Service Operation Impedance and its role in projecting some key features in Service Contracts

Sid Kargupta[1] and Sue Black[2]

[1] EMC Consulting, EMC, Southwark Bridge Road, London SE1 9EU, UK
sid.kargupta@emc.com
[2] Dept. Of Information & Software Systems, University of Westminster, Middx HA1 3TP,
s.e.black@wmin.ac.uk

Abstract: This paper introduces the notion of implicit Operation Impedance (I) and Operation Potential (V) in Service Provider-Consumer contracts. 'I' is the *runtime* composite resultant of all the activity delays of the components supporting the Service Operation. This work establishes that 'I', which impacts the overall Operation Performance (P), is influenced by the underlying *application* components' activities in *distinct* patterns. A high-level *runtime* abstract model is empirically deduced between 'I', 'V' and 'P' by applying established mathematical techniques. Model based *indicative* values of some features are *computed* against variability of the operation's components. Lookup datasets against different system configurations are created to associate these *computed* values to the *actual* empirical values of other features. Established mathematical techniques applied with appropriate regression types to enable trend extrapolation/interpolation. The datasets/patterns affirmed effectiveness of the 'I' based model as a means of *decoupled*, *bidirectional* i.e. top-down and bottom-up impact assessment of modifications to the operation's underlying *application* components on 'P' ('V' constant) or 'V' ('P' constant) *without* repetitive full scale *external* performance/benchmark testing. This also enables fine tuning of application components to *retrofit* prescribed Quality of Service (QoS). The paper briefly mentions a Matrix Transpose/Inverse technique for future assessment of multiple component changes simultaneously.

1 Introduction

Service Operations of a Service Provider are catered by underlying application components laid on top of system components. For every Service Operation, the *activities* of these components cumulatively create the composite impedance 'I' implicit to that operation, which eventually impacts the operation's Performance 'P'. The application components are often modified due to changes in business requirements while the underlying system remains the same. Extending on the fundamentals of previous work [1], this research explores one level of abstraction from system resources to application components and verifies a higher level pattern based projection of certain non-functional features of a Service Operation for modifications to the supporting application components. Pure functional models do not capture quantitative information about resource consumption behavior [1]. So, the

Service Operation's application components are decomposed into *atomic* activities or *Delay Points* like in-memory Data Processing, File I/O, Database Interaction, XML processing etc., which interface with the system resources (both Queue and Delay) and contribute to the overall Service Operation Impedance 'I'. The paper tries to establish that the *total delay* (or Impedance) for *each* type of Delay Point across all the supporting application components influence 'I' and hence 'P' and 'V' in a *distinct* pattern i.e.

$$I = f(\sum I_{DLPi}) \; [i=1 \text{ to } n]$$

where I_{DLP1} is the Impedance by a particular Delay Point of Component1. *Atomicity* of Delay Points is very important as Delay Point types determine their nature of system resource usage, which then manifests as the Delay Point impact pattern. Delay Points should not overlap. 'I' acts as a *connector* between the Service Operation's internal application Delay Points and external non-functional features. This research focuses on variations to application components/Delay Points instead of inbound workload. *Operation Potential* (V) is the differential between the maximum request load the Service Operation can cater to maintaining QoS (aka Service Operation's "stress point") and the Service Operation's contractual request load. *Operation Performance* (P) is the measure of Service Operation's performance under a given load. The less the response time, the more is 'P'. So, 'P' is computed as the reciprocal of the Average Response Time (ART) of the Service Operation. *Operation Impedance* (I) is the *runtime* composite delay introduced by the different Delay Points across all the components supporting the Service Operation. Network latencies (inter-component and Provider-Consumer) contribute to 'I' as well.

2 Problem Statement and Motivation

Significant research has been performed towards measuring and predicting throughput, response time and congestion using queuing network principles. Ways to model, analyze and plan for web performance problems have been illustrated in details [1]. High performance website design techniques involving redundant hardware, load balancing, web server acceleration and efficient management of dynamic data [2] have been discussed. Methods are devised for dynamic selection of services based on user specified preferences and to predict performance of component based services depending on the underlying technology platforms [3, 4]. In [5, 6], different methods of generating performance models and prediction have been discussed. An assembler tool and a methodology to automatically generate performance models for component based systems have been explored. A performance prediction approach comprising of gathering empirical performance results on COTS middleware infrastructure, a reasoning framework for understanding architectural trade-offs and relationships to technology features and predictive mathematical models to describe application behavior on the middleware technology has been investigated. Different model-based software performance prediction approaches have been classified and evaluated in [7]. Queuing network based methodologies, Architectural Pattern based methodologies, software performance analysis through UML descriptions and other approaches have been discussed.

Further research [8, 9, 10] has explored various methods of component based performance evaluation with *top-down* approach focusing on inbound workload, profiling, software containers, UMLs and transactions. However, often application developers find it convenient to analyze application level outputs than system resource or service level diagnostics, for which other human resources are required. Hence, it will be helpful to explore *generic*, application level, *bottom-up* methods to assess during development the impact of application component modifications at Delay Point granularity on other non-functional features. The notion of 'I' to facilitate the above through *visual patterns* remains unexplored. A high level abstract *runtime* model for 'V', 'P', 'I' and Delay Points related to Service Operations remains to be discussed. Typically, we still have to recourse to performance/benchmark testing of the whole system for impact analysis of application component modifications.

3 Aims and Objectives

This research aims to achieve the following objectives:

1) For Service Operations, empirically deduce a high level abstract runtime model for Service Operation Potential 'V', Service Operation Performance 'P' and overall Service Operation Impedance 'I'.

2) Decompose the application components supporting the Service Operation into atomic Delay Points.

3) Compute model based *indicative* values of Service Operation Impedance and extract its distinct variation patterns against variability of *actual* component Delay Point impedances and other non-functional features. Use Least Square Fitting (LSF) and appropriate regression types to derive pattern lines to enable bottom-up and top-down projections of the non-functional features related to service load and performance.

4 Proposed Methodology

To increase precision of the model and *standardize* request resource requirements, partitioning of the request load is achieved by constraining the model and method to Service Operation level. Different Service Operations from the same Provider may have different resource requirements.

4.1 Deducing the high level, abstract *runtime* model for V, I and P

A Service Framework comprising of Web Services, Servlets, RMI Server, Socket Server, a multi-threaded Web Service Client etc. was created to simulate a Service Contract with provision to vary the various component Delay Points. Tests were run by gradually increasing the request load to the Service Operation. Assuming a *stress point* for the Service Operation, we observed a typical *finite queue* system curve [1] for 'V' versus 'P'. Accepting approximation error, for simplifying the model, *Piecewise Linear* model is applied to divide 'V' values into 3 *bands*, each with a

linear regression (affine form) as the best fit for 'P'. Direct proportionality between 'P' and 'V' considered for each 'V' *band*:

P = IV + *c* where I is the *constant of proportionality* with I and *c* band specific

At a *given* time T_1, for requests to the *same* Service Operation, the request/process type, system configuration, resource requirements and contract load condition will be ideally the same. Today, services are run on multi-core, multi CPU servers. So, for simplicity, we assumed Multi-Processor Single Class Queuing Network (open or closed) model approximation [1]. With *m* resources and *D* service demand at each resource, the service demand at the single resource queue will be *D*/*m* and for the delay resource will be $D(m\text{-}1)/m$. Under light load, the Residence time (R_i') is *D* (proven) and under heavier load, it will be dominated by the single resource queue:

$$R_i' = V_i W_i + D_i$$

where V_i is the average no. of visits, W_i is the average waiting time and D_i is the service demand for a request at queue *i*. As the requests are to the *same* Service Operation, applying all the above constraints, D_i and V_i will ideally be same for all requests. As we used the ART of responses in test runs, the variability of W_i is averaged out. Considering all the above, R_i' is assumed consistent for all requests at queue *i*. The experiments had co-located components with local calls between them. Also, only formal Service Contracts are in scope with dedicated, controlled network traffic and *not* any random service access over public network. Hence, at *runtime*, no unpredictable fluctuation of network bandwidth or latency is assumed. Average resource usage effect of other Service Operations on requests of the tested Service Operation is assumed. With *all* the above constraints, we assumed *consistency* of overall impedance for processing requests to the *same* Service Operation at T_1 for a 'V' *band* and mapped the *runtime* Operation Impedance to the proportionality constant 'I'.

4.2 Pattern Extraction and Validation for Data Processing Delay Points

Some illustrative components are created with Data Processing, File I/O, XML Processing and other Delay Points. Keeping the rest of the configuration constant ('V' kept positive), the Data Processing Delay Point intensities of the components were incrementally varied. Empirical data for *actual* overall 'P', *computed indicative* values of overall 'I' (say 'I_O') based on the model:

$$P = IV + c$$

for the *relevant* 'V' *band*, the *actual* average Data Processing Delay Point impedance (I_{DP}) and the Data Processing Impedance Factor ($IF_{DP} = I_O/I_{DP}$) was recorded. The following data models 'I_{DP}' versus 'I_O', 'I_{DP}' versus 'IF_{DP}' and 'I_O' versus 'P' showed *distinct* trends in variation, which were consistent but *not* purely linear. Accepting approximation error, for simplicity, LSF for *Linear*, *Exponential, Polynomial* and

Power regression types and *Piecewise Linear* models were verified. For 'I_{DP}'(x_i) versus 'I_O'(y_i), pattern line with *Polynomial* regression of 3rd order was the best fit:

$$y_i = 2E{+}07x_i^3 - 250431x_i^2 + 3008.6x_i + 8.7436$$

For 'I_O'(x_i) versus 'P'(y_i) and 'I_{DP}'(x_i) versus 'IF_{DP}'(y_i) pattern line with *Power* regression was best fit:

$$y_i = f(x_i) = Ax_i^B \text{ where } B = b, A = e^a, a \text{ and } b \text{ are LSF coefficients}$$

Similar types of *distinct* patterns i.e. *Polynomial* regression of 3rd order as best fit for 'I_{FIO}'(x_i) versus 'I_O'(y_i) and Power regressions for 'I_O'(x_i) versus 'P'(y_i) etc. are extracted for all the above non-functional features by varying the File I/O Delay Points. Although the types are similar, the functions had *different* values from the Data Processing Delay Point patterns. For example, for 'I_{FIO}'(x_i) versus 'I_O'(y_i), the best fit pattern line with *Polynomial* regression of 3rd order had the regression function:

$$y_i = 27.807x_i^3 - 77.133x_i^2 + 296.92x_i + 7.737$$

Tests are performed to validate the extracted patterns. Results affirmed (with some approximation errors) the *distinct* underlying patterns of variations in 'I_O' due to changes in application components/Delay Points under a given load. From a *projected* value of 'IF_{DP}' corresponding to a given *actual* 'I_{DP}', we could also *project* 'I_O':

$$I_O = IF_{DP} \text{ x } I_{DP} + e$$

where '*e*' is the error factor. Figures1, 2 and 3 present the empirical graphs of 'I_{DP}' versus 'I_O', 'I_O' versus 'P' and 'I_{DP}' versus 'IF_{DP}'. Pattern validation is highlighted.

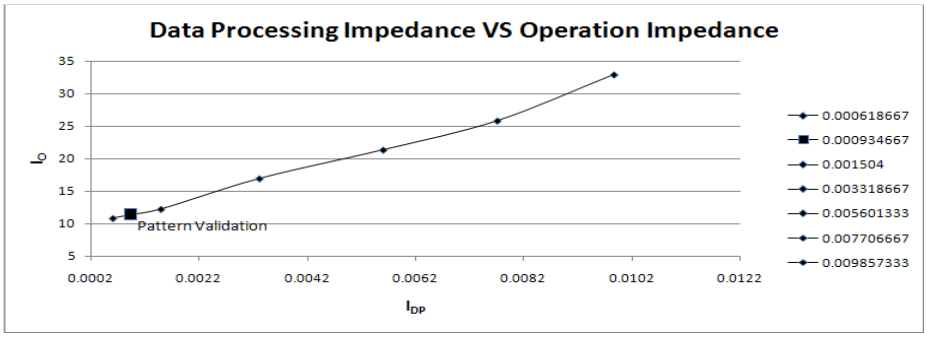

Fig. 1: Empirical Data Graph for I_{DP} vs I_O for varying Data Processing

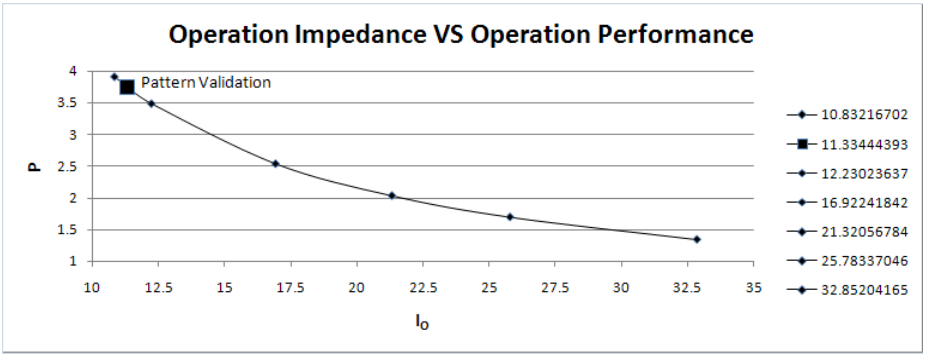

Fig. 2: Empirical Data Graph for I_O vs P for varying Data Processing

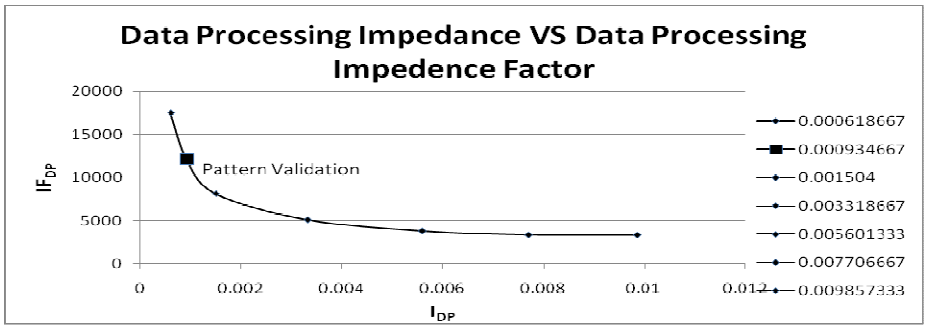

Fig. 3: Empirical Data Graph for I_{DP} vs IF_{DP} for varying Data Processing

Figures 4and 5 present the empirical graphs of 'I_{FIO}' versus 'I_O', 'I_O' versus 'P' for File I/O processing variations. Pattern validation is highlighted.

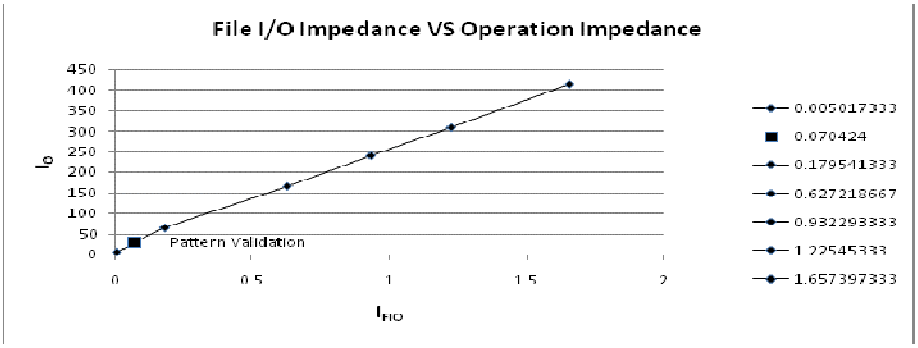

Fig. 4: Empirical Data Graph for I_{FIO} vs I_O for varying File I/O

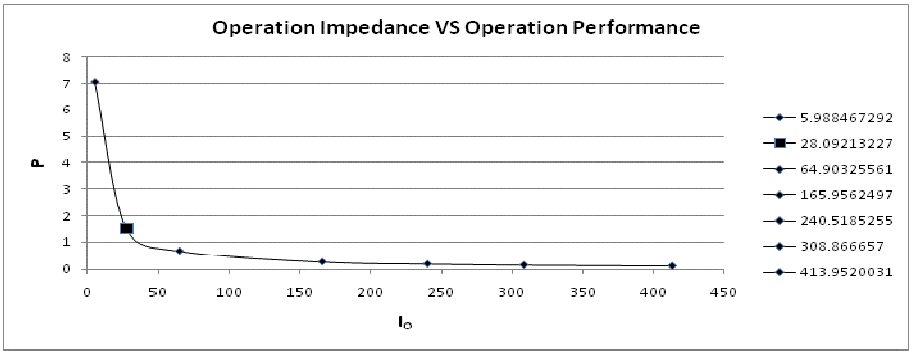

Fig. 5: Empirical Data Graph for I_O vs P for varying File I/O

4.3 Plan for Further Work

For precision, Delay Point atomicity needs to be increased e.g. file type specific File I/O Delay Point. Model calibration needs to be verified. More Delay Points need to be tested e.g. database interaction/contention has not been verified yet. Delay Points were varied one type at a time but real world component modifications will be more complex with multiple Delay Point types modified simultaneously. For this, a method involving *Matrix Transpose and Inverse* technique can be adopted for both *linear* and *polynomial* relations. Different combinations of Delay Point variations and corresponding 'I_O' can be recorded in *Matrices*. Atomic Delay Points may be treated as *independent* variables. We can find out the *best fit* Delay Point Impedance coefficient vector *X*:

$$X = (A^T A)^{-1} A^T B$$

where *A* is the matrix containing rows of Delay Point Impedances I_{DP}, I_{FIO} etc. from different test runs and *B* is the single column matrix of 'I_O' for each row in *A*. *X* will facilitate 'I_O' projection for any arbitrary combination of Delay Point Impedances. Minimizing components system resource sharing by spreading the service framework would be good. All of these should enhance overall method precision.

5 Main Contribution to Web Engineering

The model will facilitate simple, generic, pattern based means for *bidirectional* i.e. top-down and bottom-up projections (with some error factors) of 'P', 'V', 'I' and application Delay Point impedances. It will guide fine tuning of the application components at Delay Point levels to *retrofit* new 'V', 'P' or both requirements. Importantly, we believe this method will allow upfront impact analysis of application component changes during development by the developers themselves without the need of additional testing/system admin resources or much *external* tool overhead. This should help address the typical resourcing issues faced during service component

enhancements and provide potential for time, resource and cost savings. Also, *repetitive* performance or benchmark (e.g. TPC-C, TPC-App) testing of the *whole* system will not be required. It will be required initially during pattern creation through application component's Delay Point variation simulation. Thereafter, during future modifications, the developers will need to record the total delay of the modified Delay Point types across the Service Operation components while system testing with some load and plot the data on the patterns for the *applicable* 'V' band. We wouldn't need software monitors and hence overcome their inherent overhead and OS dependency shortcomings [1].

References

1. Daniel A. Menasce, Virgilio A. F. Almeida: Capacity Planning for Web Services. Metrics, Models and Methods. Prentice Hall PTR, Upper Saddle River, NJ 07458 (2002)
2. Arun Iyengar, Jim Challenger, Daniel Dias and Paul Dantzig: High Performance Web Site Design Techniques. Web Design, IEEE Internet Computing, March-April (2000)
3. D. Ardagna and B. Pernici, Adaptive Service Composition in Flexible Processes, IEEE Transactions on Software Engineering, Vol. 33, No.6, June (2007)
4. Yan Liu, Alan Fekete and Ian Gorton, Design-Level Performance Prediction of Component-Based Applications, IEEE Transactions on Software Engineering, Vol.31, No.11 (2005)
5. Xiuping Wu, David McMullan and Murray Woodside, Component Based Performance Prediction, Proceedings of the 6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction (2003)
6. Shiping Chen, Ian Gorton, Anna Liu and Yan Liu, Performance Prediction of COTS Component-based Enterprise Applications, Journal of Systems and Software, Vol 74, Issue 1, Pages 35-43 (2005)
7. Simonetta Balsamo, Antinisca Di Marco, Paola Inverardi and Marta Simeoni, Model-Based Performance Prediction in Software Development: A Survey, IEEE Transactions on Software Engineering, Vol.30, No.5 (2004)
8. Connie U. Smith and Lloyd G. Williams, Performance Engineering Evaluation of Object-Oriented Systems with SPE.ED, Computer Performance Evaluation: Modelling Techniques and Tools, No.1245,Springer-Verlag, Berlin (1997)
9. Christopher Stewart and Kai Shen, Performance Modeling and System Management for Multi-component Online Services, Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, May2-4, Boston, MA, USA (2005)
10. K.S. Jasmine and R. Vasantha, Design Based Performance Prediction of Component Based Software Products, Proceedings of the World Academy of Science, Engineering and Technology, Vol 24, ISSN 1307-6884, October (2007)