3. RESEARCH METHODOLOGIES: AN INTRODUCTION

3.1 MEANING OF RESEARCH

Research in common parlance refers to a search for knowledge. One can also define research as a scientific and systematic search for pertinent information on a specific topic. In fact, research is an art of scientific investigation. Dictionary definition of research is a careful investigation or inquiry especially through search for new facts in any branch of knowledge. Some people consider research as a movement from the known to the unknown. It is actually a voyage of discovery. We all possess the vital instinct of inquisitiveness. When the unknown confronts us more and more, our inquisitiveness makes us probe and attain understanding of the unknown. This inquisitiveness is the mother of all knowledge and the method, which one employs for obtaining the knowledge of whatever the unknown, can be termed as research.

Research is an academic activity and as such the term should be used in a technical sense. According to Clifford Woody, research comprises defining and redefining problems, formulating hypothesis or suggested solutions; collecting, organising and evaluating data; making deductions and reaching conclusions; and at last carefully testing the conclusions to determine whether they fit the formulating hypothesis.

D. Slesinger and M. Stephenson in the Encyclopaedia of Social Sciences define research as “the manipulation of things, concepts or symbols for the purpose of generalising to extend, correct or verify knowledge, whether that knowledge aids in construction of theory or in the practice of an art.” Research is, thus, an original contribution to the existing stock of knowledge making for its advancement. It is the pursuit of truth with the help of study, observation, comparison and experiment. In short, the search for knowledge through objective and systematic method of finding solution to a problem is research. The systematic approach concerning generalisation and the formulation of a theory is also research. As such the term 'research' refers to the systematic method consisting of enunciating the problem, formulating a hypothesis, collecting the facts or data, analysing the facts and reaching certain conclusions either in the form of solution(s) towards the concerned problem or in certain generalisations for some theoretical formulation.
3.2 OBJECTIVES OF RESEARCH

The purpose of research is to discover answers to questions through the application of scientific procedures. The main aim of research is to find out the truth which is hidden and which has not been discovered as yet. Though each research study has its own specific purpose, we mention some general objectives of research below:

i) To gain familiarity with a phenomenon or to achieve new insights into it (studies with this object in view are termed as exploratory or formulative research studies);

ii) To portray accurately the characteristics of a particular individual, situation or a group (studies with this object in view are known as descriptive research studies);

iii) To determine the frequency with which something occurs or with which it is associated with something else (studies with this object in view are known as diagnostic research studies); and

iv) To test a hypothesis of a causal relationship between variables (such studies are known as hypothesis-testing research studies).

3.3 TYPES OF RESEARCH

Descriptive vs. Analytical: Descriptive research includes surveys and fact-finding enquiries of different kinds. The major purpose of descriptive research is description of the state of affairs as it exists at present. In social science and business research we quite often use the term Ex post facto research for descriptive research studies. The main characteristic of this method is that the researcher has no control over the variables; he can only report what has happened or what is happening. Most ex post facto research projects are used for descriptive studies in which the researcher seeks to measure such items as, for example, frequency of shopping, preferences of people, or similar data. Ex post facto studies also include attempts by researchers to discover causes even when they cannot control the variables.

The methods of research utilized in descriptive research are survey methods of all kinds, including comparative and correlation methods. In analytical research, on the
other hand, the researcher has to use facts or information already available, and analyze these to make a critical evaluation of the material.

**Applied vs. Fundamental**: Applied research aims at finding a solution to an immediate problem of a society or an industrial/business organisation, whereas fundamental research is mainly concerned with generalisation and formulation of a theory. Gathering knowledge for knowledge's sake is termed fundamental research. Research concerning some natural phenomenon or relating to pure mathematics are examples of fundamental research. Similarly, research studies concerning human behaviour carried on with a view to make generalisations about human behaviour, are also examples of fundamental research. However, research aimed at certain conclusions facing a concrete social or business problem is an example of applied research. Research to identify social, economic or political trends that may affect a particular institution, marketing research, evaluation research are examples of applied research. Thus, the central aim of applied research is to discover a solution for pressing practical problems, whereas basic research is directed towards finding information that has a broad base of applications and thus, adds to the already existing organized body of scientific knowledge.

**Quantitative vs. Qualitative**: Quantitative research is based on the quantitative measurements of some characteristics. It is applicable to phenomena that can be expressed in terms of quantities. Qualitative research, on the other hand, is concerned with qualitative phenomenon, i.e., phenomena relating to or involving quality or kind. For instance, when we are interested in investigating the reasons for human behaviour (i.e., why people think or do certain things), we quite often talk of ‘Motivation Research’, an important type of qualitative research. This type of research aims at discovering the underlying motives and desires using in-depth interviews. Other techniques of such research are word association tests, sentence completion tests, story completion tests and similar other projective techniques.

Attitude or opinion research i.e., research designed to find out how people feel or what they think about a particular subject or institution is also qualitative research. Qualitative research is especially important in the behavioural sciences where the aim is to discover the underlying motives of human behaviour. Through such research we can analyse the various factors which motivate people to behave in a particular manner.
or which make people like or dislike a particular thing. It may be stated, however, that to apply for qualitative research in practice is relatively a difficult job and therefore, while doing such research, one should seek guidance from experimental psychologists.

**Conceptual vs. Empirical:** Conceptual research is that related to some abstract idea(s) or theory. It is generally used by philosophers and thinkers to develop new concepts or to reinterpret existing ones. On the other hand, empirical research relies on experience or observation alone, often without due regard for system and theory. It is data-based research, coming up with conclusions which are capable of being verified by observation or experiment. We can also call it as experimental type of research. In such a research it is necessary to get facts at firsthand, at their source, and actively to go about doing certain things to stimulate the production of desired information. In such a research, the researcher must first provide himself with a working hypothesis or guess as to the probable results. He then works to get enough facts (data) to prove or disprove his hypothesis. He then sets up experimental designs which he thinks will manipulate the persons or the materials concerned so as to bring forth the desired information. Such research is thus characterised by the experimenter's control over the variables under study and his deliberate manipulation of one of them to study its effects. Empirical research is appropriate when proof is sought that certain variables affect other variables in some way. Evidence gathered through experiments or empirical studies are considered to be the most powerful support possible for testing a given hypothesis.

**Some other types of research:** All other types of research are variations of one or more of the above stated approaches, based on either the purpose of research, or the time required to accomplish research, on the environment in which research is done, or on the basis of some other similar factors. Form the point of view of time, we can think of research either as one-time research or longitudinal research. In the former case the research is confined to a single time-period, whereas in the latter case the research is carried on over several time periods. Research can be field-setting research or laboratory research or simulation research, depending upon the environment in which it is to be carried out. Research can as well be understood as clinical or diagnostic research. Such research follows case-study methods or indepth approaches to reach the basic casual relations. Such studies usually go deep into the causes of things or events that interest us, using very small samples and very deep probing data gathering devices.
The research may be exploratory or it may be formalized. The objective of exploratory research is the development of hypotheses rather than their testing, whereas formalized research studies are those with substantial structure and with specific hypotheses to be tested. Historical research is that which utilizes historical sources like documents, remains, etc. to study events or ideas of the past, including the philosophy of persons and groups at any remote point of time. Research can also be classified as conclusion-oriented and decision-oriented. While doing conclusion-oriented research, a researcher is free to pick up a problem, redesign the enquiry as he proceeds and is prepared to conceptualize as he wishes. Decision-oriented research is always for the need of a decision maker and the researcher in this case is not free to embark upon research according to his own inclination. Operations research is an example of decision oriented research since it is a scientific method of providing executive departments with a quantitative basis for decisions regarding operations under their control.

3.4 RESEARCH APPROACHES

The above description of the types of research brings to light the fact that there are two basic approaches to research, viz., quantitative approach and the qualitative approach. The former involves the generation of data in quantitative form which can be subjected to rigorous quantitative analysis in a formal and rigid fashion. This approach can be further sub-classified into inferential, experimental and simulation approaches to research.

The purpose of inferential approach is to form a data base to infer characteristics or relationships of population. This usually means survey research where a sample of population is studied (questioned or observed) to determine its characteristics, and it is then inferred that the population has the same characteristics.

Experimental approach is characterised by much greater control over the research environment and in this case some variables are manipulated to observe their effect on other variables.

Simulation approach involves the construction of an artificial environment within which relevant information and data can be generated. This permits an observation of the dynamic behaviour of a system (or its sub-system) under controlled conditions. The
term 'simulation' in the context of business and social sciences applications refers to "the operation of a numerical model that represents the structure of a dynamic process. Given the values of initial conditions, parameters and exogenous variables, a simulation is run to represent the behaviour of the process over time". Simulation approach can also be useful in building models for understanding future conditions.

**Qualitative approach** to research is concerned with subjective assessment of attitudes, opinions and behaviour. Research in such a situation is a function of researcher's insights and impressions. Such an approach to research generates results either in non-quantitative form or in the form which is not subjected to rigorous quantitative and analysis. Generally, the techniques of focus group interviews, projective techniques and depth interviews are used. All these are explained at length in chapters that follow.

### 3.5 SIGNIFICANCE OF RESEARCH

“All progress is born of inquiry. Doubt is often better than overconfidence, for it leads to inquiry, and inquiry leads to invention" is a famous Hudson Maxim in context of which the significance of research can well be understood. Increased amounts of research make progress possible. *Research inculcates scientific and inductive thinking and it promotes the development of logical habits of thinking and organisation.*

*The role of research in several fields of applied economics, whether related to business or to the economy as a whole, has greatly increased in modern times.* The increasingly complex nature of business and governance has focussed attention on the use of research in solving operational problems. Research, as an aid to economic policy, has gained added importance, both for governance and business.

*Research provides the basis for nearly all government policies in our economic system.* For instance, government's budgets rest in part on an analysis of the needs and desires of the people and on the availability of revenues to meet those needs. The cost of needs has to be equated to probable revenues and this is a field where research is most needed. Through research we can devise alternative policies and can as well examine the consequences of each of these alternatives.

Decision-making may not be a part of research, but research certainly facilitates the decisions of the policy maker. Government has to chalk out programmes for dealing with all facets of the country's various operations and most of these are related directly
or indirectly to economic conditions. The plight of cultivators, the problems of big and small business and industry, working conditions, trade union activities, the problems of distribution, even the size and nature of defence services are matters requiring research. Thus, research is considered necessary with regard to the allocation of nation's resources.

Another area in government, where research is necessary, is collecting information on the economic and social structure of the nation. Such information indicates what is happening in the economy and what changes are taking place. Collecting such statistical information is by no means a routine task, but it involves a variety of research problems. These days nearly all governments maintain large staff of research technicians or experts to carry on this work. Thus, in the context of government, research as a tool to economic policy has three distinct phases of operation, viz., (i) investigation of economic structure through continual compilation of facts; (ii) diagnosis of events that are taking place and the analysis of the forces underlying them; and (iii) the prognosis, i.e., the prediction of future developments.

*Research has its special significance in solving various operational and planning problems of business and industry.* Operations research and market research, along with motivational research, are considered crucial and their results assist, in more than one way, in taking business decisions. Market research is the investigation of the structure and development of a market for the purpose of formulating efficient policies for purchasing, production and sales.

Operations research refers to the application of mathematical, logical and analytical techniques to the solution of business problems of cost minimisation or of profit maximisation or what can be termed as optimisation problems. Motivational research of determining why people behave as they do is mainly concerned with market characteristics. In other words, it is concerned with the determination of motivations underlying the consumer (market) behaviour. All these are of great help to people in business and industry who are responsible for taking business decisions.

Research with regard to demand and market factors has great utility in business. Given knowledge of future demand, it is generally not difficult for a firm, or for an industry to adjust its supply schedule within the limits of its projected capacity. Market analysis has become an integral tool of business policy these days. Business budgeting, which
ultimately results in a projected profit and loss account, is based mainly on sales estimates, which in turn depends on business research. Once sales forecasting is done, efficient production and investment programmes can be set up around which are grouped as the purchasing and financing plans. Research, thus, replaces intuitive business decisions by more logical and scientific decisions.

*Research is equally important for social scientists in studying social relationships and in seeking answers to various social problems.* It provides the intellectual satisfaction of knowing a few things just for the sake of knowledge and also has practical utility for the social scientist to know for the sake of being able to do something better or in a more efficient manner. Research in social sciences is concerned with

(i) the development of a body of principles that helps in understanding the whole range of human interactions, and

(ii) the practical guidance in solving immediate problems of human relations.

In addition to what has been stated above, the significance of research can also be understood keeping in view the following perspectives:

(i) To those students who are to write a master's or Ph.D. thesis, research may mean a careerism or a way to attain a high position in the social structure;

(ii) To professionals in research methodology, research may mean a source of livelihood;

(iii) To philosophers and thinkers, research may mean the outlet for new ideas and insights;

(iv) To literary men and women, research may mean the development of new styles and creative work; and

(v) To analysts and intellectuals, research may mean the development of new theories.

Thus, research is the fountain of knowledge for the sake of knowledge and an important source of providing guidelines for solving different business, governmental and social problems. It is a sort of formal training which enables one to understand the new developments in one's field in a better way.
3.6 RESEARCH METHODS Vs METHODOLOGY

It seems appropriate at this juncture to explain the difference between research methods and research methodology. *Research methods* may be understood as all those methods/techniques that are used for conduction of research. *Research methods or techniques, thus, refer to the methods the researchers use in performing research operations.* In other words, all those methods which are used by the researcher during the course of studying his research problem are termed as research methods. Since the object of research, particularly the applied research, it to arrive at a solution for a given problem, the available data and the unknown aspects of the problem have to be related to each other to make a solution possible. Keeping this in view, research methods can be put into the following three groups:

i) In the first group we include those methods which are concerned with the collection of data. These methods will be used where the data already available is not sufficient to arrive at the required solution;

ii) The second group consists of those statistical techniques which are used for establishing relationships between the data and the unknowns; and

iii) The third group consists of those methods which are used to evaluate the accuracy of the results obtained.

Research methods falling in the above stated last two groups are generally taken as the analytical tools of research. At times, a distinction is also made between research techniques and research methods. *Research techniques* refer to the behaviour and instruments we use in performing research operations such as making observations, recording data, techniques of processing data and the like. *Research methods* refer to the behaviour and instruments used in selecting and constructing research technique. For instance, the difference between methods and techniques of data collection can better be understood from the details given in the following Table 3.1:

From what has been stated above, we can say that methods are more general. It is the method that generates techniques. However, in practice, the two terms are taken as interchangeable and when we talk of research methods we do, by implication, include research techniques within their compass.
Research methodology is a way to systematically solve the research problem. It may be understood as a science of studying how research is done scientifically. In it we study the various steps that are generally adopted by a researcher in studying his research problem along with the logic behind them. It is necessary for the researcher to know not only the research methods/techniques but also the methodology. Researchers not only need to know how to develop certain indices or tests, how to calculate the mean, the mode, the median or the standard deviation or chi-square, how to apply particular research techniques, but they also need to know which of these methods or techniques, are relevant and which are not, and what would they mean and indicate.

Table 3.1 Different research methods and techniques for data collection

<table>
<thead>
<tr>
<th></th>
<th>Type</th>
<th>Methods</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Library Research</td>
<td>Analysis of historical records</td>
<td>Recording of notes, Content analysis, Tape and Film listening and analysis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Analysis of documents</td>
<td>Statistical compilations and manipulations, reference and abstract guides, contents analysis.</td>
</tr>
<tr>
<td>2</td>
<td>Field Research</td>
<td>Non-participant direct observation</td>
<td>Observational behavioural scales, use of score cards, etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Participant observation</td>
<td>Interactional recording, possible use of tape recorders, photographic techniques.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mass observation</td>
<td>Recording mass behaviour, interview using independent observers in public places.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mail questionnaire</td>
<td>Identification of social and economic background of respondents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Opinionnaire</td>
<td>Use of attitude scales, projective techniques, use of sociometric scales.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Personal interview</td>
<td>Interviewer uses a detailed schedule with open and closed questions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Focussed interview</td>
<td>Interviewer focuses attention upon a experience and its effects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group interview</td>
<td>Small groups of respondents are interviewed simultaneously.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Telephone survey</td>
<td>Used as a survey technique for information and for discerning opinion; may also be used as a follow up of questionnaire.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Case study and life history</td>
<td>Cross-sectional collection of data for intensive analysis, longitudinal collection of data of intensive character.</td>
</tr>
<tr>
<td>3</td>
<td>Laboratory Research</td>
<td>Small group study of random behaviour, play and role analysis</td>
<td>Use of audio-visual recording devices, use of observers, etc.</td>
</tr>
</tbody>
</table>
The following sections in this chapter explain in detail the models and the methods used for constructing defect predictors. The measure and the corresponding data used in the experiment to evaluate the performance of defect predictors.

### 3.7 DEFECT PREDICTION MODELS

Most of the defect prediction models are based on machine learning. Depending on what to predict (bug-proneness of the number of bugs), models based on machine learning are divided into two types: classification and regression. Since new machine learning techniques are being developed, active or semi-supervised learning techniques have been applied to build better defect prediction models as well Li & Lu (2012). Apart from machine learning models, non-statistical model such as BugCache has been proposed.

![Figure 3.1 Use frequency of defect prediction models in the representative defect prediction](image1)

![Figure 3.2 Use frequency of classification machine learners in the representative defect prediction](image2)
Figure 3.2 shows the use frequency of defect prediction models in the representative defect prediction. Since statistical models based on machine learning studied for a long time, classification and regression models are dominant models. As Kim et al. (2009) proposed BugCache, there were a couple of studies investigating BugCache models as well as case studies. Classification and regression have the similar prediction process since they are based on machine learning. The difference between classification and regression models is what to predict.

Classification models usually identify bug-proneness, according to R. Moser & T. Zimmermann et al. (2009), while regression models predict the number of bugs. The answer for the question, ‘Which model should be used by quality assurance teams?’ is depended on the intended purpose of the model users. Figure 3.2 shows the use frequency of representative machine learners in the literature. Logistic regression is the most frequent machine learners in the representative Naive Bayes and Decision Tree are also frequently used in the literature.

In terms of machine learners for regression models, Linear Regression and Negative Binomial Regression have been mostly used in the literature Weyuker (2010). As new machine learning approaches such as active or semi-supervised learning have been proposed, software engineering community has tried to adopt those approaches for defect prediction. Lu and Cukic (2012) proposed defect prediction models based on active learning where a sample set of instances is selected and the instances are asked to an oracle (human professionals) if the instances could be a good training set. Li et al. (2012) proposed CoFest is a sampling approach based on semi-supervised learning by repeatedly evaluating prediction performance with a random sample through Random Forest to find the best sample ACoFest is an extended version of CoFest by applying active learning.

Kim et al. (2009) proposed BugCache, which maintains the priority of bug-prone entities in a cache. In their evaluation on seven open source projects, 10% of files have 73%-95% of whole defects. The BugCache facilitates locality information of bugs such as temporal and special locality (if a bug of an entity is introduced recently or the entity is changed with other entities, those entities might have bugs with higher change).
3.8 EVALUATION MEASURES

For defect prediction performance, various measures have been used in the literature.

**Measures For Classification**: To measure defect prediction results by classification models, we should consider the following prediction outcomes first:

i) True Positive (TP): buggy instances predicted as buggy.

ii) False Positives (FP): clean instances predicted as buggy.

iii) True Negative (TN): clean instances predicted as clean.

iv) False Negative (FN): buggy instances predicted as clean.

With these outcomes, we can define the following measures that are mostly used in the defect prediction literature.

(i) **False Positive Rate (FPR)**

False positive rate is also known as probability of false (PF) alarm from Menzies (2007). PF measures how many clean instances are predicted as buggy among all clean instances.

\[
\text{FPR} = \frac{FP}{TN + FP}
\]

(ii) **Accuracy**

Accuracy considers both true positives and true negatives over all instances. In other words, accuracy shows the ratio of all correctly classified instances. However, accuracy is not proper measure particularly in defect prediction because of class imbalance of defect prediction datasets. For example, average buggy rate of PROMISE datasets used by Peters et al. (2012) is 18%. If we assume a prediction model that predicts all instances as clean, the accuracy will be 0.82 although no buggy instances are correctly predicted. This does not make sense in terms of defect prediction performance. Thus, accuracy has not been recommended for defect prediction Rahman (2012).

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}
\]

(iii) **Precision**

\[
\text{Precision} = \frac{TP}{TP + FP}
\]
(iv) Recall

Recall is also known as probability of detection (PD) or true positive rate (TPR) from Menzies (2007). Recall measures correctly predicted buggy instances among all buggy instances.

\[
\frac{TP}{TP + FN}
\]

(v) F-measure

F-measure is a harmonic mean of precision and recall from Lee (2011).

\[
\frac{2 \times (Precision \times Recall)}{Precision + Recall}
\]

Since precision and recall have trade-offs, f-measure has been used by Wu (2011).

(vi) Area Under the Curve (AUC)

AUC measures the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve is plotted by PF and PD together. Figure 3.3 explains about a typical ROC curve. PF and PD vary based on threshold for prediction probability of each classified instance. By changing the threshold, we can draw a curve as shown in Figure 3.3. When the model gets better, the curve tends to be close to the point of PD=1 and PF=0. Thus, AUC of the perfect model will have “1”. For a random model, the curve will be close to the straight line from (0, 0) to (1, 1) Menzies (2007), AUC with 0.5 is regarded as the random prediction. Other measures such as precision and recall can vary according to prediction threshold values. However, AUC considers prediction performance in all possible threshold values. In this reason, AUC is a stable measure to compare different prediction models Rahman (2012).
Figure 3.3 A typical ROC curve from Menzies (2007)

Area Under Cost-Effectiveness Curve (AUCEC)

AUCEC is a defect prediction measure considering lines of code (LOC) to be inspected or tested by quality assurance teams or developers. The idea of cost-effectiveness for defect prediction models is proposed by Arisholm et al. (2007). Cost-effectiveness represents how many defects can be found among top n% LOC inspected or tested. In other words, if a certain prediction model can find more defects with less inspecting and testing effort comparing to other models, we could say the cost-effectiveness of the model is higher.

Figure 3.4 shows cost-effectiveness curves as examples. The x-axis represents the percentage of LOC while y-axis represents the percentage of defects found. F. Rahman (2011). In the left side of the figure, three example curves are plotted. Let assume the curves, O, P, and R, represent the optimal, practical, and random models, respectively. If we consider the area under the curve, the optimal model will have the highest AUCEC comparing to other models. In case of the random model, AUCEC will be 0.5. The higher AUCEC of the optimal model means that we can find more defects by inspecting or testing less LOC than other models.

However, considering AUCEC from the whole LOC may not make sense. In the right side of Figure 3.4, the cost-effectiveness curves of the models, P1 and P2, are identical so that considering the whole LOC for AUCEC does not give any meaningful insight. Having said that, if we set a threshold as top 20% of LOC, the model P2 has higher AUCEC than R and P2. In this reason, we need to consider a particular threshold for the percentage of LOC to use AUCEC as a prediction measure.
B. Measures for regression

To measure defect prediction results from regression models, measures based on correlation calculation between the number of actual bugs and predicted bugs of instances have been used in many defect prediction papers (A. Bacchelli, 2010), (N. Nagappan, 2006), (Y. Shin, 2011). The representative measures are Spearman’s correlation, Pearson correlation, R² and their variations. These measures also have been used for correlation analysis between metric values and the number of bugs (T. Zimmermann, 2008).

3.9 DISCUSSION ON MEASURES

Figure 3.5 shows the count of evaluation measures used in the representative defect prediction papers for classification. As shown in the figure, f-measure is the most frequently used measure for defect classification. Since there are trade-off between precision and recall, comparing different models are not easy as some models have high precision but low recall and vice versa for other models. Since f-measure is a harmonic mean of precision and recall and provides one.
Figure 3.5 Count of measures used in the representative defect prediction for classification

Single score as prediction performance, f-measure have been used to compare different prediction models in many defect prediction datasets. However, f-measure varies by different thresholds (cut-offs) for prediction probability of an instance. When a model predicts an instance as buggy, it provides prediction probability that represents if the instance is buggy. Lessmann et al. (2008) pointed out that what thresholds were used in evaluation usually overlooked in the defect prediction literature so that it led to inconsistent prediction results across defect prediction datasets.

To overcome the limitation of f-measures, researchers also used other measures such as AUC and AUCEC that are independent from the thresholds. Particularly, AUCEC recently used a lot by Rahmann et al. (2012) since AUCEC could be a good measure to evaluate prediction models in the view of practical use of models.

3.10 DEFECT PREDICTION METRICS

Defect prediction metrics play the most important role to build a statistical prediction model. Most defect prediction metrics can be categorized into two kinds: code metrics and process metrics. Code metrics are directly collected existing source code while process metrics are collected from historical information archived in various software repositories such as version control and issue tracking systems.

**Code Metrics**

Code metrics also known as product metrics, measure complexity of source code. Its ground assumption is that complexity source is more bug-prone. To measure code complexity, researchers proposed various metrics.
Table 3.2 Representative code metrics

<table>
<thead>
<tr>
<th>Code Metrics</th>
<th>Size Authors</th>
<th>Halstead Authors</th>
<th>McCabe Authors</th>
<th>CK Authors</th>
<th>OO Authors</th>
</tr>
</thead>
</table>

The size metrics measure “volume, length, quantity, and overall magnitude of software products” (S. D. Conte, 1986). The representative of size metrics is lines of code (LOC). To our knowledge, Akiyama’s model was the earliest study to predict defects using LOC (F. Akiyama, 1971). Afterwards, LOC was used in most defect prediction papers to build a model.

Halstead proposed several size metrics based on the number of operators and operands (M. H. Halstead, 1977). The proposed metrics are program vocabulary, length, volume, difficult, effort, and time. Most metrics are related to size or quantity.

McCabe proposed the cyclomatic metric to represent complexity of software products T. McCabe (1976). Cyclomatic metric is computed by the number of nodes, arcs and connected components in control flow graphs of source code. This metric represents how much control paths are complex. Since McCabe’s cyclomatic metric measure the complexity of source code structure, its characteristic is inherently different from size and Halstead metrics which measures volume and quantity of source code. Ohlsson and Alberg adopted McCabe’s cyclomatic metric to predict fault-prone modules in telephone switched N. Ohlsson (1996) and other defect prediction studies also used McCabe’s cyclomatic metric to build a prediction model.

Since object-oriented programming is getting popular, code metrics for object-oriented languages have proposed to improve development process. The representative metrics for object oriented programs are Chidamber and Kemerer (CK) metrics S. R. Chidamber (1994). Table 3.3 lists CK metrics.

The CK metrics were designed from the characteristics from object-oriented languages such as inheritance, coupling, and cohesion. Basili et al. (1996) validated if it is possible to build a defect prediction model using the CK metrics. After that, many studies include CK metrics to build prediction models.
Table 3.3 CK metrics (S. R. Chidamber, 1994)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC</td>
<td>Weights methods per class</td>
</tr>
<tr>
<td>DIT</td>
<td>Depth of inheritance tree</td>
</tr>
<tr>
<td>NOC</td>
<td>Number of children</td>
</tr>
<tr>
<td>CBO</td>
<td>Coupling between object classes</td>
</tr>
<tr>
<td>RFC</td>
<td>Response for a class</td>
</tr>
<tr>
<td>LCOM</td>
<td>Lack of cohesion in methods</td>
</tr>
</tbody>
</table>

Besides the CK metrics, other object-oriented (OO) metrics based on volume and quantity of source code, have been proposed as well (F. B. e Abreu, 1994). As size metrics, the OO metrics simply counts the number of instance variables, methods as shown in Table 3.4. Many defect prediction studies for object-oriented programs have used the OO metrics to build prediction models (M. D’Ambros, 2012), (S. Kim, 2011), (T. Lee, 2011), (G. Pai, 2007), (R. Wu, 2011), (H. Zhang, 2010), (T. Zimmermann, 2008).

**Process Metrics**

Table 3.5 lists seven representative process metrics. In this section, we briefly introduce the metrics and their design concepts.

(i) **Relative Code Change Churn**

Nagappan and Ball (2005) proposed 8 relative code churn metrics (M1-M8) measuring the amount of code changes. For example, M1 metric is computed by churned LOC (the accumulative number of deleted and added lines between a base version and a new version of a source file) divided by Total LOC. Other metrics (M2-M8) consider various normalized changes such as deleted LOC divided by total LOC, file churned (the number of changed files in a component) divided by file count, and so on. In the case study by Nagappan and Ball, the relative churn metrics is proved as a good predictor to explain the defect density of a binary and bug-proneness.
Table 3.4 Example class-level object oriented metrics used by D’amabros et al. (2012)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FanIn</td>
<td>Number of other classes that reference the class</td>
</tr>
<tr>
<td>FanOut</td>
<td>Number of other classes referenced by the class</td>
</tr>
<tr>
<td>NOA</td>
<td>Number of attributes</td>
</tr>
<tr>
<td>NOPA</td>
<td>Number of public attributes</td>
</tr>
<tr>
<td>NOPRA</td>
<td>Number of private attributes</td>
</tr>
<tr>
<td>NOAI</td>
<td>Number of attributes inherited</td>
</tr>
<tr>
<td>LOC</td>
<td>Number of lines of code in a class</td>
</tr>
<tr>
<td>NOM</td>
<td>Number of methods</td>
</tr>
<tr>
<td>NOPM</td>
<td>Number of public methods</td>
</tr>
<tr>
<td>NOPRM</td>
<td>Number of private methods</td>
</tr>
<tr>
<td>NOMI</td>
<td>Number of methods inherited</td>
</tr>
</tbody>
</table>

Table 3.5 Representative process metrics

<table>
<thead>
<tr>
<th>Process Metrics</th>
<th># of Metrics</th>
<th>Metric Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative code change churn (N. Nagappan, 2005)</td>
<td>8</td>
<td>Version control</td>
</tr>
<tr>
<td>Change (R. Moser, 2008)</td>
<td>17</td>
<td>Version control</td>
</tr>
<tr>
<td>Change Entropy (A. E. Hassan, 2009)</td>
<td>1</td>
<td>Version control</td>
</tr>
<tr>
<td>Code metric churn, Code Entropy (M. D’Ambros, 2010), (M. D’Ambros, 2012)</td>
<td>2</td>
<td>Version control</td>
</tr>
<tr>
<td>Popularity (A. Bacchelli, 2010)</td>
<td>5</td>
<td>E-mail archive</td>
</tr>
<tr>
<td>Ownership (C. Bird, 2011)</td>
<td>4</td>
<td>Version control</td>
</tr>
<tr>
<td>Micro interaction metrics (T. Lee, 2011)</td>
<td>56</td>
<td>Mylyn</td>
</tr>
</tbody>
</table>

(ii) Change Metrics

Change metrics are to measure the extent of changes in the history recorded in version control systems. For example, we can count the number of revisions/bug-fix changes/refactoring of a file and the number of authors editing a file. Moser et al. (2008) extracted 18 change metrics from the Eclipse repositories to conduct a comparative analysis between code and change metrics. Moser et al.’s change metrics also include added and deleted LOC similar to relative code change churn. However, Moser et al.’s change metrics did not consider any relativeness by the total LOC and the file count but consider average and maximum values of change churn metrics. Moser et al.’s metrics also include maximum and average of change sets (the number of files committed together) and age metrics (age of a file in weeks and the weighted age normalized by added LOC) R. Moser (2008). Moser et al. concluded that change metrics are better predictors than code metrics.
(iii) Change Entropy

Hassan applied Shannon’s entropy to capture how changes are complex and proposed history complexity metric (HCM) A. E. Hassan (2009). To validate the HCM, Hassan built statistical linear regression models based on HCM or two change metrics, the number of previous modifications and previous faults on six open-source projects. Their evaluation on six open-source projects showed that prediction models build using HCM outperform those using the two change metrics. The idea adopting the Entropy concept to measure change complexity is novel but comparing models by HCM to those by only two change metrics reveals the weakness of the evaluation of HCM. In addition, evaluation was conducted in the subsystem-level rather than the file-level.

(iv) Code metric churn, code entropy

D’Ambros et al. conducted extensive comparisons in the study of defect prediction metrics (2010). In their metric comparison, there is no study about code metric churn and code Entropy while code churn and change Entropy metrics have studied as introduced in previous subsections. Thus, D’Ambros et al. proposed code metric churn (CHU) and code Entropy (HH) metrics.

In contrast to code change metrics based on the amount of lines, CHU measures the change in biweekly basis of code metrics such as CK metrics and OO metrics. Since code metric churn computes the amount of changes in biweekly basis, CHU captures the extent of changes more precisely than code change churn that computes the amount of changes between a base revision and a new revision. Four variants of CHU by applying decay functions (WCHU, LDCHU, EDCHU, and LGDCHU) also were proposed M. D’Ambros (2010).

While change Entropy is computed based on the count of file changes, code Entropy (HH) is computed based on the count of involved files when a certain code metric is changed. As in CHU, D’Ambros et al. also defined four variants of HH by applying decay functions (HWH, LDHH, EDHH, and LGDHH) M. D’Ambros (2010).

In the comparison evaluation, D’Ambros et al. concluded that WCHU and LDHH metrics led to good prediction results on all subjects used in their experiments M. D’Ambros (2010). However, limitations of these novel metrics is heavy computation resources and data because it tracks biweekly changes from version control systems M. D’Ambros (2010).
(v) Popularity

A. Bacchelli et al. (2010) proposed popularity metrics by analyzing e-mail archives by developers in a group mailing list. The main idea of popularity metrics is more discussed software artifacts in e-mail archives are more bug-prone and given in the Table 3.6 lists as the popularity metrics. Most of the metrics quantify how many times a certain class are discussed in E-mails. The extracting metrics from e-mail archives is novel but their evaluation of the metrics shows that popularity metrics themselves did not outperform other code and process metrics.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POP-NOM</td>
<td>The number of mails discussing a class</td>
</tr>
<tr>
<td>POP-NOCM</td>
<td>The number of characters in all mail discussing a class</td>
</tr>
<tr>
<td>POP-NOT</td>
<td>The number of e-mail threads discussing different topics for a class</td>
</tr>
<tr>
<td>POP-NOMT</td>
<td>The number of e-mails in a thread discussing a class in at least one of mails in a thread</td>
</tr>
<tr>
<td>POP-NOA</td>
<td>The number of authors motioning about the same class</td>
</tr>
</tbody>
</table>

(vi) Ownership and Authorship

Bird et al. (2011) proposed four ownership metrics based on authorship of a component. This study started from the question, “How much does ownership affect quality?”. The ownership of a component is defined by the portion of commits of the component and minor and major contributors are defined by less than and more than 5% portions of the ownership respectively. Four ownership metrics are defined as follows; MINOR (the number of minor contributors), MAJOR (the number of major contributors), TOTAL (the total number of contributors), and OWNERSHIP (portion of ownership of the contributor with the highest portion of ownership). They concluded that higher ownership leads to less bug-prone. Rahman et al. (2011) conducted the fine-grained investigation on relationship between defects and human factors such as ownership and developer experience. The most interesting finding of this study is that quality assurance should be focused on source code files touched by less experienced developers.
(vii) **Micro Interaction Metrics**

Lee et al. proposed micro interaction metrics (MIM) extracted from Mylyn that captures developer interactions to Eclipse T. Lee (2011). The main idea of MIM is from the fact that defects could be introduced by mistakes of developers. For example, more editing time of a certain source code file may cause more bug-proneness. Since Mylyn data contains developer’s interactions to Eclipse, he extracted 56 metrics from Mylyn data and compared their performance with code and process metrics. In their experiments, MIM outperformed code and process metrics in both classification and regression. However, MIM is highly depended on Mylyn, a plug-in of Eclipse so that MIM might be hard to apply for other development environments that does not support the tool like Mylyn T. Lee (2011).

**Other Metrics**

Apart from code and process metrics, researchers proposed new kinds of metrics based on existing knowledge such as network measure by A. Meneely *et al.* (2008), M. Pinzger *et al.* (2008) and anti-pattern S. E. S. Taba (2013). Meneely *et al.* (2008) extracted developer metrics from a developer social network that represents collaboration structure extracted from source code repositories. Based on this developer social network, this study found that software failure is highly correlated with developer network metrics A. Meneely *et al.* (2008). Pinzger *et al.* (2008) also constructed developer network but also with software modules, i.e., developer-module network. This network represents how each developer contributes to each module so that the network is called ‘contribution network’ as well M. Pinzger *et al.* (2008) found that the centrality measures for the contribution network can predict significantly post-release defects. Zimmermann *et al.* (2008) constructed dependency (such as data and call dependencies) graphs of binaries and conducted network analysis on those dependency graphs. From various network analysis measures such as centralness, closeness, betweenness, and so on, Zimmermann *et al.* (2008) built prediction models and compared them to models constructed by code and process metrics. In their evaluation, network measure could predict more bug-prone binaries than code and process metrics.
Taba et al. (2013) proposed four anti-pattern metrics. Anti-patterns are poor design of software so that there might be higher chance to introduce defects in the source code files. In their evaluation with two open source projects, anti-pattern metrics could improve prediction performance in terms of f-measure.

Figure 3.6 Use frequency of defect prediction metrics in representative defect prediction

3.11 CODE METRICS Vs. PROCESS METRICS

Figure 3.6 shows the use frequency of metrics in the representative defect prediction papers. Since code metrics such as size, Hastead, McCabe, CK and OO metrics have used from 1970s or 1990s, absolute use frequency of code metrics is higher than process metrics. In addition, code metrics used a lot for comparison study whenever new kinds of metrics are proposed. Most process metrics have been proposed in 2000s from when software repositories such as version control and issue tracking systems get popular.

There are lots of debates if code metrics are good defect predictors and process metrics are better than code metrics. Menzies et al. (2007) confirmed that code metrics are still useful to build a defect prediction model. However, according to F.Rahman et al. (2013) recent study comparing code and process metrics, code metrics is less useful than process metrics because of stagnation of code metrics.

3.12 PREDICTION GRANULARITY

In the literature, defect prediction models were constructed in various levels of granularity such as sub-system. N. Fenton (2008). Since the resource allocation for software quality assurance can be conducted by quality assurance teams’ own focus, studies on defect prediction models seem to be conducted on various granularity levels.
A recent study by Hata et al. (2012) proposed method-level defect prediction and concluded that method-level defect prediction is more cost-effective than other higher granularity levels such as package- and file-levels. S.Kim et al. (2008) proposed a novel defect prediction model called change classification. Different from common defect prediction models, change classification can be directly helpful to developers since a change classification model can provide an instant prediction result whenever a developer makes any change on source code files and commit it to a version control system. However, change classification models are too heavy to use in practice since the models are built by more than ten thousand features. To use defect prediction models in practice, we should consider cost-effectiveness F. Rahman, (2012). One of the ways to improve cost-effectiveness of prediction models is to predict defects in finer-grained levels H. Hata et al.(2012). In this sense, researchers need to more focus on defect prediction on finer-grained levels such as line-level defect prediction and change classification.

3.13 PRE-PROCESSING FOR DEFECT DATA SETS

Pre-processing is a widely used step in machine learning. Since most defect prediction studies are based on machine learning, there are several studies using preprocessing techniques S. Kim (2011). Depending on each study, preprocessing techniques are selectively used or not used since many studies are conducted by different metrics, models, and subjects.

**Normalization**

Normalization is a common technique to give the same weight for metric values to improve performance of classification models. Menzies et al.(2007) recommended to use the logarithmic filter (log-filter) to normalize metric values for metrics that have exponential distribution. Other studies using the same experimental subjects used by also applied the log-filter. Nam et al. (2013) observed that cross-prediction performance varies from different normalization techniques. Nam et al. defined rules for selecting proper normalization techniques such as min-max normalization, z-score and variations of z-score to improve the performance of cross-predictions.

**Feature Selection and Extraction**

Shivaji et al. (2013) pointed out that the poor performance of defect prediction models is due to a number of metrics to build the models. In this reason, he proposed a feature
selection technique that can improve change classification and also applied feature subset selection using information gain before conducting cross-company defect prediction (B. Turhan 2009). Since defect prediction datasets may have the multi co-linearity issue, researchers applied principal component analysis (PCA) to extract new features for prediction models.

**Noise Reduction**

Since defect data are usually collected from version control and issue tracking systems automatically by using tools and algorithms, defect data may be bias as in Bird et al. (1996) proposed to reduce noise. Wu et al. (2010) proposed ReLink that automatically recovers the correct links between commit logs and issue IDs. R. Kim et al. (2009) proposed a noise detection and elimination algorithm called Closest List Noise Identification (CLNI). S. Kim et al. (2011).

### 3.14 COMPARISON OF PROPOSED DEFECT PREDICTION WITH EXISTING MODELS

Since new software projects do not have enough training data, building a good prediction model for the new projects is a challenging issue. For example, Zimmermann et al. (2009) conducted 622 cross-predictions and found only 3.4% actually worked. To improve cross-prediction performance, researchers focused on studies based on transfer learning and cross-prediction feasibility.

**Transfer Learning**

Transfer learning is one of very active research areas in machine learning S. J. Pan (2010). Figure 3.7 explains difference between traditional machine learning and transfer learning. Traditional machine learning assumes that distribution of training and test data is same. Thus, in case that the distribution changed, it is required to rebuild the model with newly collected data. However, collecting new data and labelling them is cost-expensive. By this reason, researchers in machine learning community have focused on transfer learning where we can transfer knowledge from a domain with enough training data to another domain with few training data to build a leaning model as shown in Figure 3.7. Software engineering community has been adopted transfer learning concepts and techniques for cross-project defect prediction. Table 3.7 summarizes the representative works for cross-project defect prediction using transfer learning.
Table 3.7 Cross-project defect prediction based on transfer learning techniques

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing</td>
<td>N/A</td>
<td>Feature selection, Log-filter</td>
<td>Log-filter</td>
<td>Normalization</td>
</tr>
<tr>
<td>Machine learner</td>
<td>C4.5</td>
<td>Naive Bayes</td>
<td>TNB</td>
<td>Logistic Regression</td>
</tr>
<tr>
<td>#Subjects</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>#Predictions</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>Avg. F-measure</td>
<td>0.67 (W:0.79,C:0.58)</td>
<td>0.35 (W:0.37,C:0.26)</td>
<td>0.39 (NN:0.35,C:0.33)</td>
<td>0.46 (W:0.46,C:0.36)</td>
</tr>
</tbody>
</table>

a) Metric Compensation

Watanabe et al. (2008) applied metric compensation to transform target data by using source data. The main idea of metric compensation is to normalize each metric (feature) value of target data by using the average metric value of corresponding metric of source data. In detail metric compensation is designed as follows:

\[
newv_{T,i,j} = \frac{VT_{i,j} \times avg_{v_{s,i}}}{avg_{v_{T,i}}}
\]
where $\text{new}_{V_{T,i,j}}$ is each compensated metric value of $i$-th instance of $j$-th metric of target, $v_{T,i,j}$ is each metric value of $i$-th instance of $j$-th metric of target, and $\text{avg}_{V_{S,j}}$ and $\text{avg}_{V_{T,j}}$ are average values of $j$-th metric of source and target respectively.

With two project datasets, Watanabe et al. (2009) conducted cross-project predictions and reported precision and recall in the research. We computed average $f$-measure for cross-predictions with/without metric compensation and within-predictions (W) as shown in Table 3.7. Average $f$-measure (0.67) of cross-predictions with metric compensation outperforms that (0.58) of cross-predictions without metric compensation but still is worse than that (0.79) of within-predictions.

The major limitation of the work conducted by Watanabe et al. (2008) is the weak evaluation of their approach; only two cross-predictions were conducted and there is no statistical test to validate their research questions.

b) NN Filter

Turhan et al. (2009) applied the nearest neighbour filter (NN filter) to improve performance of cross-company defect prediction. The basic idea of the NN filter is to collect similar source instances to target instances to train a prediction model. In other words, if we can build a prediction model using selected source instances that have similar data characteristics to target instances, the model may perform better on predicting target instances than the model trained by using all source instances. The NN filter chooses 10 source instances as nearest neighbours for each target instance.

To evaluate performance of cross-company defect prediction using the NN filter, conducted experiments with ten proprietary datasets from NASA and SOFTLAB. In addition, they conducted Mann-Whitney U test to validate their experimental results. As we computed average $f$-measure from their PD and PF results, the average $f$-measure (0.35) in cross-predictions with the NN filter is better than that (0.26) without the NN filter. However, within-predictions were still the best comparing to cross-predictions.

c) Transfer Naive Bayes

Ma et al. (2012) proposed Transfer Naive Bayes (TNB) for cross-company defect prediction. The basic idea of TNB is to compute new prior probability and conditional probability of Naive Bayes model by using the weight value of a source instance. The weight of the source
instance is computed according to the similarity between the source instance and target instances.

To compute the instance similarity between a source instance and target instances, min and max values of each feature of a test dataset is used. In other words, the similarity is computed by the number of features of a source instance whose feature values are in between min and max values of a corresponding target feature. Then, the weight value is computed by using Newton’s Universal Gravitation law. The more weight of a training instance means the more similarity to test instances. Finally, TNB model is computed with new prior and conditional probabilities by using these weights for source instances. As shown in Table 3.6, TNB (0.39) led to better prediction performance than NN filter (0.35) in terms of average f-measure. Please note different cross-prediction (CC) results without NN filter by Turhan et al. (2009) and Ma et al. (2012). The reason is that Ma et al. did not apply feature selection in their experimental setting. The limitation of this study is that TNB is not applicable for other machine learning algorithms that do not use prior and conditional probabilities. In addition, Ma et al. (2012) did not report within-prediction results so that we could not conclude cross-prediction using TNB is comparable with within-prediction.

d) TCA+

Nam et al. proposed TCA+ for cross-project defect prediction J. Nam (2013). TCA+ is an extended version of transfer component analysis (TCA) that is a state-of-the-art transfer learning algorithm proposed by Pan et al. (2010). TCA tries to find a common latent feature space where the distribution of source and target datasets are similar by using projection. Projection is a feature extraction technique to reduce feature space in machine learning. Principal component analysis (PCA) is a representative feature extraction approach by projecting instances in lower dimensional space. While PCA tries to keep original data characteristics on lower dimensional feature space, TCA tries to find a lower dimensional feature space where source and target data have similar distribution as well as to keep original data characteristics as PCA does. Figure 3.8 clearly shows how PCA and TCA results are different. PCA result shown in the center of the figure explains that distributions between source (red) and target (blue) are still different in the latent feature space. However, TCA result in the right side of the figure shows that distribution between source and target is similar in the new latent feature space. Applying TCA for
cross-predictions, Nam et al. (2013) observed that prediction performance varies from what normalization approach is applied for preprocessing.

Figure 3.8 shows instances in the original feature space (two dimensional). Nam et al. (2013) proposed TCA+ by adding decision rules to select proper normalization options into TCA. For normalization options, min-max, z-score, and variations of z-score are used in their experiments J. Nam (2013). As shown in Table VI, cross-prediction result (0.46) in TCA+ show comparable to within-prediction result (0.46) in terms of average f-measure.

e) Discussion on Cross-Predictions based on Transfer Learning

Until now, we compare various approaches to cross-project defect prediction. The main goal of cross-project defect prediction is to reuse existing defect datasets to build a prediction model for a new project or a project lacking in the historical data. However, all approaches discussed above could conduct cross predictions across datasets with the same feature space. As shown in Table 3.6 the number of subjects used in TCA+ is 8 but the number of predictions are 26. If we consider all possible cross-prediction combinations, it should be 56. However, it could not be done because of the different feature space of datasets. In TCA+ experiments, the size of feature space of three datasets is 26 while that of five datasets is 61. Thus, cross-predictions could be possible within the datasets with the same feature space, ie. 6 cross-predictions (= 3 x (3 - 1)) and 20 cross-predictions (= 5 x (5 - 1)). Achieving cross-predictions on datasets with different feature.

Cross-Prediction Feasibility

There are few studies on cross-project feasibility Zimmermann et al. (2009) built a decision tree to validate cross-project predictability by using project characteristics such as
languages used and number of developers. However, the decision tree was constructed and validated within the subjects used in their empirical study so that the decision tree could not be used general purpose.

He et al. (2012) also constructed the decision tree based on cross prediction results to validate cross-project feasibility. Their decision tree is built by difference of distributional characteristics of source and target datasets such as mean, median, variance, skewness and so on. However, validation of the decision tree is conducted on the best prediction results on different samples of training sets so that the validation results do not fully support its validity.

3.15 APPLICATIONS ON DEFECT PREDICTION

One of the major goals of defect prediction models is effective resource allocation for inspecting and testing software products. However, the case studies using defect prediction models in industry is few. In this reason, many studies by Rahman et al. (2012), considers cost-effectiveness. A recent case study conducted in Google by Lewis et al. (2013) comparing BugCache and Rahman’s algorithm based on the number of closed bugs found that developers preferred Rahman’s algorithm. However, developers still did not get benefits from using defect prediction models C. Lewis (2013).

One of the recent studies conducted by Rahman (2014) showed that defect prediction could be helpful to prioritize warnings reported by static bug finders such as FindBug. An another possible application is that we can apply defect prediction results to prioritize or select test cases. In regression testing, executing all test suites for regression testing is very costly so that many prioritization and selection approaches for test cases have been proposed S. Yoo (2012). Since defect prediction results provide bug-prone software artefacts and their ranks T. Zimmermann (2008), it might be possible to use the results for test case prioritization and selection.

3.16 OTHER EMERGING TOPICS

Apart from the representative papers discussed in previous sections, there are interesting and emerging topics in defect prediction study. One topic is about defect data privacy and the other topic is the comparative study between defect prediction models and static bug finders F. Rahman (2014).
Defect Data Privacy

Peters et al. (2012) proposed MORPH that mutates defect datasets to resolve privacy issue in defect datasets. To accelerate cross-project defect prediction study, publicly available defect datasets are necessary. However, software companies are reluctant to share their defect datasets because of “sensitive attribute value disclosure”. Thus, cross-project defect prediction studies usually conducted on open source software products or very limited proprietary systems. Experiments conducted by Zimmermann et al. for cross-project defect prediction are not reproducible since Microsoft defect datasets are not publicly available. To address this issue, MORPH moves instances in a random distance by still keeping class decision boundary. In this way, MORPH could privatize original datasets and still achieve good prediction performance as in models trained by original defect datasets.

Comparing Defect Prediction Models to Static Bug Finders

In contrast with defect prediction models (DP), static bug finders (SBF) detect bugs by using “semantic abstractions of source code”. Compared defect prediction techniques and static bug finders in terms of cost-effectiveness F. Rahman (2014) found that DP and SBF could compensate each other since they may find different defects. In addition, SBF warnings prioritized by DP could lead to better performance than SBF’s native priorities of warnings. This comparative study provided meaningful insights that explains how different research streams have the same goal can be converged together to achieve the better prediction/detection of defects.

3.17 CHALLENGING ISSUES

Defect predictions studies are still have many challenging issues. Even though there are many outstanding studies, it is not easy to apply those approaches in practice because of following reasons:

Most studies were verified in open source software projects so that current prediction models may not work for any other software products including commercial software. However, proprietary datasets are not publicly available because of privacy issue F. Peters et al. (2012). Although he proposed MORPH algorithm to increase data privacy, MORPH was not validated in cross-project defect prediction. Investigating privacy issue in cross-project defect prediction is required since if we have more available proprietary datasets, evaluation of prediction models will be more sound.
Cross prediction is still a very difficult problem in defect prediction in terms of two aspects. Different feature space: There are many publicly available defect datasets. However, we cannot use many of datasets for cross prediction since datasets from different domains have different number of metrics (features). Prediction models based on machine learning cannot be built on the datasets, which have different feature spaces. Feasibility: Studies on cross prediction feasibility are not mature yet. Finding general approaches to check the feasibility in advance will be very helpful for practical use of cross prediction models.

Since software projects are getting larger, file-level defect prediction may not be enough in terms of cost-effectiveness. There are still few studies for finer prediction granularity. Studies on finer-grained defect prediction such as line-level defect prediction and change classification are required.

Defect prediction metrics and models proposed until now may not always guarantee generally good prediction performance. As software repositories evolve, we can extract new types of development process information, which never used for defect prediction metrics/models. New metrics and models need to be kept investigating.
4. CLASSIFIER MINING OF SOFTWARE DEFECT PREDICTION

4.1 INTRODUCTION

Though there has been a rapid growth in software development, owing to various reasons, software comes with many defects. In Software development process, testing of software is the main phase which reduces the defects of the software. If a developer or a tester can predict the software defects properly, it will reduce the cost, time and effort. We proposed to undertake a comparative analysis of software defect prediction based on classification rule mining. We developed a scheme for this purpose and chose different classification algorithms for comparison. This evaluation analyzes the prediction performance of competing learning schemes for given historical data sets (NASA MDP Data Set). The result of this scheme evaluation shows that we have to choose different classifier rule for different data set.

*The content of this chapter is published in Journal of Computer Science and Engineering (3785 – 0855), P32-45, Volume 2, Issue 3, April 2015, The reprint is appended towards the end of this thesis.*

To improve the software productivity and quality, software engineers are applying data mining algorithms to various SE tasks. Many algorithms can help engineers figure out how to invoke API methods provided by a complex library or framework with insufficient documentation. In terms of maintenance, such type of data mining algorithms can assist in determining what code locations must be changed when another code location is changed. Software engineers can also use data mining algorithms to hunt for potential bugs that can cause future in-field failures as well as identify buggy lines of code (LOC) responsible for already-known failures. The second and third columns of Table 4.1 list several example data mining algorithms and the SE tasks to which engineers apply by Tao Xie (2009).
4.2 PROPOSED SCHEME

Overview of the framework before building defect prediction model and using them for prediction purposes, we first need to decide which learning scheme or learning algorithm should be used to construct the model. Thus, the predictive performance of the learning scheme should be determined, especially for future data. However, this step is often neglected and so the resultant prediction model may not be Reliable. As a consequence, we use a software defect prediction framework that provides guidance to address these potential shortcomings.

The framework consists of two components:

i) Scheme evaluation, and

ii) Defect prediction.

Figure 4.1 contains the details. At the scheme evaluation stage, the performances of the different learning schemes are evaluated with historical data to determine whether a certain learning scheme performs sufficiently well for prediction purposes or to select the best from a set of competing schemes.

![Figure 4.1 Proposed Framework](image)
From Figure 4.1, we can see that the historical data are divided into two parts: a training set for building learners with the given learning schemes, and a test set for evaluating the performances of the learners, it is very important that the test data are not used in any way to build the learners. This is a necessary condition to assess the generalization ability of a learner that is built according to a learning scheme and to further determine whether or not to apply the learning scheme or select one best scheme from the given schemes.

At the defect prediction stage, according to the performance report of the rst stage, a learning scheme is selected and used to build a prediction model and predict software defect. From Figure 4.1, we observe that all of the historical data are used to build the predictor here. This is very different from the first stage; it is very useful for improving the generalization ability of the predictor. After the predictor is built, it can be used to predict the defect-proneness of new software components.

MGF proposed by Tim Menzies (2007) a baseline experiment and reported the performance of the Naive Bayes data miner with log- filtering as well as attribute selection, which performed the scheme evaluation but with in appropriate data. This is because they used both the training (which can be viewed as historical data) and test (which can be viewed as new data) data to rank attributes, while the labels of the new data are unavailable when choosing attributes in practice.

4.3 SCHEME EVALUATION

The scheme evaluation is a fundamental part of the software defect prediction framework. At this stage, different learning schemes are evaluated by building and evaluating learners with them. The first problem of scheme evaluation is how to divide historical data into training and test data. As mentioned above, the test data should be independent of the learner construction. This is a necessary precondition to evaluate the performance of a learner for new data. Cross-validations usually used to estimate how accurately a predictive model will perform in practice. One round of cross-validation involves partitioning a data set into complementary subsets, performing the analysis on one subset, and validating the analysis on the other subset. To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

In our framework, a percentage split used for estimating the performance of each predictive model, that is, each data set is first divided into 2 parts, and after that a
predictor is learned on 60% instances, and then tested on the remaining 40%. To overcome any ordering effect and to achieve reliable statistics, each holdout experiment is also repeated M times and in each repetition the data sets are randomized. So overall, $M*N$ (N=Data sets) models are built in all during the period of evaluation; thus $M*N$ results are obtained on each data set about the performance of the each learning scheme.

After the training-test splitting is done each round, both the training data and learning scheme(s) are used to build a learner. A learning scheme consists of a data preprocessing method, an attribute selection method, and a learning algorithm.

Evaluation of the proposed framework is comprised of:

1. A data preprocessor
   - The training data are preprocessed, such as removing outliers, missing values, and discretizing or transforming numeric attributes using NASA preprocessing tool.

2. An attribute selector
   - User can select all or any of the attributes provided by the NASA MDP Data Set.

3. Learning Algorithms
   - NaiveBayse Simple from bayse classification
   - Logistic classification
   - From Rule based classification
     - Decision Table
     - OneR
     - JRip
     - PART
   - From Tree based classification
     - J48
     - J48Graft

4.4 SCHEME EVALUATION ALGORITHM

Data: Historical Data Set
Result: The mean performance values

1 $M=12$: No of Data Set
2 $i=1$;
3 while $i<=M$ do
4 Read Historical Data Set $D(i)$;
5 Split Data set Instances using % split;
6 Train($i$)=60% of $D$; % Training Data;
7 Learning(Train($i$),scheme);
8 Test Data=$D(i)$-Train($i$);% Test Data;
9 Result=TestClassifier(Test($i$),Learner);
10 end

*Algorithm 4.1 Scheme Evaluation*
4.5 DEFECT PREDICTION

The defect prediction part of our framework is straightforward; it consists of predictor construction and defect prediction. During the period of the predictor construction:

i) A learning scheme is chosen according to the performance report.

ii) A predictor is built with the selected learning scheme and the whole historical data. While evaluating a learning scheme, a learner is built with the data and tested on the test data. Its final performance is the mean over rounds. This reveals that the evaluation indeed covers all the data. Therefore as we use all of the historical data to build the predictor, it is expected that constructed predictor has stronger generalization ability.

iii) After the predictor is built, new data are preprocessed in the same way as historical data, then the constructed predictor can be used to predict software defect with preprocessed new data.

4.6 DIFFERENCE BETWEEN PROPOSED FRAMEWORK AND OTHERS

Thus, to summarize, the main difference between our framework and that of others in the following:

i) We choose the entire learning scheme, not just one out of the learning algorithm, attribute selector, or data preprocessor

ii) We use the appropriate data to evaluate the performance of a scheme NASA MDP Data Set by Martin Shepperd (2013)

iii) We choose percentage split for training data set (60%) and test dataset (40%)

Data Set

We used the data taken from the public NASA MDP repository, which was also used by MGF and many others, e.g., Stefan Lessmann (2008). Thus, there are 12 data sets in total from NASA MDP repository. Table 4.1 and Table 4.2 provides some basic summary information. Each data set is comprised of a number of software modules (cases), each containing the corresponding number of defects and various software static code attributes. After preprocessing, modules that contain one or more defects were labeled as defective. A very detailed description of code attributes or the origin of the MDP data sets can be obtained from Tim Menzies (2007).
### Table 4.1 NASA MDP Data Sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>System</th>
<th>Language</th>
<th>Total Loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMI-5</td>
<td>Spacecraft Instrument</td>
<td>C</td>
<td>17K</td>
</tr>
<tr>
<td>KC3-4</td>
<td>Storage management for ground data</td>
<td>Java</td>
<td>8K and 25K</td>
</tr>
<tr>
<td>KC1-2</td>
<td>Storage management for ground data</td>
<td>C++</td>
<td>*</td>
</tr>
<tr>
<td>MW1</td>
<td>Database</td>
<td>C</td>
<td>8K</td>
</tr>
<tr>
<td>PC1,2,5</td>
<td>Flight Software for Earth orbiting Software</td>
<td>C</td>
<td>26K</td>
</tr>
<tr>
<td>PC3,4</td>
<td>Flight Software for Earth orbiting Software</td>
<td>C</td>
<td>30-36K</td>
</tr>
</tbody>
</table>

### Table 4.2 Data Sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Attribute</th>
<th>Module</th>
<th>Defect</th>
<th>Defect(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM1</td>
<td>38</td>
<td>344</td>
<td>42</td>
<td>1.22</td>
</tr>
<tr>
<td>JM1</td>
<td>22</td>
<td>9593</td>
<td>1759</td>
<td>18.34</td>
</tr>
<tr>
<td>KC1</td>
<td>22</td>
<td>2096</td>
<td>325</td>
<td>15.5</td>
</tr>
<tr>
<td>KC3</td>
<td>40</td>
<td>200</td>
<td>36</td>
<td>18</td>
</tr>
<tr>
<td>MC1</td>
<td>39</td>
<td>9277</td>
<td>68</td>
<td>0.73</td>
</tr>
<tr>
<td>MC2</td>
<td>40</td>
<td>127</td>
<td>44</td>
<td>34.65</td>
</tr>
<tr>
<td>MW1</td>
<td>38</td>
<td>264</td>
<td>27</td>
<td>10.23</td>
</tr>
<tr>
<td>PC1</td>
<td>38</td>
<td>759</td>
<td>61</td>
<td>8.04</td>
</tr>
<tr>
<td>PC2</td>
<td>37</td>
<td>1585</td>
<td>16</td>
<td>1.0</td>
</tr>
<tr>
<td>PC3</td>
<td>38</td>
<td>1125</td>
<td>140</td>
<td>12.4</td>
</tr>
<tr>
<td>PC4</td>
<td>38</td>
<td>1399</td>
<td>178</td>
<td>12.72</td>
</tr>
<tr>
<td>PC5</td>
<td>39</td>
<td>17001</td>
<td>503</td>
<td>2.96</td>
</tr>
</tbody>
</table>

### 4.7 Performance Measurement

The Performance measured according to the Confusion matrix given in Table 4.3, which is used by many researchers. Table 4.3 illustrates a confusion matrix for a two class problem having positive and negative class values.

#### Table 4.3 Confusion Matrix

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>True Positive</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Negative</td>
<td>False Positive</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

Software defect predictor performance of the proposed scheme based on Accuracy, Sensitivity, Specificity, Balance, and ROC Area defined as

\[
Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \\
= \frac{TRUE\ POSITIVE + TRUE\ NEGATIVE}{TRUE\ POSITIVE + FALSE\ POSITIVE + TRUE\ NEGATIVE + FALSE\ NEGATIVE} \\
= \text{The percentage of prediction that is correct}
\]

\[pd = True\ positive\]
\[Rate(tpr) = Sensitivity = \frac{TP}{TP + FN}\]

= The Percentage Of positive labeled instance that predicted as POSITIVE

\[Specify = \frac{TN}{FP + TN}\]

= The Percentage Of positive labeled instance that predicted as negative

\[Pf = False Positive Rate(fpr) = 1 - specificity\]

= The percentage of Negative labeled instances that predicted as negative

Formal definitions for pd and pf are given in the formula. Obviously, higher pds and lower pfs are desired. The point (pd=1, pf=0) is the ideal position where we recognize all defective modules and never make mistakes.

MGF introduced a performance measure called balance, which is used to choose the optimal (pd, pf) pairs. The definition is shown below from which we can see that it is equivalent to the normalized Euclidean distance from the desired Point (0, 1) to (pf,pd) in a ROC curve.

\[
\text{BALANCE} = 1 - \frac{\sqrt{(1 - pd)^2 + (0 - pf)^2}}{\sqrt{2}}
\]

The receiver operating characteristic (ROC) Charles E Metz (1998), curve is often used to evaluate the performance of binary predictors. A typical ROC curve is shown in Figure 4.2. The y-axis shows probability of detection (pd) and the x-axis shows probability of false alarms (pf).

Formal definitions for pd and pf are given above. Obviously, higher pds and lower pfs are desired. The point (pf=0, pd=1) is the ideal position where we recognize all defective modules and never make mistakes.

The Area under ROC Curve (AUC) is often calculated to compare different ROC curves. Higher AUC values indicate the classifier is an average, more to the upper left region of the graph. AUC represents the most informative and commonly used, thus it is used as another performance measure for our work.
Figure 4.2 Scheme evaluation of the proposed framework

4.8. RESULTS

This section provides simulation results of some of the Classification algorithm techniques collected by simulation on Software tool named Weka (version 3.6.9) with java embedded program. In the thesis, however, proposed schemes are more comprehensively compared with competent schemes. According to best accuracy value we choose 8 classification algorithms among many classification algorithms. All the evaluated values are collected and compare with different performance measurement parameter.

Accuracy

From the accuracy Table 4.4 we can see different algorithms giving different accuracy on different data set. But the average performance nearly same. For Storage management software (KC1-3) LOG, J48G giving better Accuracy value. For database software written in JAVA programming language (MW1) only PART giving better accuracy value. The performance graph is given in the Figure 4.4.

Sensitivity from the Table 4.5 we see that NB algorithm gives better performance in maximum data set. In case of Decision Table gives the sensitivity zero (sometimes) that means it considering all the class as a true negative. It cannot be considered for defect prediction. LOG, OneR, PART, J48, J48G algorithms giving average performance.