cannot afford this investment under such adverse conditions. Thus, means are required to conduct an early and quick estimation of the feasibility of predicting defects with acceptable performance in the context of a specific project or organization.
- If it is the vision that practitioners base critical decisions in software engineering such as what to test less on defect prediction results, they have to be sure

not to endanger product quality by missing critical defects. Thus, in addition to results of a prediction, an additional measure has to indicate the reliability of the prediction, so practitioners are informed to what extent they can rely on the results and know to what extent they are taking risks if they do so.

Acknowledgements. This paper has been partly supported by The Technology-Grant-South-East-Asia No. 1242/BAMO/2005 Financed by ASIA-Uninet.

References

1. Basili, V., Caldiera, G., Rombach, H.D.: The Goal Question Metric Approach. . 2 (1994) 6
2. Fenton, N., Neil, M.: A Critique of Software Defect Prediction Models. IEEE Trans. Softw. Eng. 25 (1999) 15
3. Graves, T.L., Karr, A.F., Marron, J.S., Siy, H.: Predicting fault incidence using software change history. IEEE Transactions on Software Engineering 26 (2000) 8
4. Khoshgoftaar, T., Bhattacharyya, B., Richardson, G.: Predicting Software Errors, During Development, Using Nonlinear Regression Models: A Comparative Study. IEEE Transaction On Reliability 41 (1992) 5
5. Kitchenham, B.: Guidelines for performing Systematic Literature Reviews in Software Engineering. (2007)
6. Kitchenham, B., Kutay, C., Jeffery, R., Connaughton, C.: Lessons learnt from the analysis of large-scale corporate databases. Proceedings of the 28th international conference on Software engineering. ACM, Shanghai, China (2006)
7. Kitchenham, B.A., Mendes, E., Travassos, G.H.: Cross versus Within-Company Cost Estimation Studies: A Systematic Review. Software Engineering, IEEE Transactions on 33 (2007) 316-329
8. Koru, A.G., Hongfang, L.: Building Defect Prediction Models in Practice. IEEE Softw. 22 (2005) 23-29
9. Li, P.L., Herbsleb, J., Shaw, M.: Forecasting Field Defect Rates Using a Combined Time-Based and Metrics-Based Approach: A Case Study of OpenBSD. the 16th IEEE International Symposium on Software Reliability Engineering. IEEE Computer Society (2005)
10. Li, P.L., Herbsleb, J., Shaw, M., Robinson, B.: Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB Inc. Proceedings of the 28th international conference on Software engineering. ACM, Shanghai, China (2006)
11. Li, P.L., Herbsleb, J., Shaw, M., Robinson, B.: Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB Inc. the 28th International Conference on Software engineering. ACM, Shanghai, China (2006)
12. Menzies, T., Di Stefano, J., Ammar, K., McGill, K., Callis, P., Chapman, R., Davis, J.: When Can We Test Less? : Proceedings of the 9th International Symposium on Software Metrics. IEEE Computer Society (2003)
13. Menzies, T., Greenwald, J., Frank, A.: Data Mining Static Code Attributes to Learn Defect Predictors. Software Engineering, IEEE Transactions on 33 (2007) 2-13
14. Moser, R., Pedrycz, W., Succi, G.: A Comparative Analysis of the Efficiency of Change Metrics and Static Code Attributes for Defect Prediction the 30th International Conference on Software Engineering. ACM, Leipzig, Germany (2008)
15. Nachtsheim, C.J., Kutner, M.H.: Applied Linear Regression Models. McGraw-Hill Education (2004)
16. Nagappan, N., Ball, T.: Use of relative code churn measures to predict system defect density. the 27th International Conference on Software Engineering. ACM, St. Louis, MO, USA (2005)
17. Nagappan, N., Ball, T., Zeller, A.: Mining metrics to predict component failures. the 28th International Conference on Software Engineering. ACM, Shanghai, China (2006)

18. Nagappan, N., Ball, T., Zeller, A.: Mining metrics to predict component failures. Proceedings of the 28th international conference on Software engineering. ACM (2006) 452-461
19. Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S.: Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. Software Engineering, IEEE Transactions on 33 (2007) 402-419
20. Ostrand, T.J., Weyuker, E.J.: How to measure success of fault prediction models. Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting. ACM, Dubrovnik, Croatia (2007)
21. Ostrand, T.J., Weyuker, E.J., Bell, R.M.: Automating algorithms for the identification of fault-prone files. Proceedings of the 2007 international symposium on Software testing and analysis. ACM, London, United Kingdom (2007)
22. Pai, G.J., Dugan, J.B.: Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods. Software Engineering, IEEE Transactions on 33 (2007) 675-686
23. Ramler, R., Wolfmaier, K.: Issues and Effort in Integrating Data from Heterogeneous Software Repositories and Corporate Databases. 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM'08), Kaiserslautern, Germany (forthcoming)
24. Staples, M., Niazi, M.: Experiences using systematic review guidelines. Journal of Systems and Software 80 (2007) 1425-1437
25. Sunghun, K., Thomas, Z., E. James Whitehead, J., Andreas, Z.: Predicting Faults from Cached History. Proceedings of the 29th International Conference on Software Engineering. IEEE Computer Society (2007)
26. Wahyudin, D., Winkler, D., Schatten, A., Tjoa, A.M., Biffl, S.: Defect Prediction using Combined Product and Project Metrics A Case Study from the Open Source "Apache" MyFaces Project Family. 34th EUROMICRO Conference on Software Engineering and Advanced Applications SPPI Track Parma Italy (2008)
27. Weyuker, E.J., Ostrand, T.J., Bell, R.M.: Using Developer Information as a Factor for Fault Prediction. the Third International Workshop on Predictor Models in Software Engineering. IEEE Computer Society (2007)
28. Zimmermann, T., Premraj, R., Zeller, A.: Predicting Defects for Eclipse. International Workshop on Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007 (2007) 9-9