PROMISE 2010

Usage of Multiple Prediction Models Based On Defect Categories

Bora Caglayan[1], Ayse Tosun[2], Andriy Miranskyy[3], Ayse Bener[4], Nuzio Ruffolo[5]
Boğaziçi University[1,2], IBM Canada Ltd.[3,5], Ryerson University[4]
Department of Computer Engineering[1,2],
Ted Rogers School of Information Technology Management[4]
Istanbul, Turkey[1,2] Toronto, Canada[3,4,5]
{bora.caglayan[1], ayse.tosun[2]}@ boun.edu.tr, {andriy[3], ruffolo[5]}@ ca.ibm.com,
{ayse.bener[4]}@ ryerson.edu

OUTLINE

- Introduction and Motivation
- Research Question & Contributions
- Dataset and Data Extraction
- Methodology
- Results
- Discussion
- Conclusion and Future Work

INTRODUCTION

• **Classic approach in defect prediction**: Binary classification

• **Idea:** Different categories of defects may result from different patterns. Predicting defects by category may have practical benefits for effort planners.

•**Approach:** Select from different prediction models for different classes of defects

Research Question

- How can we increase the information content of defect predictor outcomes?

Contributions Of The Work

1. Mine the data repository of a large-scale enterprise software product and extract churn metrics and defect types

2. Analyze relations between metrics and defect categories

3. Build a general defect prediction model

4. Build a category-based defect prediction model and combine category-based defect prediction models to compare with the general defect prediction model

5. Try the methodology with another categorization method

Dataset

- A module of architectural functionality in a large-scale enterprise product.
  - 20 years old code base
  - Language: C/C++
  - Average File Size: ~3 kLOC
  - Number of Methods: 7742
  - Snapshot Date: 10 months before release
  - ~500 kLOC

Dataset

Metrics used

- Static Code Metrics
  - McCabe
  - Halstead
  - LOC
- Churn Metrics

Attribute	Description	Attribute	Description
McCabe metrics			
Cyclomatic density, vd(G)	the ratio of module's cyclomatic complexity to its length	*Essential complexity, ev(G)*	the degree to which a module contains unstructured constructs
Design density,dd(G)	condition/ decision	*Cyclomatic complexity, v(G)*	# linearly independent paths
Essential density,ed(G)	(ev(G)-1)/(v(G)-1)	*Maintenance severity*	ev(G)/v(G)
Decision count	# of decision points	*Condition count*	# of conditionals
Branch count	#of branches		
Halstead metrics			
Unique operands	n1	*Total operators*	N1
Total operands	N2	*Unique operators*	n2
Difficulty (D)	1/L	*Length (N)*	N1 + N2
Level (L)	(2/n1)*(n2/N2)	*Programming effort (E)*	D*V
Volume (V)	N*log(n)	*Programming time (T)*	E/18
Lines of code metrics			
Executable LOC	Source lines of code that contain only code and white space	*Lines of Comment*	lines of comments
Blank Lines	Blank Lines		
Churn metrics			
Number of edits	number of edits done on a file	*Lines Added*	total number of lines added
Number of unique committers	number of unique committers edited a file	*Lines Removed*	total number of lines removed

Dataset - Defect Types

FT Defects	ST Defects	Field Defects
Defects that are associated with the bugs found during function test	Defects that are associated with the bugs found during system test	Defects that are associated with the bugs found in the field (by the customers)

Defect Type	FT	ST	Field
ALL	0.604	0.466	0.617
FT	1	0.247	0.477
ST		1	0.466
Field			1

Methods	Count	Percentage
Total Methods	7742	100%
Methods with any Defects	2006	26%
Methods with FT Defects	337	4%
Methods with ST Defects	269	3%
Methods with Field Defects	445	5%

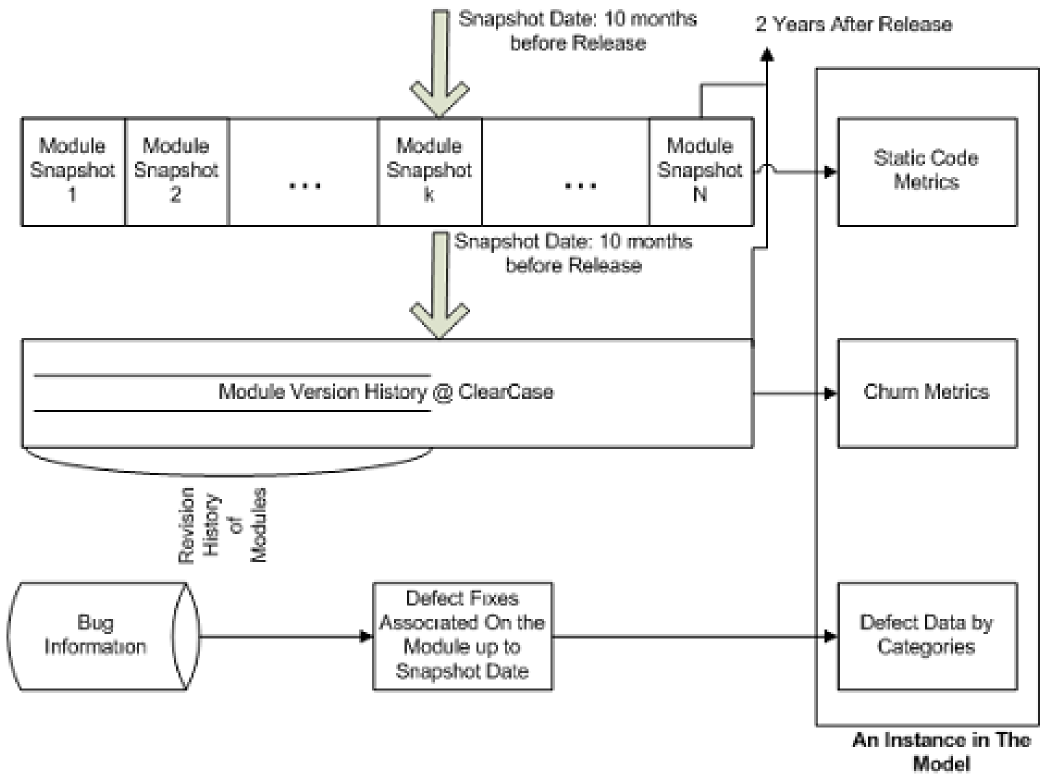
Dataset – For Replication

- Eclipse Dataset (Available on Promise Repository)
- Versions 2.0, 2.1, 3.0

Defect Types In Eclipse

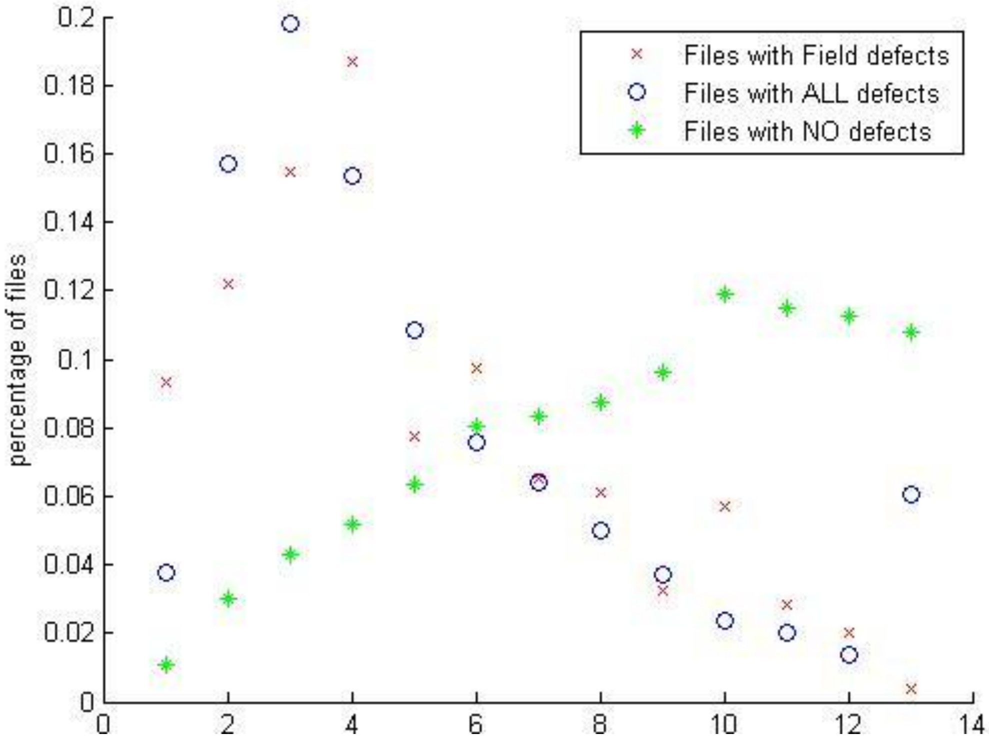
- Pre-Release   (spans 6 months before a release)
- Post-Release (spans 6 months after a release)

DATA EXTRACTION & DEFECT MATCHING



Distribution of Metrics According to Defect Types

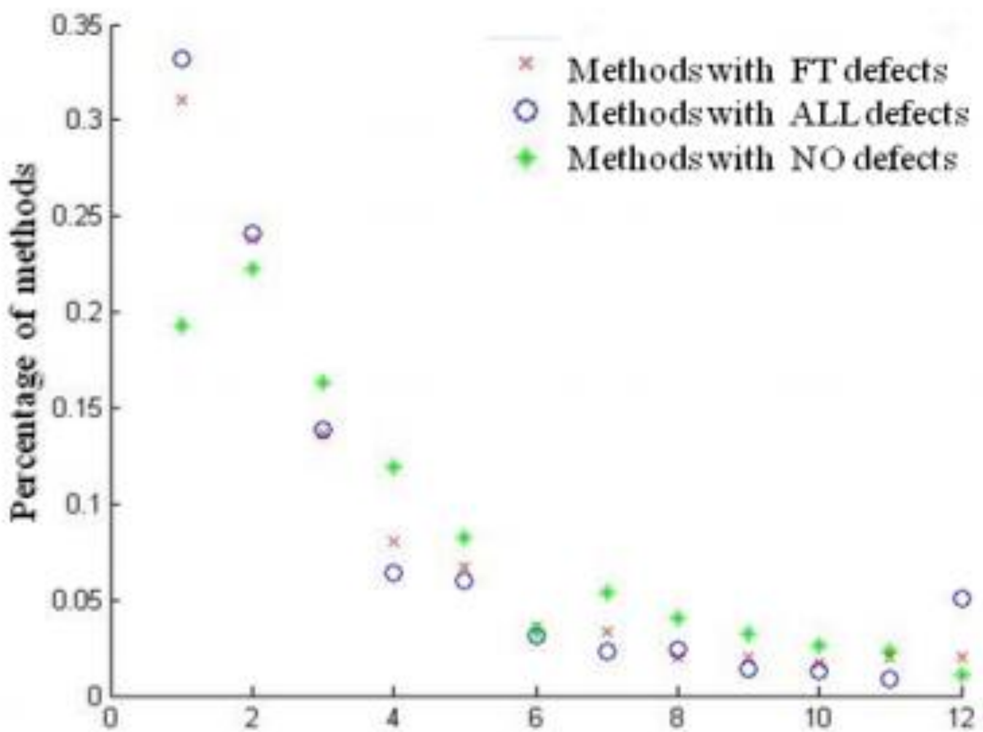
Certain attributes follow similar trends for all methods, but methods with Field defects, methods with all defects and methods with no defects have different medians.



Total operands

Distribution of Metrics According to Defect Types

Certain attributes follow almost the same trends for all methods.



Cyclometric Complexity

Top 10 Metrics For Different Defect Types

FVT Defects

0.04419 edits
0.04342 removed_line
0.03476 Comment LOC
0.03354 added_line
0.02903 Total LOC
0.02564 executable_loc
0.02523 Branch_count
0.02523 decision_count
0.0252 condition_count
0.02423 cyclomatic_complexity

SVT Defects

(1) 0.02797 edits
(2) 0.02619 all_churn
(3) 0.02594 removed_line
(4) 0.02281 added_line
(5) 0.01913 unique_operands
(6) 0.01871 halstead_vocabulary
(7) 0.01712 Total LOC
(8) 0.01681 Comment LOC
(9) 0.01639 executable_loc
(10) 0.01599 halstead_volume

Field Defects

(1) 0.0659 all_churn
(2) 0.06544 edits
(3) 0.06151 added_line
(4) 0.05865 removed_line
(5) 0.03051 halstead_vocabulary
(6) 0.03039 unique_operands
(7) 0.03019 halstead_length
(8) 0.03001 halstead_volume
(9) 0.02996 Halstead Error
(10) 0.02985 total_operands

- *Calculated with infogain algorithm.*
- *LOC & CC is more important for FVT or SVT than field defects. Churn is important in all defects.*

METHODOLOGY – PREDICTION MODEL

- Comparison of Combination of Various Methods

Pre-Processing
- Normalize
- Oversample
- Undersample
- None

Feature Selection
- PCA
- Info gain
- None

Algorithms
- BayesNet
- Naïve Bayes
- Logistic Regression

4 X 3 X 3 = 36 Combinations

METHODOLOGY – PREDICTION MODEL

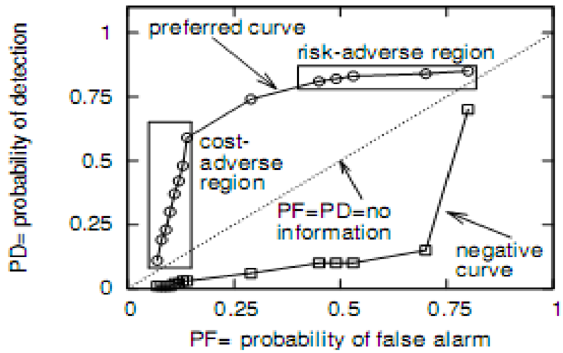
Pseudo code

$M = 10$
$N = 10$
$Datasets = \{ALLDefects, FTDefects, STDefects, FieldDefects\}$
$Algorithms = \{BayesNet, NaiveBayes, LogisticRegression\}$
$FeatureSelection = \{PCA, InfoGain, Allmetrics\}$
$PreProcessing = \{Normalize, Logfilter, Oversample, Undersample\}$
for each $data \in Datasets$
do
for each $feature \in FeatureSelection$
do
for each $filter \in PreProcessing$
do
for each $algorithm \in Algorithms$
do
$repeatMtimes$
$data' \longleftarrow randomizetheorderofdata$
$data'' \longleftarrow generateNbinsfromdata'$
for $i \longleftarrow 1to10$
do
$testset = data''[i]$
$trainset = data'' - data''[i]$
$predictions[i, ,] \longleftarrow applyfeature, filterandalgorithmtotrainingandtestsets$

METHODOLOGY – PERFORMANCE MEASURES

Defects		Actual	
		no	yes
Prd	no	TN	FN
	yes	FP	TP

pd= TP / (FN+TP)
pf = FP / (TN+FP)



	precision	pd	pf	effort#
Early in a contractor/client relationship	hi			
Risk adverse (e.g. airport bomb detection, morning sickness)		hi	hi	
Cost adverse (e.g. budget conscious)		med	lo	
Arisholm and Briand [2006]				$< pd$

#effort = LOC in the modules predicted to be faulty

RESULTS

Bayes Net with no feature selection gave the best results for all defect types

	Algorithm: Bayes Net, 10-Fold Cross Validation			
	No Resampling		With Resampling (Over or undersampling)	
Prediction Type	**pd**	**pf**	**pd**	**Pf**
General Defect Prediction	0.64	0.27	0.62	0.25
FVT Defect Prediction	0.64	0.17	0.69	0.17
SVT Defect Prediction	0.54	0.21	0.66	0.25
Field Defect Prediction	0.43	0.06	0.67	0.25
Defect Category Sensitive Defect Prediction [Field or FVT or SVT]			**0.71**	**0.28**

RESULTS – Comparison of Generic Model with Defect Category Sensitive Model

General Defect Predictor
Pd = 0.64
Pf = 0.27

vs.

FT Defect Predictor
Pd = 0.69
Pf = 0.17

<or>

ST Defect Predictor
Pd = 0.66
Pf = 0.25

<or>

FIELD Defect Predictor
Pd = 0.67
Pf = 0.25

Defect Category Sensitive Defect Predictor
Pd = 0.71
Pf = 0.28

pd **significantly higher** with $p < 0.05$ pf difference is **insignificant** in defect category sensitive defect prediction.

RESULTS – Replication with Eclipse

Defect Category	pd	pf
All Defects	0.75	0.38
Pre-Release Defects	0.81	0.46
Post-Release Defects	0.67	0.32
Combination of Defect Categories	0.76	*0.38*

Defect Category	pd	pf
All Defects	0.65	0.27
Pre-Release Defects	0.65	0.26
Post-Release Defects	0.65	0.27
Combination of Defect Categories	0.65	0.26

The increase in prediction success is not significant in terms of pd and pf (Mann-Whitney U Test with $p < 0.05$).

THREATS to VALIDITY

Construct

•While labeling defect categories, we used the descriptions of testing phases and double-checked these labels to overcome any problems.

Internal

•In order to avoid sampling bias in our experiments, we used 10-fold cross validation.

External

•For external validation we used Eclipse dataset for conceptual replication of our experiments.

CONCLUSIONS

- **RQ :** How can we increase the information content of defect predictor outcomes?
  - We can use category information to build a defect prediction model which can also increase prediction rates by 10% in our dataset.

Theoretical Contributions

- Our work shows that category based predictors can give better results in terms of both information content of prediction and prediction performance

Practical Contributions

- In addition defect category sensitive model can predict the type of defect which can be beneficial to effort planners.

Future Work

- We will do research on systematic categorization of defects.

Thank you for your attention…

Question/Comments

bora.caglayan@boun.edu.tr