Developing Accurate Software Quality Models Using a Faster, Easier , and Cheaper Method

Taghi M. Khoshgoftaar
Linda Lim
Erik Geleyn

Empirical Software Engineering Laboratory
Dept. of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
(561)297-3994
taghi@cse.fau.edu
http://www.cse.fau.edu/esel.html

Overview

Introduction

Case Based Reasoning

Case Study Methodology

Empirical Case Study Experiments

Conclusions

Introduction

Models and measurements are the means for understanding, controlling, and improving development processes.

Various classification and prediction models are available.

Case based reasoning (CBR) is an effective technique.

Our empirical work demonstrates that substantial benefits can be achieved using a simple prediction model such as CBR.

Case Based Reasoning

Concepts

- Based on automated reasoning processes.
- Easy to use and results are easy to understand and interpret.
- Looks at past cases that are similar to the present case in an attempt to predict or classify the specific attributes desired.

Case Based Reasoning

Working hypothesis

Current cases that are in development will more than likely also be fault-prone if past cases having similar attributes were fault-prone.

Case Based Reasoning

Additional advantages

- The ability to alert users when a new case is outside the bounds of current experience.

- The ability to interpret the automated classification through the detailed description of the most similar case.

- The ability to take advantage of new or revised information as it becomes available.

- The ability for fast retrieval as the size of the library scales up.

Case Based Reasoning

Fit and Test Data Sets

- Past cases contained in a library. Each case contains all the known attributes, independent and dependent variables.

- This set of past cases is collectively known as the fit data set.

- The target (test) data set containing the present cases are usually from a current project.

- In the target (test) data set, only the independent variables are known.

- By applying the models built using the fit data set to the target data set, we can determine the dependent variables for the present cases.

Case Based Reasoning

Similarity Functions

Calculates the distance, d_{ij}, from the current case, x_i, to each of the cases in the library, c_j.

Several type of similarity functions are available:

- Absolute Distance
- Euclidean Distance
- Mahalonobis Distance

Case Based Reasoning

Mahalonobis Distance

- Alternative to Euclidean Distance
- Explicitly accounts for correlation among attributes
- Independent variables do not need to be standardized

$$d_{ij} = (c_j - x_i)' S^{-1} (c_j - x_i) \qquad (1)$$

Prime (') means transpose, and S is the variance-covariance matrix of the independent variables over the entire case library. S^{-1} is its inverse.

Case Based Reasoning

Solution Algorithm

A solution algorithm is used to estimate the actual value of the dependent variable.

Two solution algorithms are available:

- Unweighted Average

- Inverse-Distance Weighted Average

Case Based Reasoning

Inverse-Distance Weighted Average

Distance between the current case and the closest cases in the library are weights in a weighted average.

$$\delta_{ij} = \frac{1/d_{ij}}{\sum_{j \in N} 1/d_{ij}} \tag{2}$$

$$\widehat{y}_i = \sum_{j \in N} \delta_{ij} y_i \tag{3}$$

Case Based Reasoning

Classification Methods

Used to classify a dependent variable into a particular class (fault-prone, not fault-prone).

The types of classification methods include:

- Data Clustering

- Majority Voting

Case Based Reasoning

Data Clustering

The library is partitioned into clusters according to the actual class of each case. Distances to the clusters are computed for the current case.

$$Class(x_i) = \begin{cases} NFP, & \text{if } \frac{d_{fp}(x_i)}{d_{nfp}(x_i)} > \frac{C_I}{C_{II}} \\ FP, & \text{Otherwise} \end{cases} \quad (4)$$

Case Based Reasoning

Model evaluation

- Prediction: We use Average Absolute Error (AAE) and Average Relative Error (ARE):

$$AAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i| \quad (5)$$

$$ARE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \widehat{y}_i}{y_i + 1}\right| \quad (6)$$

- Classification: The misclassification rates are determined by assessing the actual classifications. A type I misclassification predicts a not fault-prone to be fault-prone while a type II misclassification predicts a fault-prone module to be not fault-prone.

Case Study Methodology

Definitions

Case study: Research technique where we identify key factors that may affect the outcome of an activity and document the activity.

Projects for a case study should:

1. Be developed by a group, rather than an individual programmer.

2. Be developed by professionals, rather than students.

3. Be developed in an industrial environment rather than an artificial setting.

4. Be large enough to be comparable to real industry projects.

Case Study Methodology

Building the models

---

1. Preprocess measurements to improve the model.

2. Choose a model validation strategy, such as re-substitution, data splitting, cross-validation, multiple releases, or multiple projects. We used cross-validation and data splitting.

3. Prepare *fit* and *test* data sets.

Case Study Methodology

Building the models

---

4. Select significant independent variables from a set of candidates, based on the *fit* data set.

5. Estimate parameters (similarity function, number of nearest neighbors, solution algorithm) of the model on the selected independent variables, based on the *fit* data set.

6. Use the model to predict the quality factor or the class of each module in the *test* data set, and compare predictions to actual outcome for the *test* data set.

Case Study Methodology

Validation Strategy

- Cross-validation: Let one observation be the target data set and all others be the fit data set. Build a model and evaluate it for the current observation or target data set. Repeat for each observation, resulting in n models. Let the misclassification rates summarize the n evaluations of the models.

- Data Splitting: Derives a fit data set and a target (test) set by randomly sampling from the cases available and impartially partitioning them into two data sets.

Case Study Methodology

Empirical Modeling Tool

The Software Measurement Analysis and Reliability Toolkit (SMART):

- Developed at the Empirical Software Engineering Lab (ESEL)

- Microsoft Visual C++, and runs on Windows 95 or Windows NT

- Graphical User Interface to allow users to select model parameters such as similarity functions, solution algorithms, number of cases, etc

Empirical Case Study Experiments

System description

The two data sets were obtained from the industry. The project consisted of two large Windows-based applications used primarily for customizing the configuration of wireless products. The data sets were obtained from the initial release of these applications. The applications are written in C++, and they provide similar functionality.

Empirical Case Study Experiments

System description

Applications	Service Configuration Software
Language	C++
Application 1: AENSCL*	320 million
Application 1: Actual Lines of Code	29 million
Application 2: AENSCL*	300 million
Application 2: Actual Lines of Code	27.5 million
Number of source files	1400

* AENCSL is Assembly Equivalent Non-Commented Source Lines of Code

Empirical Case Study Experiments

Software Metrics

Symbol	Description
Product Metrics, Statement metrics	
BASE_LOC	Number of lines of code for the source file version just prior to the coding phase. This represents auto-generated code.
SYST_LOC	Number of lines of code for the source file version delivered to system test.
BASE_COM	Number of lines of commented code for the source file version just prior to the coding phase. This represents auto-generated code.
SYST_COM	Number of lines of commented code for the source file version delivered to system test.
Process Metrics	
INSP	Number of times the source file was inspected prior to system test.

Empirical Case Study Experiments

Experiments

We conducted 6 classification experiments using the following classification rule:

$$Class = \begin{cases} FP, & \text{if } y \leq Threshold \\ NFP, & \text{Otherwise} \end{cases} \tag{7}$$

- Experiments 1-3 used the entire data set as both the fit and target data set. For thresholds 1,2 and 3.

- Experiments 4-6 used 2/3 of data as fit data set and 1/3 as target data set. For thresholds 1,2 and 3.

Empirical Case Study Experiments

Classification Experiments

- *Threshold 1*: a file is fault-prone if it contains one or more faults- $y \geq 1$

  In this case 402 files (33%) are actually considered as fault-prone and 809 (67%) as not fault-prone.

- *Threshold 2*: a file is fault-prone if it contains two or more faults- $y \geq 2$

  262 files (22%) are actually considered as fault-prone and 949 (78%) as not fault-prone.

- *Threshold 3*: a file is fault-prone if it contains three or more faults- $y \geq 3$

  200 files (17%) are actually considered as fault-prone and 1011 (83%) as not fault-prone.

Empirical Case Study Experiments

Classification Experiments 1-3

We used cross-validation to build and validate the model. The two parameters in our classification models are n_N, and the cost ratio. Our objective was to obtain balanced misclassification rates with type II misclassification as low as possible.

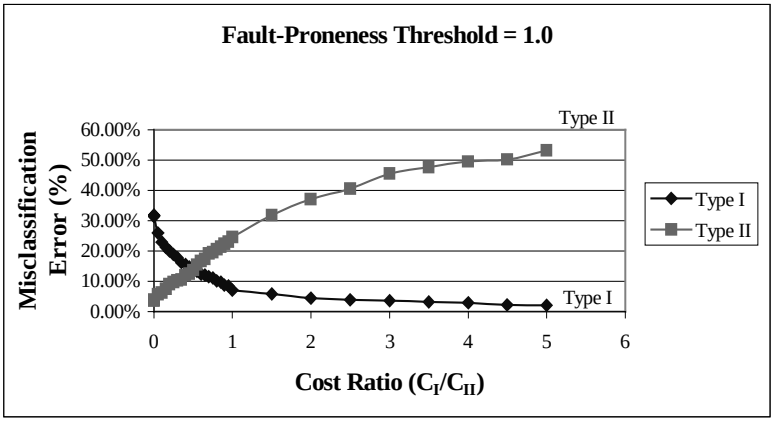
Empirical Case Study Experiments

Classification Experiment 1, Cost Ratios

Cost ratio CI/CII	Fit(Cross - Validation	
	Type I	Type II
0.0005	31.77%	3.73%
0.001	31.77%	3.73%
0.0015	31.64%	3.73%
0.002	31.64%	3.73%
0.0025	31.40%	3.73%
0.05	25.96%	5.72%
0.1	22.99%	6.22%
0.15	21.26%	7.46%
0.2	20.03%	8.96%
0.25	18.91%	9.70%
0.3	17.80%	10.20%
0.35	16.19%	10.45%
0.4	15.58%	11.94%
0.45	14.83%	12.44%
0.5	**14.34%**	**13.68%**
0.55	13.23%	15.42%
0.6	12.36%	16.67%
0.65	11.99%	17.41%
0.7	11.50%	19.15%
0.75	11.13%	19.65%
0.8	10.14%	20.65%
0.85	9.64%	21.39%
0.9	8.78%	22.39%
0.95	8.41%	23.13%
1	7.05%	24.63%
1.5	5.81%	31.84%
2	4.45%	37.07%
2.5	3.83%	40.55
3.00	3.59%	45.52%
3.50	3.21%	47.51%
4.00	2.97%	49.50%
4.50	2.23%	50.25%
5.00	2.10%	53.23%

Empirical Case Study Experiments

Classification Experiment 1, Cost Ratios



Empirical Case Study Experiments

Return on Investment (ROI) for Experiment 1

- The best cost ratio determined empirically is 0.50.
- Type I cost = 1 unit.
- Type II cost = (1/0.50) 2 units.
- Actual Type II cost = 3 units (Cost of forfeited benefit + reviews).

Empirical Case Study Experiments

Return on Investment (ROI) for Experiment 1

- Cost of reviews would be 463 units (809*0.1434*1) + (402*(1-0.1368)*1).
- Reliability improvement for 347 (402*0.8632) fault-prone files.
- We avoided 1041 units (347*3) in debugging costs later.
- ROI 1041:463 (We invested 463 units of effort that yielded to 1041 units saved).

Empirical Case Study Experiments

Evaluation of the Classification Results for Experiment 1-3

T	Type I	Type 2	ROI Debug cost	Prior (%) Fault-prone	Reduction (%) Fault-prone
1	14.34%	13.68%	1041:463	33.20%	4.54%
2	13.91%	14.50%	721:356	21.64%	3.13%
3	14.74%	14.00%	602:321	16.52%	2.13%

ROI can be much greater than what our model results indicate. Since it only accounts for the direct saving allowed by the early prediction of the fault-prone modules.

Empirical Case Study Experiments

Classification Experiments 4-6

We used cross-validation using the fit data set and data splitting using the target data set.

- Fit data set: 807 observations.
- Target data set: 404 observations.
- We performed 50 splits for accurate model results.
- Same three fault-proneness thresholds were used.

Empirical Case Study Experiments

All Classification 4-6, Results

T	Cross-Validation Fit/Target		Data Splitting Fit/Target		Average	
	Type I	Type II	Type I	Type II	n_N	CI/CII
1	15.91%	15.55%	15.75%	16.37%	2.28	0.57
2	15.09%	14.71%	15.08%	16.19%	2.10	0.40
3	15.65%	15.28%	15.08%	16.55%	3.16	0.37

Conclusions

Lessons Learned

Our research gave evidence of the following results:

- As the delivery date for a product quickly approaches, time becomes an increasingly precious commodity.

- The use of the software quality models built can allow management to make more appropriate use of the precious time left.

- This will ensure that the established reliability and quality standards can still be achieved.

- We have proven that CBR is a simple modeling methodology that can be used to develop accurate and useful models faster, easier, and cheaper.

References