Estimating Web Service Quality of Service Parameters using Source Code Metrics and LSSVM

Lov Kumar¹  Santanu Rath¹  Ashish Sureka²

¹NIT Rourkela, India (lovkumar505@gmail.com)
²Ashoka University, India (ashish.sureka@ashoka.edu.in)

QuASoQ 2017 (co-located to APSEC 2017)
# Table of Contents

1. Research Motivation and Aim  
   - Objectives and Context Setting

2. Related Work

3. Research Framework  
   - Dependent Variables- QoS Parameters  
   - Predictor Variables: Source Code Metrics

4. Empirical Analysis  
   - Experimental Dataset  
   - Principal Component Analysis (PCA)  
   - Rough Set Analysis (RSA)  
   - Machine Learning Based Approach  
   - Statistical Significance Tests and Procedures

5. Conclusion
Service Oriented Computing and Architecture

Prediction of Web Service QoS parameters

Service Oriented Computing and Architecture (SOA) paradigm consists of assembling and combining loosely coupled software components called as services for developing distributed system.

Prediction of Web Service QoS parameters is important for both the developers and consumers of the service [6].

Predicting quality of Object-Oriented (OO) Software System using different kinds of source code metrics is an area which has attracted several researchers’ attention in the past [2][17][10][4].
15 Quality of Service Parameters

**Prediction using source code metrics**

Fifteen different quality of service parameters such as Availability, Best Practices, Compliance, Conformity, Documentation, Interoperability, Latency, Maintainability, Modularity, Response Time, Reusability, Reliability, Successability, Throughput, and Testability.

Thirty seven different source code metrics on a dataset consisting of two hundred real-world Web Services.

LSSVM method with three different types of kernel functions: linear kernel, polynomial kernel and RBF kernel.
Source Code Metrics and Feature Extraction

Predictors and Indicators

**Six different sets of source code metrics** are used: all metrics (AM) for source code (thirty seven metrics), Baski and Misra Metrics suite (BMS), Harry M. Sneed Metrics suite (HMS), Object-Oriented source code metrics (OOM),

**Feature Selection and Extraction:** Principal Component Analysis (PCA) method and Rough Set Analysis (RSA)
Research Contributions

1. Application of 37 source-code metrics for prediction of 15 different Web Service QoS parameters by using LSSVM machine learning classifier with three different variants of kernel functions.

2. Application of two feature selection techniques i.e., PCA and RSA to select suitable set of source code metrics for building a predictive model.
Research shows that the quality of OO software can be estimated using several source code metrics [4] [9][1][8][7].

Bingu Shim et al. [13]

Bingu Shim et al. have defined five different quality parameters i.e., effectiveness, flexibility, discoverability, reusability and understandability for service oriented applications [13].
Mikhail et al. [11][12]
Mikhail et al. have defined SCMs in order to measure the structural coupling & cohesion of service-oriented systems [11][12].

Vuong Xuan Tran et al. [15]
Vuong Xuan Tran et al. proposed a novel approach to design and develop QoS systems and describe an algorithm to evaluate its ranking in order to compute the quality of Web services [15].
Table of Contents

1 Research Motivation and Aim
   - Objectives and Context Setting

2 Related Work

3 Research Framework
   - Dependent Variables- QoS Parameters
   - Predictor Variables: Source Code Metrics

4 Empirical Analysis
   - Experimental Dataset
   - Principal Component Analysis (PCA)
   - Rough Set Analysis (RSA)
   - Machine Learning Based Approach
   - Statistical Significance Tests and Procedures

5 Conclusion
Al-Masri et al. define 9 quality of service parameters of Web Services. They compute the QoS parameters using Web service benchmark tools.

The QoS parameters are: Availability (AV), Best Practices (BP), Compliance (CP), Documentation (DOC), Latency (LT), Response Time (RT), Reliability (REL), Successability (SA), Throughput (TP), Maintainability, Modularity, Reusability, Testability, Interoperability and Conformity.
Table of Contents

1 Research Motivation and Aim
   - Objectives and Context Setting

2 Related Work

3 Research Framework
   - Dependent Variables- QoS Parameters
   - Predictor Variables: Source Code Metrics

4 Empirical Analysis
   - Experimental Dataset
   - Principal Component Analysis (PCA)
   - Rough Set Analysis (RSA)
   - Machine Learning Based Approach
   - Statistical Significance Tests and Procedures

5 Conclusion
We compute nineteen different Object-Oriented source code metrics from the bytecode of the compiled Java files of the Web Services in our experimental dataset using CKJM extended tool\(^a\) [4].

\(^a\)http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/

Java class files from the WSDL file are generated using WSDL2Java Axis2 code generator\(^a\), which is available as an Eclipse plug-in.

\(^a\)https://sourceforge.net/projects/wsl2javawizard/
Henry M. Sneed WSDL Metric Suite

Sneed et al. develop a tool for measuring Web Service interfaces [14][5].

The suite primarily consists of six different source code metrics to measure complexity of service interfaces: Data Flow Complexity, Interface Relation Complexity, Interface Data Complexity, Interface Structure Complexity, Interface Format Complexity and Language Complexity.
Baski and Misra proposed a tool to compute six different complexity metrics of WSDL file [3].

These metrics are based on the analysis of the structure of the exchanged messages described in WSDL file which becomes the basis for computing the data complexity.
# Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting

2. Related Work

3. Research Framework
   - Dependent Variables- QoS Parameters
   - Predictor Variables: Source Code Metrics

4. Empirical Analysis
   - Experimental Dataset
     - Principal Component Analysis (PCA)
     - Rough Set Analysis (RSA)
     - Machine Learning Based Approach
     - Statistical Significance Tests and Procedures

5. Conclusion
Web Service dataset collected by Al-Masri et al. \(^a\) is used to measure the performance of the proposed LSSVM based approach.

\(^a\)http://www.uoguelph.ca/~qmahmoud/qws/

We use 200 Web Services for the analysis. The reason for selection of 200 web-services is stated in our earlier work [6] as the study presented in this paper is an extension of the previous work.
<table>
<thead>
<tr>
<th>Section</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Research Motivation and Aim</td>
<td>Objectives and Context Setting</td>
</tr>
<tr>
<td>2 Related Work</td>
<td></td>
</tr>
<tr>
<td>3 Research Framework</td>
<td>Dependent Variables- QoS Parameters</td>
</tr>
<tr>
<td></td>
<td>Predictor Variables: Source Code Metrics</td>
</tr>
<tr>
<td>4 Empirical Analysis</td>
<td>Experimental Dataset</td>
</tr>
<tr>
<td></td>
<td>Principal Component Analysis (PCA)</td>
</tr>
<tr>
<td></td>
<td>Rough Set Analysis (RSA)</td>
</tr>
<tr>
<td></td>
<td>Machine Learning Based Approach</td>
</tr>
<tr>
<td></td>
<td>Statistical Significance Tests and Procedures</td>
</tr>
<tr>
<td>5 Conclusion</td>
<td></td>
</tr>
</tbody>
</table>
The main motivation of using PCA is for transforming high dimension data space into lower dimension data space.

The lower dimension data consists of the most significant features [16]. We label the new metrics (or features) after applying PCA as principal component domain metrics.

We apply PCA with varimax rotation technique on all the software metrics.
Principal Component Analysis (PCA)

Data Set → Feature normalization - zero mean value → Eigen value and Eigen vector computation using MATLAB command ($\text{eign} = \text{eig}(\text{data})$) → Reduced set of features (metrics) are evaluated → Principal components selection based on eigenvalue being greater than 1.0

**Figure:** Sequence of Steps for Applying PCA
### Principal Component Analysis (PCA) Results

<table>
<thead>
<tr>
<th>PC</th>
<th>Eigenvalue</th>
<th>variance %</th>
<th>% Cumulative</th>
<th>Interpreted Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC1</td>
<td>6.40</td>
<td>17.30</td>
<td>17.30</td>
<td>Ce, Ca, RFC, CBO, LCO, LCOM3, CAM, DAM</td>
</tr>
<tr>
<td>PC2</td>
<td>5.8</td>
<td>15.76</td>
<td>33.06</td>
<td>DP, FP, OP, MRS, OPS, IDFC, IRC</td>
</tr>
<tr>
<td>PC3</td>
<td>3.67</td>
<td>9.94</td>
<td>43.00</td>
<td>CE, MiRV, MDC, MeRV, DW, MR</td>
</tr>
<tr>
<td>PC4</td>
<td>3.39</td>
<td>9.16</td>
<td>52.17</td>
<td>ILC, DMR, ISC, IDC</td>
</tr>
<tr>
<td>PC5</td>
<td>3.34</td>
<td>9.03</td>
<td>61.2</td>
<td>MOA, CBM, IC</td>
</tr>
<tr>
<td>PC6</td>
<td>2.50</td>
<td>6.77</td>
<td>67.98</td>
<td>MFA, NOC, DIT, IFC</td>
</tr>
<tr>
<td>PC7</td>
<td>2.23</td>
<td>6.02</td>
<td>74.00</td>
<td>NPM, WMC</td>
</tr>
<tr>
<td>PC8</td>
<td>2.14</td>
<td>5.79</td>
<td>79.79</td>
<td>AMC, MRV</td>
</tr>
<tr>
<td>PC9</td>
<td>1.36</td>
<td>3.7</td>
<td>83.5</td>
<td>LCOM</td>
</tr>
</tbody>
</table>
Table of Contents

1 Research Motivation and Aim
   • Objectives and Context Setting

2 Related Work

3 Research Framework
   • Dependent Variables- QoS Parameters
   • Predictor Variables: Source Code Metrics

4 Empirical Analysis
   • Experimental Dataset
   • Principal Component Analysis (PCA)
   • Rough Set Analysis (RSA)
   • Machine Learning Based Approach
   • Statistical Significance Tests and Procedures

5 Conclusion
Feature Selection using Rough Set Analysis (RSA)

Before the application of RSA, the input data need to be categorized. In our study, K-means clustering approach is applied for the purpose of data categorization.

After the application of K-means clustering approach, we obtain 3 clusters and the data were categorized into three groups: High, Medium, and Low correlation.
Rough Set Analysis (RSA) based Feature Selection

**Figure:** Rough Set Analysis (RSA) based Feature Selection

1. **Data Set**
2. **Feature Categorization into groups using k-means clustering method**
3. **Calculation of Lower and upper approximation of all possible sets**
4. **Identification of best set of source-code metrics**
5. **Computation of Accuracy for all possible sets**
### Source Code Metrics Identified using Rough Set Analysis

<table>
<thead>
<tr>
<th>QoS</th>
<th>Selected Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>MiRV, Ca, CC, CAM, IC, MFA, LC, SC, LCOM3, WMC, FC, MeRV</td>
</tr>
<tr>
<td>Response Time</td>
<td>Ca, DMR, LC, SC, WMC, MFA, CC, IC, CAM, LCOM3</td>
</tr>
<tr>
<td>Successability</td>
<td>CAM, LCOM3, DAM, FC, LC, DFC, MRV, ME, SC, WMC, LCO, MOA</td>
</tr>
<tr>
<td>Throughput</td>
<td>MiRV, Ce, CC, CAM, ME, MFA, LC, SC, CBM, MRV, FC, MeRV, MOA</td>
</tr>
<tr>
<td>Compliance</td>
<td>MiRV, NPM, CC, WMC, CAM, MOA, SC, LC, FC, DFC, ME, Ca, MRV, DAM</td>
</tr>
<tr>
<td>Reliability</td>
<td>LCOM3, MFA, FC, LC, DFC, CAM, SC, WMC, LCO, MOA</td>
</tr>
<tr>
<td>Latency</td>
<td>IC, FC, LC, DMR, MRV, MOA, ME, CAM, DC, DFC, NOC, LCO, NPM</td>
</tr>
<tr>
<td>Best Practices</td>
<td>MiRV, Ca, CC, CAM, ME, MFA, LC, SC, MRV, FC, MOA, WMC, DFC, NPM</td>
</tr>
<tr>
<td>Maintainability</td>
<td>CBM, DP, LCOM3, MFA, Ce, CAM, MOA</td>
</tr>
<tr>
<td>Documentation</td>
<td>CC, LC, ME, IC, SC, CAM, Ca, DFC, MRV, WMC, MeRV, FC, NPM</td>
</tr>
<tr>
<td>Reusability</td>
<td>LCOM3, FC, MDC, LCOM, DMR, SC, LC, DFC</td>
</tr>
<tr>
<td>Modularity</td>
<td>AMC, LC, DMR, SC, Ca, IC, DFC, ME, DP, MiRV, MOA, MRV, WMC</td>
</tr>
<tr>
<td>Interoperability</td>
<td>MiRV, CC, SC, MeRV, LC, WMC, MFA, DIT, CBO</td>
</tr>
<tr>
<td>Testability</td>
<td>FC, CC, RFC, ME, NOC, MiRV, DIT, SC, LC</td>
</tr>
<tr>
<td>Conformity</td>
<td>CAM, Ca, ME, DFC, FC, WMC, MRV, LC</td>
</tr>
</tbody>
</table>

Lov Kumar, Santanu Rath, Ashish Sureka

Estimating Web Service QoS Parameters
Estimating Web Service QoS Parameters

Table of Contents

1 Research Motivation and Aim
   • Objectives and Context Setting

2 Related Work

3 Research Framework
   • Dependent Variables- QoS Parameters
   • Predictor Variables: Source Code Metrics

4 Empirical Analysis
   • Experimental Dataset
   • Principal Component Analysis (PCA)
   • Rough Set Analysis (RSA)
   • Machine Learning Based Approach
   • Statistical Significance Tests and Procedures

5 Conclusion
We use LSSVM as regression technique to generate models for predicting QoS parameters.

We also examine LSSVM different kernel functions to investigate if we can achieve better result and compare the performance of various kernel functions.

We apply statistical significance tests to compare the performance of one prediction technique over other approaches.
Proposed Steps Used for the QoS Prediction

- WSDL File
  - Baski and Misra Metric suite
  - Sneed Tool
  - Henry M. Sneed Metrics
- QoS
  - Object Oriented metrics (OOM)
- Source Code
  - CKJM Extended
- Object-Oriented Metrics
- PCA
- RSA
- Least Square Support Vector Machines (LSSVM method)
- Validation of developed models
- Linear Kernel
- Polynomial Kernel
- RBF Kernel
- Performance Evaluation
- Statistical Test
- Model comparison
## Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting

2. Related Work

3. Research Framework
   - Dependent Variables- QoS Parameters
   - Predictor Variables: Source Code Metrics

4. Empirical Analysis
   - Experimental Dataset
   - Principal Component Analysis (PCA)
   - Rough Set Analysis (RSA)
   - Machine Learning Based Approach
   - Statistical Significance Tests and Procedures

5. Conclusion
Procedure and Results

We conduct t-test to determine which prediction method and feature selection techniques performs relatively better or does the models perform equally well.

We analyze all the results based on the 0.05 significance level, i.e. two models are significantly different (null hypothesis rejected) if the p-value is less than 0.05 (the cut-off value).
Figure: Box-Plot Visual Analysis (Linear Kernel)
Lov Kumar, Santanu Rath, Ashish Sureka

Estimating Web Service QoS Parameters
Box-Plot Visual Analysis (RBF Kernel)

Figure: Box-Plot Visual Analysis (RBF Kernel)
In case of linear kernel function, we observe that the model built by considering selected set of metrics using RSA as input has low values of MMRE, MAE and RMSE in comparison with other sets of metrics.

In case of polynomial kernel function, we observe that the model built by considering all metrics has low value of MMRE, MAE and RMSE in comparison to other sets of metrics.
Model developed by considering Baski and Misra Metric has low value of MMRE, MAE, and RMSE in comparison with other sets of metrics.

Model developed by considering selected set of metrics using RSA as input results in better performance as compared to other metrics.
Model developed by considering AM as input results in better performance as compared to others.

Model developed by considering BMS as input obtained better performance as compared to others.
Result of t-test: Among Different Metrics Set

<table>
<thead>
<tr>
<th></th>
<th>MMRE</th>
<th></th>
<th>MAE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P-Value</td>
<td></td>
<td>Mean Difference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AM OOM HMS BMS RSA PCA</td>
<td>AM OOM HMS BMS RSA PCA</td>
<td>AM OOM HMS BMS RSA PCA</td>
<td></td>
</tr>
<tr>
<td>AM</td>
<td>1.000 0.063 0.031 0.063 0.031 0.031</td>
<td>1.000 0.031 0.031 0.031 0.031 0.031</td>
<td>0.031 0.031 1.000 0.063 0.031 0.031</td>
<td></td>
</tr>
<tr>
<td>OOM</td>
<td>0.063 1.000 0.031 0.031 0.031 0.031</td>
<td>0.031 1.000 0.031 0.031 0.031 0.031</td>
<td>0.031 0.031 0.063 1.000 0.031 0.031</td>
<td></td>
</tr>
<tr>
<td>HMS</td>
<td>0.031 0.031 1.000 0.031 0.094 0.031</td>
<td>0.031 0.031 0.031 0.031 1.000 0.031</td>
<td>0.031 0.031 0.031 0.031 1.000 0.031</td>
<td></td>
</tr>
<tr>
<td>BMS</td>
<td>0.063 0.031 0.031 1.000 0.031 0.031</td>
<td>0.031 0.031 0.031 0.031 1.000 0.031</td>
<td>0.031 0.031 0.031 0.031 1.000 0.031</td>
<td></td>
</tr>
<tr>
<td>RSA</td>
<td>0.031 0.031 0.094 0.031 1.000 0.031</td>
<td>0.031 0.031 0.031 0.031 1.000 0.031</td>
<td>0.031 0.031 0.031 0.031 1.000 0.031</td>
<td></td>
</tr>
<tr>
<td>PCA</td>
<td>0.031 0.031 0.031 0.031 0.031 1.000</td>
<td>0.031 0.031 0.031 0.031 0.031 1.000</td>
<td>0.031 0.031 0.031 0.031 0.031 1.000</td>
<td></td>
</tr>
</tbody>
</table>
Hypothesis Testing Results

From the result, we observe that there is a significant difference between the kernel functions. This interpretation is due to the fact that the p-value is lower than 0.0167 (rejecting the null hypothesis and accepting the alternate hypothesis).

However by closely examining the value of mean difference, RBF kernel function yields better result as compared to other kernel functions.
### t-test: Among different Kernel

<table>
<thead>
<tr>
<th></th>
<th>MMRE</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lin</td>
<td>Poly</td>
<td>RBF</td>
</tr>
<tr>
<td><strong>P-Value</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lin</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Poly</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>RBF</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>MMRE</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lin</td>
<td>Poly</td>
<td>RBF</td>
</tr>
<tr>
<td><strong>Mean Difference</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lin</td>
<td>0.000</td>
<td>-0.051</td>
<td>0.155</td>
</tr>
<tr>
<td>Poly</td>
<td>0.051</td>
<td>0.000</td>
<td>0.206</td>
</tr>
<tr>
<td>RBF</td>
<td>-0.155</td>
<td>-0.206</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Hypothesis Testing Results

We infer that there is no significant difference between sets of metrics. We arrive at this conclusion due to the fact that the p-value is greater than 0.0033 (accepting the null hypothesis).

By closely examining the value of mean difference, we infer that the object-oriented Metrics are yields better performance results in comparison to other sets of metrics.
We conclude that there exists a high correlation between Object-Oriented metrics and WSDL metrics.

There is a statistically significant difference between the performance of the predictive models built using three different LSSVM kernel functions.

There is no statistically significant difference between different sets of source code metrics.
Conclusions and Takeaways

No one set of source-code metrics dominate the other sets for any QoS parameter and vice-versa.

The RBF kernel for LSSVM method yields better performance results compared to other kernel functions.

The object-oriented metrics yields better result compared to other sets of source code metrics.

It is possible to estimate the QoS parameters of Web Services using source code metrics and LSSVM based method.
References I


References II


