Using Source Code Metrics and Multivariate Adaptive Regression Splines to Predict Maintainability of Service Oriented Software

Lov Kumar\textsuperscript{1} Santanu Kumar Rath\textsuperscript{1} Ashish Sureka\textsuperscript{2}

\textsuperscript{1}NIT Rourkela, India  
Email: lovkumar505@gmail.com, skrath@nitrkl.ac.in

\textsuperscript{2}ABB India, India  
Email: ashish.sureka@in.abb.com

HASE 2017
Service-Oriented Computing (SOC) paradigm is the dominant development platform for software systems.

One most effective way to control the maintenance costs is to utilize software metrics during the development phase.

Web Services are observed to be described using a WSDL (Web Service Description Language) document.

Maintainability prediction for web service interfaces and WSDL documents is relatively unexplored.
Aim 1: To investigate the usefulness of metrics for the source code implementing a web service for maintainability prediction

Aim 2: To examine the application of multivariate adaptive regression splines (MARS) method to build estimators based on source code metrics as predictor variables and maintainability as target variable

Aim 3: To compare the performance of MARS model with performance of the model developed using multivariate linear regression models (MLR), and support vector machine (SVM) methods
Research Motivation and Aim

Related Work and Research Contributions

Experimental Dataset and Setup

Experimental Analysis and Results

Conclusion

References

Table of Contents

1. Research Motivation and Aim

2. Related Work and Research Contributions
   - Related Work
   - Research Contributions

3. Experimental Dataset and Setup
   - Data Collection
   - Dependent and Independent Variables

4. Experimental Analysis and Results
   - Framework
   - Feature Ranking Technique
   - Feature Subset Selection Methods
   - Accuracy, Precision and Recall

5. Conclusion

6. References

- Lov Kumar, Santanu Kumar Rath, Ashish Sureka
  - Using Source Code Metrics and Multivariate Adaptive Regression
Bingu Shim et al. [131]
Authors observed that quality attributes depend on various numbers of design properties, i.e., consumability, cohesion, coupling, complexity, size, different types of granularity. They derive metrics and compute the relation between design metrics and quality.

Cristian Mateos et al. [100]
Authors work on the hypothesis that object-oriented source code metrics avoid the occurrence of anti-pattern. They considered twelve different types of source code metrics such as: AMC, CAM, CBO, WMC, LOC, LCOM, LCOM3, RFC, EPM, VTC, ATC, DAM to find the anti-pattern occurrence at the WSDL level.
Literature Survey and Analysis

Mikhail Perepletchikov et al. [119][120]
Authors propose cohesion and coupling metrics for predicting maintainability parameters of Service-Oriented Software [119] [120]. Authors redefined the cohesion and coupling metrics for service-oriented software.

Yu, Yijun [148]
OO source code metrics are being used for quality indicators of SOA system [148] [100] [39]
# Table of Contents

1. Research Motivation and Aim

2. Related Work and Research Contributions
   - Related Work
   - Research Contributions

3. Experimental Dataset and Setup
   - Data Collection
   - Dependent and Independent Variables

4. Experimental Analysis and Results
   - Framework
   - Feature Ranking Technique
   - Feature Subset Selection Methods
   - Accuracy, Precision and Recall

5. Conclusion

6. References
Unique and Novel Contributions

[1] Development of Data Collection and Preparation Process

[2] Development of a maintainability prediction model for service oriented application using MARS and source code metrics

[3] Selection of right set of source code metrics to improve the performance of maintainability prediction model

[4] Empirical analysis to demonstrate the effectiveness of the proposed approach
Table of Contents

1. Research Motivation and Aim
2. Related Work and Research Contributions
   - Related Work
   - Research Contributions
3. Experimental Dataset and Setup
   - Data Collection
   - Dependent and Independent Variables
4. Experimental Analysis and Results
   - Framework
   - Feature Ranking Technique
   - Feature Subset Selection Methods
   - Accuracy, Precision and Recall
5. Conclusion
6. References
11 Different Source Code Metrics

1. Depth of inheritance tree (DIT)
2. Weighted method per class (WMC)
3. Coupling between object (CBO)
4. Number of children (NOC)
5. Number of methods (NOM)
6. Lack of cohesion among methods (LCOM)
7. Response for class (RFC)
Source Code Metrics

11 Different Source Code Metrics

8 Data abstraction coupling (DAC)
9 SIZE2
10 SIZE1
11 McCabe Cyclomatic complexity (MVG)

Source code metrics are computed using tool such as CKJM and LOC metrics. These metrics include the popular metrics suite proposed by Chidamber and Kemerer [35], and Li and Henry [92]
Data Collection and Preparation Process

1. **Ver. 1** and **Ver. 2** WSDL
2. **WSDLDiff tool**
3. **Change Elements**
4. **WSimport Tool**
5. **Java Files**
6. **Classes Identification**
7. **Changed Classes**
8. **Metrics Collection**
9. **Metrics**
10. **Data Set**
Data Collection and Preparation Process

**Step 1: WSDLDiff Tool**

WSDLDiff tool is used for comparing subsequent versions of WSDL interfaces ([38]) and finding the name (Operation, Message etc.) and type (addition, move, removed, update etc.) of the elements affected by the changes.

**Step 2: WSimport Tool**

Wsimport tool is used to parse WSDL document of web-service and generate Java files.
Data Collection and Preparation Process

**Step 3: Metrics Collection**

Metrics values of source code are computed using tools such as CKJM and LOC metrics.

**Step 4: Class Identification**

The Java classes which contain the changed elements (Operation, Message etc.) are termed as changed classes. These types of classes containing **changed elements** are identified.

**Step 5: Data Set**

The change statistics from Step 4 and metrics from Step 3 are combined to generate dataset for further processing.
The number of classes for all the five versions are more than 1500. The percentage of changed classes from eBay version 863 to eBay version 865 is 11.61 and the percentage of change classes from 865 to 867 is 6.77.
Table of Contents

1 Research Motivation and Aim
2 Related Work and Research Contributions
   - Related Work
   - Research Contributions
3 Experimental Dataset and Setup
   - Data Collection
   - Dependent and Independent Variables
4 Experimental Analysis and Results
   - Framework
   - Feature Ranking Technique
   - Feature Subset Selection Methods
   - Accuracy, Precision and Recall
5 Conclusion
6 References

Lov Kumar, Santanu Kumar Rath, Ashish Sureka
Using Source Code Metrics and Multivariate Adaptive Regression
## Model Variables

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Dependent Variable</th>
<th>Independent Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Maintainability</td>
<td>DIT, WMC, NOC, RFC, CBO, NOM, LCOM, MVG, DAC, SIZE1, SIZE2</td>
</tr>
<tr>
<td>A2</td>
<td>Maintainability</td>
<td>Reduced feature attributes using feature ranking techniques</td>
</tr>
<tr>
<td>A3</td>
<td>Maintainability</td>
<td>Extracted feature attributes using feature subset selection techniques</td>
</tr>
</tbody>
</table>

Maintainability or change is considered as a dependent variable and set of source code metrics as independent variables respectively.

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Using Source Code Metrics and Multivariate Adaptive Regression Splines to Predict Maintainability of Service Oriented Software
Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Using Source Code Metrics and Multivariate Adaptive Regression
Research Methodology

[Diagram of the research methodology showing the flow from data set, through feature ranking technique, to selected features and MLR, SVM, and MARS classifiers.]

The model is developed by considering the source code metrics as input, and maintainability as output.
Four types of FR techniques and four types of FSS techniques to select optimal set of source code metrics. These all selected set of source code metrics are applied on five different versions of eBay dataset.

A total of 135 ((1 considering all features + 8 feature selection technique) * 5 datasets * 3 different classification technique) distinct classification models are considered in the study.
Research Methodology

Step 1
Eleven different source code metrics from the bytecode of the compiled Java files of web service are computed.

Step 2
Four different FR techniques are applied on all five versions of eBay web service. Each FR technique uses some parameter to sort the features and further top \( \lceil \log_2 n \rceil \) ranked features out of \( n \) features are used as input to develop a model.
Research Methodology

Step 3
Four different FSS techniques have been considered to select optimal set of features

Step 4
All elven source code metrics, selected set of source code metrics using feature selection techniques are validated using three different classification methods i.e., MLR, SVM, and MARS
## Naming Conventions for different Techniques

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Corresponding Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>All Metrics</td>
</tr>
<tr>
<td>FR1</td>
<td>CS test</td>
</tr>
<tr>
<td>FR2</td>
<td>GR Feature Evaluation</td>
</tr>
<tr>
<td>FR3</td>
<td>IG Feature Evaluation</td>
</tr>
<tr>
<td>FR4</td>
<td>OneR Feature Evaluation</td>
</tr>
<tr>
<td>FS1</td>
<td>Classifier Subset Evaluation</td>
</tr>
<tr>
<td>FS2</td>
<td>Correlation based Feature Selection</td>
</tr>
<tr>
<td>FS3</td>
<td>Filtered Subset Evaluation</td>
</tr>
<tr>
<td>FS4</td>
<td>Rough Set Analysis (RSA)</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Research Motivation and Aim
2. Related Work and Research Contributions
   - Related Work
   - Research Contributions
3. Experimental Dataset and Setup
   - Data Collection
   - Dependent and Independent Variables
4. Experimental Analysis and Results
   - Framework
   - Feature Ranking Technique
   - Feature Subset Selection Methods
   - Accuracy, Precision and Recall
5. Conclusion
6. References
## Selected metrics after Feature ranking methods

<table>
<thead>
<tr>
<th>Method</th>
<th>863</th>
<th>865</th>
<th>867</th>
<th>869</th>
<th>871</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1</td>
<td>SIZE1, MVG, DAC, SIZE2</td>
<td>DAC, MVG, CBO, DAC, LCOM</td>
<td>SIZE1, CBO, DAC, MVG</td>
<td>DAC, LCOM, MVG, SIZE1</td>
<td>DAC, LCOM, WMC, CBO</td>
</tr>
<tr>
<td>FR2</td>
<td>DAC, WMC, LCOM, RFC</td>
<td>NOM, WMC, LOCM, DAC</td>
<td>NOM, DIT, MVG, SIZE2</td>
<td>NOM, DIT, RFC, WMC</td>
<td>LOCM, NOM, WMC, RFC</td>
</tr>
<tr>
<td>FR3</td>
<td>SIZE1, MVG, SIZE2, CBO</td>
<td>DAC, MVG, LCOM, WMC</td>
<td>SIZE1, RFC, MVG, CBO</td>
<td>DAC, LCOM, MVG, SIZE1</td>
<td>LOCM, WMC, DAC, MVG</td>
</tr>
<tr>
<td>FR4</td>
<td>DAC, MVG, NOC, NOM</td>
<td>DAC, MVG, NOC, NOM</td>
<td>MVG, DAC, NOM, LCOM</td>
<td>DAC, MVG, NOC, NOM</td>
<td>DAC, NOC, DIT, CBO</td>
</tr>
</tbody>
</table>
In case of **MLR**, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.

In case of **SVM**, model developed by considering selected set of source code metrics using correlation based feature selection as input produces better result as compared to others.

In case of **MARS**, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.
Table of Contents

1 Research Motivation and Aim

2 Related Work and Research Contributions
   • Related Work
   • Research Contributions

3 Experimental Dataset and Setup
   • Data Collection
   • Dependent and Independent Variables

4 Experimental Analysis and Results
   • Framework
   • Feature Ranking Technique
   • Feature Subset Selection Methods
   • Accuracy, Precision and Recall

5 Conclusion

6 References
### Selected metrics after Feature subset selection

<table>
<thead>
<tr>
<th>Method</th>
<th>FS1</th>
<th>FS2</th>
<th>FS3</th>
<th>FS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>WMC, DIT, DAC, MVG</td>
<td>WMC, DIT, DAC</td>
<td>DAC, DIT, WMC</td>
<td>DIT, MVG, DAC, NOM, RFC, SIZE2, SIZE1</td>
</tr>
<tr>
<td>863</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>865</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>867</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>869</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>871</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In case of **MLR**, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.

In case of **SVM**, model developed by considering selected set of source code metrics using correlation based feature selection as input produces better result as compared to others.

In case of **MARS**, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.
The hardware used to carry out our experiments are: Core i5 processor with 4GB RAM and storage capacity of 250GB hard disk.

Prediction models are developed using the licensed MATLAB environment at NIT-Rourkela.

Model developed for predicting web-service maintainability using MARS yields better result as compared to MLR, and SVM methods.
# Table of Contents

1. Research Motivation and Aim
2. Related Work and Research Contributions
   - Related Work
   - Research Contributions
3. Experimental Dataset and Setup
   - Data Collection
   - Dependent and Independent Variables
4. **Experimental Analysis and Results**
   - Framework
   - Feature Ranking Technique
   - Feature Subset Selection Methods
   - Accuracy, Precision and Recall
5. Conclusion
6. References

---

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Using Source Code Metrics and Multivariate Adaptive Regression Splines to Predict Maintainability of Service Oriented Software
Lov Kumar, Santanu Kumar Rath, Ashish Sureka
Using Source Code Metrics and Multivariate Adaptive Regression

Accuracy
Using Source Code Metrics and Multivariate Adaptive Regression Splines to Predict Maintainability of Service Oriented Software

Lov Kumar, Santanu Kumar Rath, Ashish Sureka
Lov Kumar, Santanu Kumar Rath, Ashish Sureka
Using Source Code Metrics and Multivariate Adaptive Regression Splines to Predict Maintainability of Service Oriented Software
In case of **MLR** method, model developed by considering selected set of source code metrics using oneR feature evaluation as input produces better result as compared to others.

In case of **SVM**, model developed by considering selected set of source code metrics using correlation based feature selection as input produces better result as compared to others.

In case of **MARS**, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.
### t-test: Among Different Feature Selection Techniques

<table>
<thead>
<tr>
<th></th>
<th>AM</th>
<th>FR1</th>
<th>FR2</th>
<th>FR3</th>
<th>FR4</th>
<th>FS1</th>
<th>FS2</th>
<th>FS3</th>
<th>FS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>0</td>
<td>-0.04</td>
<td>0.03</td>
<td>-0.04</td>
<td>-0.14</td>
<td>-0.09</td>
<td>-0.11</td>
<td>-0.09</td>
<td>-0.04</td>
</tr>
<tr>
<td>FR1</td>
<td>0.04</td>
<td>0</td>
<td>0.07</td>
<td>0</td>
<td>-0.1</td>
<td>-0.05</td>
<td>-0.07</td>
<td>-0.05</td>
<td>-0.01</td>
</tr>
<tr>
<td>FR2</td>
<td>-0.03</td>
<td>-0.07</td>
<td>0</td>
<td>-0.07</td>
<td>-0.16</td>
<td>-0.12</td>
<td>-0.13</td>
<td>-0.12</td>
<td>-0.07</td>
</tr>
<tr>
<td>FR3</td>
<td>0.04</td>
<td>0</td>
<td>0.07</td>
<td>0</td>
<td>-0.1</td>
<td>-0.05</td>
<td>-0.07</td>
<td>-0.05</td>
<td>0</td>
</tr>
<tr>
<td>FR4</td>
<td>0.14</td>
<td>0.1</td>
<td>0.16</td>
<td>0.1</td>
<td>0</td>
<td>0.04</td>
<td>0.03</td>
<td>0.05</td>
<td>0.09</td>
</tr>
<tr>
<td>FS1</td>
<td>0.09</td>
<td>0.05</td>
<td>0.12</td>
<td>0.05</td>
<td>-0.04</td>
<td>0</td>
<td>-0.02</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>FS2</td>
<td>0.11</td>
<td>0.07</td>
<td>0.13</td>
<td>0.07</td>
<td>-0.03</td>
<td>0.02</td>
<td>0</td>
<td>0.02</td>
<td>0.06</td>
</tr>
<tr>
<td>FS3</td>
<td>0.09</td>
<td>0.05</td>
<td>0.12</td>
<td>0.05</td>
<td>-0.05</td>
<td>0</td>
<td>-0.02</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>FS4</td>
<td>0.04</td>
<td>0.01</td>
<td>0.07</td>
<td>0</td>
<td>-0.09</td>
<td>-0.05</td>
<td>-0.06</td>
<td>-0.05</td>
<td>0</td>
</tr>
</tbody>
</table>

### p-value

<table>
<thead>
<tr>
<th></th>
<th>AM</th>
<th>FR1</th>
<th>FR2</th>
<th>FR3</th>
<th>FR4</th>
<th>FS1</th>
<th>FS2</th>
<th>FS3</th>
<th>FS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>-</td>
<td>0.64</td>
<td>0.76</td>
<td>0.5</td>
<td>0.1</td>
<td>0.37</td>
<td>0.2</td>
<td>0.3</td>
<td>0.49</td>
</tr>
<tr>
<td>FR1</td>
<td>0.64</td>
<td>-</td>
<td>0.22</td>
<td>0.95</td>
<td>0.19</td>
<td>0.57</td>
<td>0.4</td>
<td>0.53</td>
<td>0.92</td>
</tr>
<tr>
<td>FR2</td>
<td>0.76</td>
<td>0.22</td>
<td>-</td>
<td>0.33</td>
<td>0.06</td>
<td>0.26</td>
<td>0.14</td>
<td>0.21</td>
<td>0.34</td>
</tr>
<tr>
<td>FR3</td>
<td>0.5</td>
<td>0.95</td>
<td>0.33</td>
<td>-</td>
<td>0.11</td>
<td>0.52</td>
<td>0.28</td>
<td>0.46</td>
<td>0.92</td>
</tr>
<tr>
<td>FR4</td>
<td>0.1</td>
<td>0.19</td>
<td>0.06</td>
<td>0.11</td>
<td>-</td>
<td>0.46</td>
<td>0.35</td>
<td>0.16</td>
<td>0.15</td>
</tr>
<tr>
<td>FS1</td>
<td>0.37</td>
<td>0.57</td>
<td>0.26</td>
<td>0.52</td>
<td>0.46</td>
<td>-</td>
<td>0.82</td>
<td>0.96</td>
<td>0.58</td>
</tr>
<tr>
<td>FS2</td>
<td>0.2</td>
<td>0.4</td>
<td>0.14</td>
<td>0.28</td>
<td>0.35</td>
<td>0.82</td>
<td>-</td>
<td>0.62</td>
<td>0.27</td>
</tr>
<tr>
<td>FS3</td>
<td>0.3</td>
<td>0.53</td>
<td>0.21</td>
<td>0.46</td>
<td>0.16</td>
<td>0.96</td>
<td>0.62</td>
<td>-</td>
<td>0.48</td>
</tr>
<tr>
<td>FS4</td>
<td>0.49</td>
<td>0.92</td>
<td>0.34</td>
<td>0.92</td>
<td>0.15</td>
<td>0.58</td>
<td>0.27</td>
<td>0.48</td>
<td>-</td>
</tr>
</tbody>
</table>

Lov Kumar, Santanu Kumar Rath, Ashish Sureka
Using Source Code Metrics and Multivariate Adaptive Regression
### Additional t-test Results

#### t-test: Among different Classifier

<table>
<thead>
<tr>
<th></th>
<th>MLR</th>
<th>SVM</th>
<th>MARS</th>
<th></th>
<th>MLR</th>
<th>SVM</th>
<th>MARS</th>
<th></th>
<th>MLR</th>
<th>SVM</th>
<th>MARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLR</td>
<td>0.00</td>
<td>0.36</td>
<td>0.00</td>
<td>0.00</td>
<td>-</td>
<td>0.58</td>
<td>0.00</td>
<td>0.00</td>
<td>-</td>
<td>0.56</td>
<td>8.13</td>
</tr>
<tr>
<td>SVM</td>
<td>-0.03</td>
<td>0.00</td>
<td>0.18</td>
<td>0.58</td>
<td>-</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.56</td>
<td>-</td>
<td>N-</td>
<td>3.61</td>
</tr>
<tr>
<td>MARS</td>
<td>0.15</td>
<td>0.18</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NaN</td>
<td>8.13</td>
<td>3.61</td>
<td>NaN</td>
<td>NaN</td>
</tr>
</tbody>
</table>

#### t-test: feature ranking Versus feature subset selection techniques (Average)

<table>
<thead>
<tr>
<th>Mean Difference</th>
<th>p-value</th>
<th>t-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0362</td>
<td>0.36</td>
<td>-0.9226</td>
</tr>
</tbody>
</table>
t-test Analysis

We use pairwise t-test to compare the performance of feature selection techniques and classification techniques.

One part of the result table shows the p-value and the other part shows the mean difference values of performance parameter.

Pairwise t-test has been employed to determine whether feature ranking methods work better than feature subset selection methods or both have performed equally well.
Final Conclusion and Takeaways

Experimental analysis reveals existence of a small set of source code metrics from a large number of available source code software metrics across various types which are able to accurately predict maintainability with few mis-classification errors.

The web-service maintainability prediction model developed using MARS shows better results in comparison to MLR and SVM techniques.

We can infer that oneR feature evaluation yields better results compared to other techniques.
There is a significant difference between different classification techniques. According to the value of mean difference, MARS yields better result compared to other techniques.

The performance of the feature selection methods is varied with the different classification methods used.


References


References IV

J. Bansiya and C. G. Davis.
A hierarchical model for Object-Oriented design quality assessment.

V. R. Basili, L. C. Briand, and W. L. Melo.
A validation of Object-Oriented design metrics as quality indicators.

Victor R Basili, Lionel C Briand, and Walcélio L Melo.
How reuse influences productivity in object-oriented systems.
*Communications of the ACM*.

Dilek Baski and Sanjay Misra.
Metrics suite for maintainability of extensible markup language web services.
References V


Jones C.
Software quality in 2010: a survey of the state of the art.

Massimo Carbone and Giuseppe Santucci.
Fast&&serious: a uml based metric for effort estimation.
In *Proceedings of the 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE02)*, pages 313–322, 2002.

Michelle Cartwright and Martin Shepperd.
An empirical investigation of an object-oriented software system.

Jie-Cherng Chen and Sun-Jen Huang.
An empirical analysis of the impact of software development problem factors on software maintainability.
References IX

S. R. Chidamber and C. F. Kemerer.
A metrics suite for Object-Oriented design.

Shyam R Chidamber, David P Darcy, and Chris F Kemerer.
Managerial use of metrics for Object-Oriented software: An exploratory analysis.

Shyam R Chidamber and Chris F Kemerer.
*Towards a metrics suite for Object-Oriented design*, volume 26. ACM.

Don Coleman, Dan Ash, Bruce Lowther, and Paul Oman.
Using metrics to evaluate software system maintainability.

Don Coleman, Bruce Lowther, and Paul Oman.
The application of software maintainability models in industrial software systems.
José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra.
Analyzing the evolution of web services using fine-grained changes.

José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra.
Predicting web service maintainability via object-oriented metrics: a statistics-based approach.

Ana Erika Camargo Cruz and Koichiro Ochimizu.
Towards logistic regression models for predicting fault-prone code across software projects.
References XI


Lov Kumar, Santanu Kumar Rath, Ashish Sureka Using Source Code Metrics and Multivariate Adaptive Regression
References XII

R. Slowinski (Ed). 

Khaled El Emam, Saïda Benlarbi, Nishith Goel, and Shesh N. Rai. 
The confounding effect of class size on the validity of object-oriented metrics. 

Khaled El Emam, Walcelio Melo, and Javam C Machado. 
The prediction of faulty classes using object-oriented design metrics. 

Ezgi Erturk and Ebru Akcapinar Sezer. 
A comparison of some soft computing methods for software fault prediction. 

L. Etzkorn, J. Bansiya, and C. Davis. 
Design and code complexity metrics for Object-Oriented classes. 
References XIII

Marios Fokaefs, Rimon Mikhail, Nikolaos Tsantalis, Eleni Stroulia, and Alex Lau.
An empirical study on web service evolution.

George Forman.
An extensive empirical study of feature selection metrics for text classification.

Jerome H Friedman.
Multivariate adaptive regression splines.

Cesare Furlanello, Maria Serafini, Stefano Merler, and Giuseppe Jurman.
Entropy-based gene ranking without selection bias for the predictive classification of microarray data.
M. Pezze G. Denaro and S. Morasca.
Towards industrially relevant fault-proneness models.  

K. Gao and T. M. Khoshgoftaar.
A comprehensive empirical study of count models for software fault prediction.  

Kehan Gao, Taghi Khoshgoftaar, and Amri Napolitano.
Exploring software quality classification with a wrapper-based feature ranking technique.  

Kehan Gao, Taghi M Khoshgoftaar, Huanjing Wang, and Naeem Seliya.
Choosing software metrics for defect prediction: an investigation on feature selection techniques.  


References XVII

Allen E.B Hudepohl J.P Hochman R, Khoshgoftar T.M.
Evolutionary neural networks: a robust approach to software reliability problems.

Ross Huitt and Norman Wilde.
Maintenance support for object-oriented programs.

Rob J Hyndman and Anne B Koehler.
Another look at measures of forecast accuracy.

A. Idri, A. Abran, and S. Mbarki.
An experiment on the design of radial basis function neural networks for software cost estimation.


Satyen Kale, Ravi Kumar, and Sergei Vassilvitskii.  
Cross-validation and mean-square stability.  

Cohesion and reuse in an Object-Oriented system.  

S Kanmani, V Rhymend Uthariaraj, V Sankaranarayanan, and P Thambidurai.  
Object-oriented software fault prediction using neural networks.  

Heena Kapila and Satwinder Singh.  
Analysis of ck metrics to predict software fault-proneness using bayesian inference.  


Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Using Source Code Metrics and Multivariate Adaptive Regression
YS Lee, BS Liang, SF Wu, and FJ Wang.
Measuring the coupling and cohesion of an object-oriented program based on information flow.

K. Levenberg.
A method for the solution of certain non-linear problems in least squares.

W. Li and S. Henry.
Maintenance metrics for the Object-Oriented paradigm.

M. Lorenz and J. Kidd.
Object-Oriented Software Metrics.


References XXV

- **Ruchika Malhotra and Yogesh Singh.**
  On the applicability of machine learning techniques for object oriented software fault prediction.

- **D. W. Marquardt.**
  An algorithm for the least-squares estimation of nonlinear parameters.

- **R. Martin.**
  Object-oriented design quality metrics an analysis of dependencies.

- **Cristian Mateos, Marco Crasso, Alejandro Zunino, and Jos? Luis Ordiales Coscia.**
  Detecting wsdl bad practices in code–first web services.

Lov Kumar, Santanu Kumar Rath, Ashish Sureka
References XXVI


- Tim Menzies, Bora Caglayan, Zhimin He, Ekrem Kocaguneli, Joe Krall, Fayola Peters, and Burak Turhan. The promise repository of empirical software engineering data, June 2012.
References XXVII

Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum.  
Selecting best practices for effort estimation.  

Bharavi Mishra, K Shukla, et al.  
Defect prediction for object oriented software using support vector based fuzzy classification model.  

Subhas Chandra Misra.  
Modeling design/coding factors that drive maintainability of software systems.  

Naouel Moha, Francis Palma, Mathieu Nayrolles, Benjamin Joyen Conseil,  
Yann-Gaël Guéhéneuc, Benoit Baudry, and Jean-Marc Jézéquel.  
Specification and detection of soa antipatterns.  
References

J Moody and J Darken C.
Fast learning in networks of locally-tunes processing units.

Nachiappan Nagappan, Laurie Williams, Mladen Vouk, and Jason Osborne.
Early estimation of software quality using in-process testing metrics: a controlled case study.

Jasmina Novakovic.
The impact of feature selection on the accuracy of naïve bayes classifier.

Hector M Olague, Letha H Etzkorn, Sampson Gholston, and Stephen Quattlebaum.
Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes.
*Software Engineering, IEEE Transactions on.*
References XXIX

Hector M Olague, Letha H Etzkorn, Sampson Gholston, and Stephen Quattlebaum.
Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes.

Paul Oman and Jack Hagemeister.
Construction and testing of polynomials predicting software maintainability.

Ganesh J Pai and Joanne Bechta Dugan.
Empirical analysis of software fault content and fault proneness using bayesian methods.

H Pao, Y.
MA addison, Wesley, 1989.
References

Z. Pawlak.
Rough sets.

Mikhail Perepletchikov, Caspar Ryan, and Keith Frampton.
Cohesion metrics for predicting maintainability of service-oriented software.

Mikhail Perepletchikov, Caspar Ryan, Keith Frampton, and Zahir Tari.
Coupling metrics for predicting maintainability in service-oriented designs.

Mikhail Perepletchikov, Caspar Ryan, and Zahir Tari.
The impact of service cohesion on the analyzability of service-oriented software.

Lov Kumar, Santanu Kumar Rath, Ashish Sureka
Using Source Code Metrics and Multivariate Adaptive Regression
Robin L Plackett.
Karl pearson and the chi-squared test.

N Narayanan Prasanth, S Ganesh, and GA Dalton.
Prediction of maintainability using software complexity analysis: An extended frt.


N Narayanan Prasanth, S Ganesh, and GA Dalton.
A quantitative approach to software maintainability prediction.

Mehwish Riaz, Emilia Mendes, and Ewan Tempero.
Predicting maintenance effort with function points.
Mehwish Riaz, Emilia Mendes, and Ewan Tempero. 
A systematic review of software maintainability prediction and metrics. 

Wagner S. 
A literature survey of the quality economics of defect-detection techniques. 

Scott L Schneberger. 
Distributed computing environments: effects on software maintenance difficulty. 

Software project effort estimation using genetic programming. 


K. Srinivasan and D. Fisher.
Machine learning approaches to estimating software development effort.

Ramanath Subramanyam and Mayuram S. Krishnan.
Empirical analysis of ck metrics for object-oriented design complexity:
Implications for software defects.

Johan AK Suykens, Jos De Brabanter, Lukas Lukas, and Joos Vandewalle.
Weighted least squares support vector machines: robustness and sparse approximation.
*Neurocomputing*.

Mende T and Koschke R.
Revisiting the evaluation of defect prediction models.
Mende T and Koschke R.
Effort-aware defect prediction models.

Empirical validation of Object-Oriented metrics on open source software for fault prediction.

Mei-Huei Tang, Ming-Hung Kao, and Mei-Hwa Chen.
An empirical study on object-oriented metrics.

D. P. Tegarden, S. D. Sheetz, and D. E. Monarchi.
A software complexity model of Object-Oriented systems.
Piotr Tomaszewski, Jim Håkansson, Håkan Grahn, and Lars Lundberg.
Statistical models vs. expert estimation for fault prediction in modified code—an
industrial case study.

Chikako Van Koten and AR Gray.
An application of bayesian network for predicting object-oriented software
maintainability.

D Wang and JA Romagnoli.
Robust multi-scale principal components analysis with applications to process
monitoring.

G. Witting and G. Finnie.
Estimating software development effort with connectionist models.
In *Proceedings of the Information and Software Technology Conference*, pages
W.Li and S.Henry.  
Object-oriented metrics that predicts maintainablity.  

Fangjun Wu.  
Empirical validation of object-oriented metrics on nasa for fault prediction.  

Yijun Yu, Jianguo Lu, Juan Fernandez-Ramil, and Phil Yuan.  
Comparing web services with other software components.  

Jialin Zhou, Zhengcheng Duan, Yong Li, Jianchun Deng, and Daoyuan Yu.  
Pso-based neural network optimization and its utilization in a boring machine.  

