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

Lov Kumar[1] Santanu Rath[1] Ashish Sureka[2]

[1]NIT Rourkela, India (lovkumar505@gmail.com)

[2]Ashoka University, India (ashish.sureka@ashoka.edu.in)

QuASoQ 2017 (co-located to APSEC 2017)

Table of Contents

Table of Contents

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.

Literature Survey - 1

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].

Literature Survey - 2

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

Dependent Variables- QoS Parameters

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

Object-Oriented Source Code Metrics

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/wsdl2javawizard/

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 Metrics

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

Web Service Dataset

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 of Contents

Feature Extraction using Principal Component Analysis (PCA)

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)

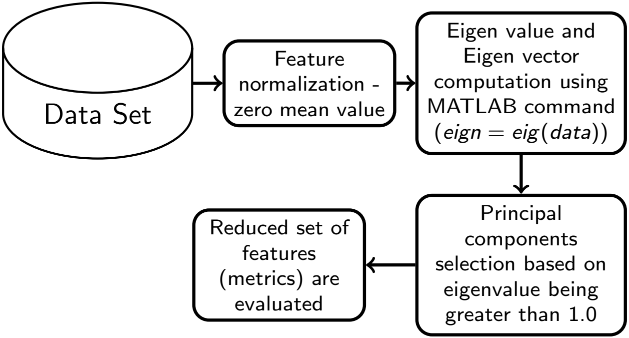

Figure: Sequence of Steps for Applying PCA

Principal Component Analysis (PCA) Results

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

Table of Contents

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

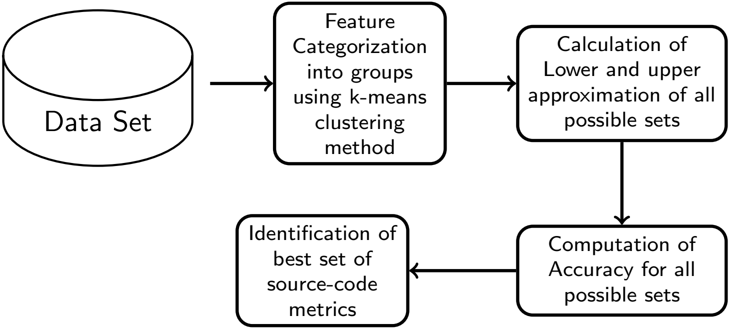

Figure: Rough Set Analysis (RSA) based Feature Selection

Source Code Metrics Identified using Rough Set Analysis

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

Table of Contents

LSSVM Model

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

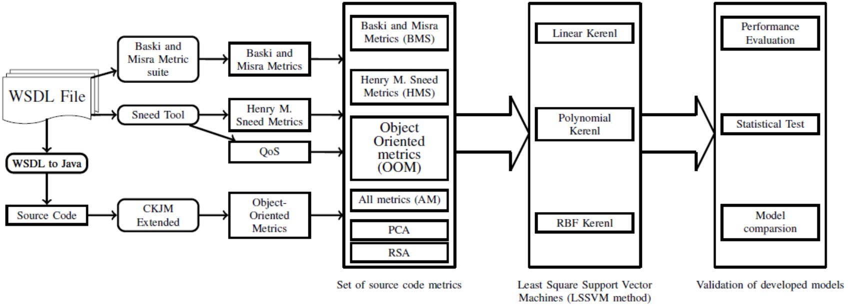

Table of Contents

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)

Box-Plot Visual Analysis (Linear Kernel)

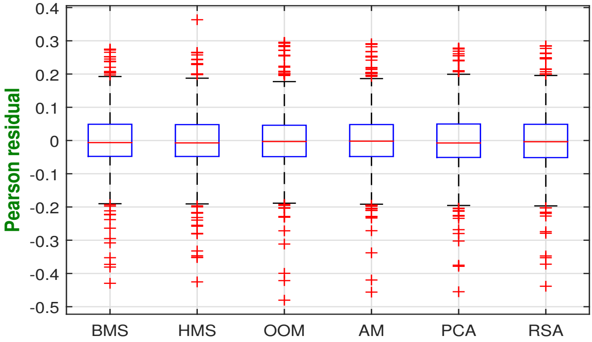

Figure: Box-Plot Visual Analysis (Linear Kernel)

Box-Plot Visual Analysis (Polynomial Kernel)

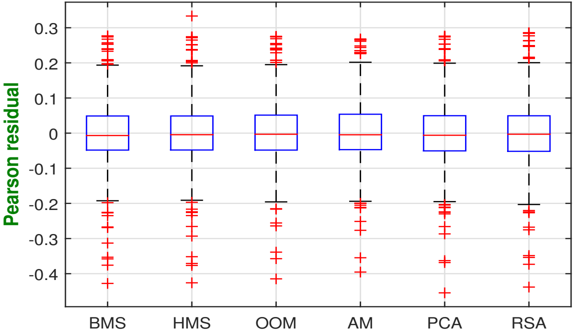

Figure: Box-Plot Visual Analysis (Polynomial Kernel)

Box-Plot Visual Analysis (RBF Kernel)

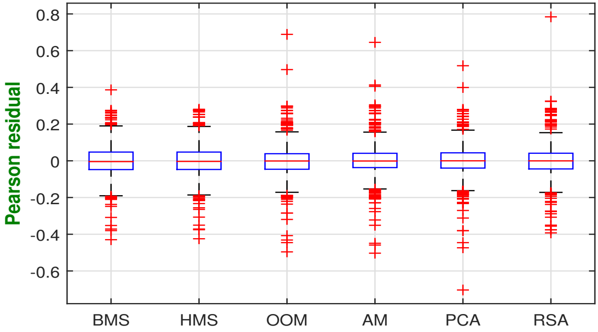

Figure: Box-Plot Visual Analysis (RBF Kernel)

Linear Kernel and Polynomial 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.

RBF kernel, RSA

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.

All Metrics, BMS 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

	P-Value											
	MMRE						MAE					
	AM	OOM	HMS	BMS	RSA	PCA	AM	OOM	HMS	BMS	RSA	PCA
AM	1.000	0.063	0.031	0.063	0.031	0.031	1.000	0.031	0.031	0.031	0.031	0.031
OOM	0.063	1.000	0.031	0.031	0.031	0.031	0.031	1.000	0.031	0.031	0.031	0.031
HMS	0.031	0.031	1.000	0.031	0.094	0.031	0.031	0.031	1.000	0.063	0.031	0.031
BMS	0.063	0.031	0.031	1.000	0.031	0.031	0.031	0.031	0.063	1.000	0.031	0.031
RSA	0.031	0.031	0.094	0.031	1.000	0.031	0.031	0.031	0.031	0.031	1.000	0.031
PCA	0.031	0.031	0.031	0.031	0.031	1.000	0.031	0.031	0.031	0.031	0.031	1.000
	Mean Difference											
	MMRE						MAE					
	AM	OOM	HMS	BMS	RSA	PCA	AM	OOM	HMS	BMS	RSA	PCA
AM	0.000	0.020	-0.392	-0.013	-0.345	-0.268	0.000	0.038	-0.082	-0.068	-0.142	-0.158
OOM	-0.020	0.000	-0.412	-0.033	-0.365	-0.288	-0.038	0.000	-0.043	-0.030	-0.103	-0.120
HMS	0.392	0.412	0.000	0.378	0.047	0.123	0.082	0.043	0.000	0.013	-0.060	-0.077
BMS	0.013	0.033	-0.378	0.000	-0.332	-0.255	0.068	0.030	-0.013	0.000	-0.073	-0.090
RSA	0.345	0.365	-0.047	0.332	0.000	0.077	0.142	0.103	0.060	0.073	0.000	-0.017
PCA	0.268	0.288	-0.123	0.255	-0.077	0.000	0.158	0.120	0.077	0.090	0.017	0.000

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

P-Value									
	MMRE			MAE			RMSE		
	Lin	Poly	RBF	Lin	Poly	RBF	Lin	Poly	RBF
Lin	1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000
Poly	0.000	1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000
RBF	0.000	1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000
Mean Difference									
	MMRE			MAE			RMSE		
	Lin	Poly	RBF	Lin	Poly	RBF	Lin	Poly	RBF
Lin	0.000	-0.051	0.155	0.000	-0.015	0.047	0.000	-0.016	0.057
Poly	0.051	0.000	0.206	0.015	0.000	0.063	0.016	0.000	0.074
RBF	-0.155	-0.206	0.000	-0.047	-0.063	0.000	-0.057	-0.074	0.000

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**.

Conclusions and Takeaways

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

[1] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of Object-Oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, October 1996.

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

[3] Dilek Baski and Sanjay Misra. Metrics suite for maintainability of extensible markup language web services. *IET Software*, 5(3):320–341, 2011.

[4] S. R. Chidamber and C. F. Kemerer. A metrics suite for Object-Oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, June 1994.

[5] José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, and Alejandro Zunino. Estimating web service interface quality through conventional object-oriented metrics. *CLEI Electron. J*, 16(1), 2013.

[6] Lov Kumar, Santanu Kumar Rath, and Ashish Sureka. Predicting quality of service (qos) parameters using extreme learning machines with various kernel methods. In *Quantitative Approaches to Software Quality*, page 11, 2016.

References II

[7] Lov Kumar, Santanu Kumar Rath, and Ashish Sureka. Using source code metrics and multivariate adaptive regression splines to predict maintainability of service oriented software. In *High Assurance Systems Engineering (HASE), 2017 IEEE 18th International Symposium on*, pages 88–95. IEEE, 2017.

[8] Lov Kumar, Santanu Kumar Rath, and Ashish Sureka. Using source code metrics to predict change-prone web services: A case-study on ebay services. In *Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), IEEE Workshop on*, pages 1–7. IEEE, 2017.

[9] W. Li and S. Henry. Maintenance metrics for the Object-Oriented paradigm. In *International Software Metrics Symposium*, pages 52–60, 1993.

[10] Ruchika Malhotra and Yogesh Singh. On the applicability of machine learning techniques for object oriented software fault prediction. *Software Engineering: An International Journal*.

[11] Mikhail Perepletchikov, Caspar Ryan, and Keith Frampton. Cohesion metrics for predicting maintainability of service-oriented software. In *QSIC*, pages 328–335. IEEE, 2007.

References III

[12] Mikhail Perepletchikov, Caspar Ryan, Keith Frampton, and Zahir Tari. Coupling metrics for predicting maintainability in service-oriented designs. In *ASWEC*, pages 329–340. IEEE, 2007.

[13] Bingu Shim, Siho Choue, Suntae Kim, and Sooyong Park. A design quality model for service-oriented architecture. In *2008 15th Asia-Pacific Software Engineering Conference*, pages 403–410. IEEE, 2008.

[14] Harry M Sneed. Measuring web service interfaces. In *Web Systems Evolution (WSE)*, pages 111–115. IEEE, 2010.

[15] Vuong Xuan Tran, Hidekazu Tsuji, and Ryosuke Masuda. A new qos ontology and its qos-based ranking algorithm for web services. *Simulation Modelling Practice and Theory*, 17(8):1378–1398, 2009.

[16] D Wang and JA Romagnoli. Robust multi-scale principal components analysis with applications to process monitoring. *Journal of Process Control*, 15(8):869–882, 2005.

[17] Yuming Zhou and Hareton Leung. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. 32(10):771–789, 2006.