A Statistical Method for Middleware System Architecture Evaluation

Jacky W. Keung[1,2], Yan Liu[1,2], Kate Foster[3] and Thong Nguyen[3]
[1]*National ICT Australia Ltd. (NICTA) Sydney Australia*
[2]*CSE, UNSW, Sydney Australia*
[3]*Air Operation Division, Defence Science and Technology Organisation, Australia*
{Jacky.Keung, Jenny.Liu}@nicta.com.au, {Kate.Foster, Thong.Nguyen}@dsto.defence.gov.au

Abstract

The architecture of complex software systems is a collection of decisions that are very expensive to change. This makes effective software architecture evaluation methods essential in today's system development for mission critical systems.

We have previously developed MEMS for evaluating middleware architectures, which provides an effective assessment of important quality attributes and their characterizations. To provide additional quantitative assessments on the overall system performance using actual runtime data, we employed a set of statistical procedures in this work. Our proposed assessment procedures comprises a standard sensitivity analysis procedure that utilizes leverage statistics to identify and remove influential data points, and an estimator for evaluating system stability and a metric for evaluating system load capacity.

Experiments were conducted using real runtime datasets. Results show that our procedures effectively identified and isolated abnormal data points, and provided valuable statistics to show system stability.

Our approach thus provides a sound statistical basis to support software architecture evaluation.

1. Introduction

In the domain of software engineering, mission critical systems incorporate component-based and distributed computing systems and these systems are becoming more powerful and complex. This growth in system complexity results in a corresponding increase in the complexity of the software driving these systems. As a result, software intensive acquisition projects often incur schedule delays, cost overruns and reduced functionality at the end. To remedy this situation, a software architecture evaluation framework can provide a means for identifying the technical risks of a proposed architecture.

Traditional approaches to architecture evaluation and design level analysis focused on solving high-level stakeholders' conflicting requirements and are more suited to greenfield project development. The evaluation techniques discussed in this paper are for evaluating COTS middleware systems at fine-grained levels of detail using statistical models. The methodology employed in the research allows a thorough investigation of the actual performance of COTS software systems, which will help provide more rigorous evidence into the decision making process during an acquisition project.

In this paper, we suggest that by using leverage statistics and significance test combining the notion of a control chart used in SPC, it is possible to determine abnormal data points in the given runtime dataset. Weighting the overall dataset and its linear model allows us to reveal its actual performance.

Results show that our method provides insights into the actual system behavior and pinpointing software components that require further improvements to meet software requirement specifications.

Section 2 presents an overview of the related work. Section 3 describes our method and the underlying theory using leverage statistics. Section 4 provides an equilibrium test to understand the limitations and conditions of the system under investigation. Section 5 provides the datasets and the analysis procedure. Section 6 presents the result, and section 7 discusses the result and provides further directions. Section 8 concludes the paper.

2. Background and Related Work

This study is linked to a previous project on evaluating middleware architecture for Airborne Mission Systems that developed a Method for Evaluating Middleware architectureS (MEMS) [1], by rating multiple software quality attributes.

1530-0803/10 $25.00 © 2010 IEEE
DOI 10.1109/ASWEC.2010.39

The output of MEMS helps to determine the suitability of the architecture to meet quality goals of the system. Measurements of performance metrics are collected based on the test scenarios developed in the previous project. Following the evaluation process defined in MEMS, empirical results are collected from the established test bed. One key challenge is that a large amount of experimental system runtime data for analysis is produced from the measurements. Therefore, a systematic analysis method is essential to evaluate and analyze the resulting datasets.

2.1 Architecture evaluation methods

Software architecture evaluation methods and techniques focus on understanding the relationship between software architecture and one or more quality attributes to ensure that the system ultimately achieves its quality goals while still supporting its functional requirements [2]. A review of these techniques can be found in [3] and [4].

MEMS falls into the category of scenario-based software architecture evaluation methods [2]. Scenarios are defined to understand how a software architecture responds with respect to attributes such as maintainability, reliability, usability and performance. Examples of scenario-based methods are: Software Architecture Analysis Methods (SAAM), Architecture Trade-off Analysis Method (ATAM) and Architecture Level Modifiability Analysis (ALMA)[4]. MEMS specifically targets middleware, which is a component of the overall software architecture to be evaluated. Therefore MEMS demands middleware specific techniques and tools to support the evaluation of the architecture. This could also mean the roles involved in MEMS are more technical, requiring architects and designers to have considerable knowledge and experience of using middleware [5].

Key scenarios that describe the behavior of middleware architecture with respect to particular quality attributes drive MEMS. MEMS was discussed in more details in [1]. In general, the MEMS evaluation process can be divided into two stages. The first stage is the development of the evaluation plan. This stage encompasses the following steps:

- Determine critical quality attributes
- Identify key architecture patterns
- Develop key scenarios
- Define metrics for each quality attribute.

The second stage of the evaluation process involves the last three steps of MEMS, namely prototyping, carrying out the experiments, and analyzing the measurements.

2.2 MEMS – Analysis of Measurements

System performance evaluation and scalability are considered quantitative attributes. In our previous work [2], experiments are designed with incremental complexity where the input parameters to the systems are varied to show the system performance and scalability. The evaluation result is drawn on the CPU utilization plot based on the result data in order to interpret system performance and scalability [2].

The response times of each measureable component are also collected for analysis (For example *Track_Writer*, *Track_Updater* and *Track_Manager* components). The influence of different number of tracks to the response time of any of these components is important for the overall middleware architecture evaluation, it provides insights into the performance and scalability of each of these components.

The response time of these components will also be influenced by external factors such as operating system, concurrent processes and networking infrastructures that are beyond the scope of middleware architecture evaluation. Our resultant datasets are contaminated with the influence of these external factors, which in return directly affects the overall assessment.

The proposed statistical analysis method in this paper is to serve the purpose of isolating these abnormal data points within a given dataset using statistics, and to provide a statistical assessment of each system component under evaluation. Thus adds an important measurement mechanism to the existing MEMS approach.

2.3 Statistical Process Control and Architecture Evaluation

The goal is to devise an effective statistical analysis approach and supporting tools to observe the relationships among various factors gauged or measured in a series of testing experiments. One of the characteristics of the runtime dataset collected from the test bed is that each experiment produces a large amount of data. Large datasets may be affected by random events that are not expected but will have an impact on the overall result.

The main statistic techniques we have applied in the experiments were inspired by the notion of Statistical Process Control (SPC) and its Control Chart invented by Walter A. Shewhart from Bell Labs in the 1920s [6]. In Shewhart's study, the company engineers had been looking at ways to improve the reliability of their telephony transmission systems. They realized the importance of reducing variation in a manufacturing

process to improve quality of products and services. Shewhart observed manufacturing data displays variation in the processes. Some display controlled variation that is natural to the process, while others display uncontrolled variation that is not present in the process causal system at all times, these uncontrolled variation are characteristics of natural occurring events [6]. The statistical process control chart was then developed to show this effect. A control chart typically consists of the following components [7]:

- Samples of observations taken from the process at different times.
- A centre line, drawn at the process characteristic mean (average) that is calculated from the observed samples.
- Upper and Lower Confidence Limits (UCL, LCL) that indicate the threshold at which the process output is considered statistically "unlikely".

In statistics, a confidence limit is an interval estimate of a population parameter. Instead of estimating the parameter by a single-point value, an interval likely to include the parameter is given. Thus, confidence limits are commonly used to indicate the reliability of an estimate. The confidence value is determined by the selected confidence level or confidence coefficient, in terms of the amount of deviation from the mean [8].

These aforementioned characteristics of a typical control chart are illustrated in Figure 1. The purpose of control charts used in SPC is to allow simple detection of events that are indicative of actual process change in the system. The control chart provides statistically objective criteria of changes, which is an effective measure of dynamic system interactions. The control chart is a simple yet sophisticated approach to identify the behavior a complex system. Based on the test-bed runtime information during execution, engineers are able to classify whether the system performance is acceptable or unacceptable based on empirical observations and analysis results indicated by this approach.

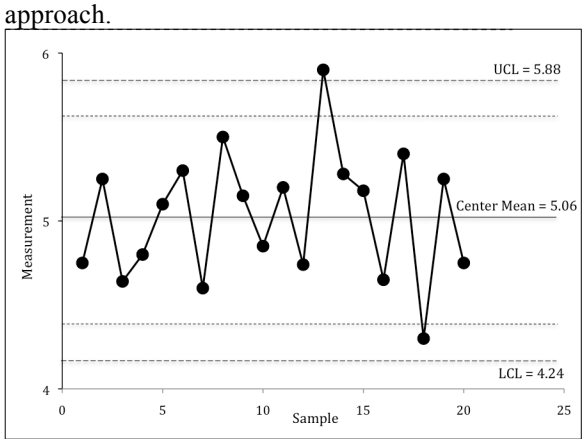

Figure 1 A Sample Control Chart of SPC

We apply the same technique to our proposed measurement analysis method for MEMS with a modified approach to better suit our circumstance.

3. Sensitivity Analysis

The proposed statistical analysis techniques used in this research are to provide a quantitative analysis method to detect and to evaluate its system stability by combining the ideas of SPC and sensitivity analysis in statistics. Statisticians are well aware of the danger of abnormal data points creating spurious relationship in datasets and have introduced a number of techniques (broadly described as sensitivity analysis), to investigate the extent to which statistical results could be an artifact of particular data points. This creates a leverage effect, where one or more data points would ultimately distort the resulting analysis and the result. One approach is to identify these "high leverage" data points. Leverage is found during analysis of modeling results, by exploring positive or negative behaviors, looking for sources of imbalance that cause the model to change significantly.

In this case, the leverage occurs when an abnormal data point will cause the overall distortion of the modeling result for the software component being evaluated. High leverage data points have a large impact on the results of the analysis. It is crucial to investigate whether these high leverage data points will eventually affect the overall system performance within an acceptable tolerance.

Sensitivity analysis involves assessing whether relationships are stable to the removal of high leverage data points from the dataset. In this Section, we explain and propose a suitable sensitivity analysis technique for the evaluation of the experimental dataset.

3.1 Statistical Assumption

In the context of this study, the experiment is carried out based on the runtime data produced on the software system managing tracks in the airborne mission system. In avionics, each "track" represents vehicular metal objects such as an aircraft in the radar-covered aerospace. The radars provide their airborne mission system with tracking reports, which are detections of the targets or aircraft tracks. The software system is responsible to manage these tracks and displays them on screens.

The dependent variable is the response time of a track actually used with respect to the amount of input tracks. The assumption in here is that the dependent variable response time has a linear relationship with the independent variable *track*, and therefore a

predictable relationship should be observed if the system is stable or its response time is predictable. In statistics, a system is considered "stable" when it has the least variance between individual observations (data points). That is where the data shows least amount of variation or fluctuation, where a predictable relationship is existed and can be modeled accurately. In contrast, if a system is "unpredictable" or has a large variance in its data, then there is no strong linear relationship between them. This provides an opportunity to use a suitable sensitivity analysis to assess the abnormal data points that provides a method to assess its system behavior and its predictability. For this particular case, we assume the software system to be stable when the response time required completing *n* repetitive or routine tasks shows a linear or predictable pattern.

The preliminary assumption for this analysis is that the variation of the data points within a given dataset should be within a tolerable range. The tolerable range is arbitrary, defined according to the required level of stability of the system. For example, we assume 10% delays in response time are tolerable for the system.

3.2 Leverage Statistics

To identify and isolate abnormal data points that cause significant influence to the interpretation of the measurement analysis, we developed a Leverage Metric (*LM*) to support our statistical analysis. Similar to that of a control chart of SPC, in this case, data points outside the acceptable ranges will be removed.

LM is calculated based on the residual of each data point to the fitted regression line of the dataset. This indicates the extent to which the trend of the data is influenced by each individual case. Similar techniques have been proven and used in other areas of software engineering, such as in software project cost estimation. It has been used to preprocess datasets before the application of cost estimation, Keung et al. has provided a more comprehensive discussion in [9].

To calculate LM_i for each case i, let $E(X_i)$ be the expected value for the data point X_i, which is the predicted value of case i based on the regression model of the dataset, X_i is the actual observed value from the experiments. LM_i is then given by:

$$LM_i = X_i - E(X_i) \quad (1)$$

LM_i is the difference (residual) between the observed data point X_i and its expected value $E(X_i)$ based on all the data points excluding data point X_i, indicating the impact of the specific case i on the entire dataset. The expected value $E(X_i)$ is a predictive function based on all data excluding X_i to predict the outcome of X_i.

Under the null hypothesis that case i is NOT abnormal, X_i will be an unbiased estimator of $E(X_i)$ and will be approximately normal $X_i \sim N(0,S^2)$. For the purpose of abnormality detection, the following z test provides a mechanism to formally verify whether the value of X_i is an abnormal one. For each case i, LM_i can be converted to its standard normal form as below:

$$Z_i = \frac{LM_i}{S} \quad (2)$$

The Z_i value provides an indication of the confidence level of a particular data point being abnormal by using the standard empirical rule [8]. The empirical rule states that for a normal distribution, almost all values lie within 3 standard deviations of the mean, where about 68% of the values lie within 1 standard deviation of the mean, or about 95% of the values lie within 2 standard deviations of the mean, or 99.7% of the values lie within 3 standard deviations of the mean. In this case, the mean is the expected value of X_i derived from the fitted linear regression model.

Consider that there is a normal distribution curve applied on each single data point X_i to evaluate the expected value $E(X_i)$. The following illustrates the application of our leverage metrics approach utilizing normal distribution on each single data point.

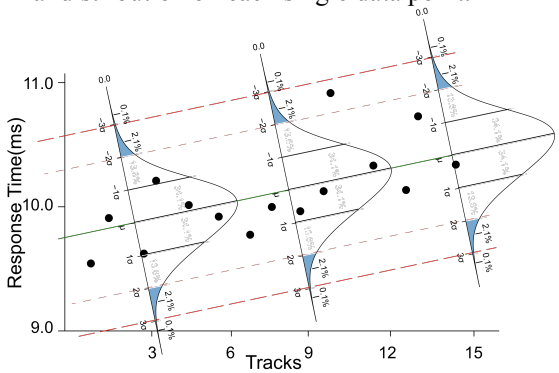

Figure 2 Example: Linear modelling with normal distribution

In Figure 2, the expected values based on linear regression should be within the acceptable limits of UCL (Upper Confidence Limit) and LCL (Lower Confidence Limit), denoted by dashed lines. Any abnormal data points outside these boundaries can be considered unusual and abnormal. This can be statistically revealed using Equation 2: If $|Z_i|$ is greater than 2 standard deviation then the implication is that the data point is significantly different from its expected/predicted value at the 0.05 significance level (95% approximately).

It is therefore important to decide the threshold value for Z_i. Obviously this depends on the required level of precision. Confidence intervals at 95% are typically used in practice. We propose the use of $|Z_i| >$

2 shall be adequate in this circumstance. Any abnormal data points outside this threshold can be easily identified. However, other threshold values can also be used depending on different precision requirements, for example at 68%, 95% and 99% significance with threshold values of 1, 2 and 3, respectively.

The larger the number of abnormal data points in a given dataset, the less stable the system. Based on this principle we are able to derive a simple metric to assess the risk component of the system under evaluation. For example, given 1,000 observations, there are 25 data points that have been identified being abnormal or unusual with a confidence level of 0.05. It can be concluded that the system is at a stability level of 97.5% (1-0.025) if we accept a tolerance level of 5% error. The stability metric *K* can be expressed as:

$$K = 1 - \frac{A}{n} \tag{3}$$

where *A* is the number of abnormal data points, and *n* is the sample size of the dataset. This provides a rationalized measure of the number of unusual and unpredictable events that may occur at any given time. *K* can also be considered as a predictability metric where the system response can be predicted.

4. Equilibrium Statistics

In science, the definition of equilibrium is the condition of a system in which competing influences are balanced. In this case, the main influence is the fluctuation of the observed data points. There are two major measures against the dataset, the mean average and the modeling of the best-fit model to determine its trend. The assumption in here is that a system equilibrium point can be observed based on the intersection of the mean line and the fitted regression line. The required runtime performance can be ascertained with a given track number as input below the system equilibrium value.

The base line is derived based on the expected value of the experiment. Using probability theory, the expected value (or mean) of a discrete random variable is the sum of the probability of each possible outcome of the experiment, multiplied by the outcome value. Thus, it represents the average one "expects" as the outcome of the trial. In this case[8]:

$$E(X) = \frac{\sum x_i}{n} \tag{4}$$

The fitted regression line is a linear combination of the model parameters and depends on actual observed values from the experiment. The regression model can be used to predict expected values based on the overall distribution of the dataset. The linear regression model represents a straight-line or a trend and can be written as [8]:

$$y_i = \beta_1 + x_i\beta_2 + \epsilon_i \tag{5}$$

where β_1 is the intercept, β_2 is the parameter estimate, x_i the regression coefficient, and epsilon is an observation error. An equivalent formulation of the above regression equation that explicitly shows the linear regression as a model of conditional expectation can be given as [8]:

$$E(y|x) = \alpha + \beta x \tag{6}$$

In this case, the equilibrium point can be observed when the two expected values met where the expected value by mean is equal to the expected value by the observed linear regression model. The formulation can be rearranged as follows to compute their intersection points (*x,y*), where:

$$\begin{aligned} y = E(X) &= \frac{\sum x_i}{n} \\ \alpha + \beta x = E(X) &= \frac{\sum x_i}{n} \\ \beta x &= E(X) - \alpha \\ x &= \frac{E(X) - \alpha}{\beta} \end{aligned} \tag{7}$$

5. Experiment

In the experiment, test scenarios are designed in incremental cases where individual operations are tested first with different workload levels, and then the experiments with combination of the operations are observed. This incremental approach allows us, firstly, to identify the performance characteristics of individual operations to pinpoint possible defective operations of the software components. Secondly, understanding the performance and scalability aspects of individual operations can also provide insights more realistic scenarios when operations are combined, especially the observation if the performance behavior patterns are consistent across different scenarios.

The following (Table 1) shows scenarios that are grouped according to their respective operations and given workloads. It also shows the total number of tracks, the number of track writers and the rate that the track writers generating loads as parameters controlled in the experiments. Each scenario now has multiple sub-experiments when the values of the parameters change.

Task	Scenario	# Tracks	# Writers	Rate/sec
Create	1	1000	1	1,10,50
	2	1000	5,10,20,50	50
Update	3	1000	1	1,10,50
	4	1000	5,10,20,50	50
Display	5	1000	N/A	50
Combining (**C**+**U**+**D**)	6	1000	1	10,25,50
	7	2000	1	10,25,50

Table 1 Test scenarios

Each scenario produces a large amount of data for data analysis. To facilitate this process we employ R-Project [10] and construct a script to compute and produce results in the experiment.

In addition, extra experiments are designed and carried out to at different times of a day to examine if the system is capable to produce similar results. This will reveal whether the system under evaluation is influenced by external factors, which are not internally assessable.

For example, such factors may include but are not limited to

- External software programs running simultaneously
- Hardware resource sharing
- Network resource sharing
- Other unknown factors

The system under investigation, especially the COTS software framework, may also contribute to the stability issue. Given all above factors, the experiments intend to first isolate the external factors with the internal factors.

6. Results

The system has been executed in the MEMS test-bed environment with the inputs shown in Table 1. It is noteworthy that the data analysis produces two sets of data, (1) the dataset without outliers and (2) the dataset with outliers. There are implications within these two datasets. The datasets with outliers can be used to assess insights of the system's behavior on performance and scalability. Outliers are indicators of potential influential factors that cause performance and scalability degradation, as well as stability of the entire system. It can be used to discover the worst-case scenarios if the testing parameters are gauged according to a given test-bed environment. The data without outliers are nominal and can be used to model the system performance under a stable condition.

The following selected results are used to illustrate the effectiveness of our method for the system under evaluation.

6.1 Detecting abnormal data points

Figure 3 is generated using R-Project, illustrating the effect of before and after abnormal data points are removed. In this case, there are 1,000 tracks, 20 writers at a rate of 50 tracks per second, that is 1,000 × 20 = 20,000 tracks or observations produced. On the top graph of Figure 3, we can see that there are a large number of outlying data points deviated from the main regression line. In this case the range of response time is between 115 and 124,125 microseconds.

We apply our leverage statistic method discussed in Section 3.2 and calculated Z_i for each i^{th} data point. Our sensitivity analysis has successfully identified 128 data points having a Z_i value greater than 2 standard deviations or 95% significance. The removal of these outlying data points stabilized the dataset and significantly changed the dataset range from ($15 < X < 124,125$) to ($115 < X < 8,4,21$). The formation of a linear trend is clearly appearing in the bottom graph of Figure 3 after abnormal data points are removed.

Using Equation 3, we are able to determine the system stability. Given that 128 observed abnormal data points, we are able to predict that the *update_or_add_track* operation in the system is stable, where $K = 0.9936$ or 99.36% according to the actual captured data used in the experiment.

6.2 Equilibrium Point

We apply our equilibrium statistics using Equations 4, 5, 6 and 7 on the selected scenarios illustrated in Figure 3 and Figure 4. The intersection point of the mean and observed linear regression model is highlighted using a red circle indicating the capacity of the system that can be ascertained at any time.

The bottom graph of Figure 3 shows the equilibrium point is located near the centre of the graph, at 10,950 tracks after 128 abnormal data points being removed and remodeled. There is a slight shift of the equilibrium point from 10,080 to 10,950, caused by the influence of these abnormal data points. This clearly indicates the effects of these outlying data points on the measurement analysis.

A further examination of our methods using the same dataset but on a much smaller scale (5,000 tracks) is shown in Figure 4. The experiment re-executed separately, where only 5,000 tracks are used as input in this scenario. Combining our sensitivity analysis and the equilibrium point technique, we are able to first isolate 42 out of 5,000 abnormal data points in this case. Before the removal of 42 data points causing heavy influences on the dataset, we observed the equilibrium point lies at 2,510 tracks on the top graph of Figure 4. We observed a more

stabilized relationship after 42 influential points were removed from the system. The equilibrium point is shifted towards 5,000 after the abnormal points were removed. Thus again indicates the importance of the sensitivity analysis used in the application of this analysis.

In the second experiment, the implication is that given 5,000 tracks the system did not reach its true capability, whereas in the first experiment the capacity of the system is clearly given at 10,950 tracks. This suggests that a larger sample size is needed for the second experiment.

Repeated experimentation also confirmed the variation between different runs are insignificant. In addition to the K=0.9936 computed in the previous section, we are confident that the system is 99.36% stable with a guaranteed capacity of 10,950 tracks for the operation of *update_or_add_track*.

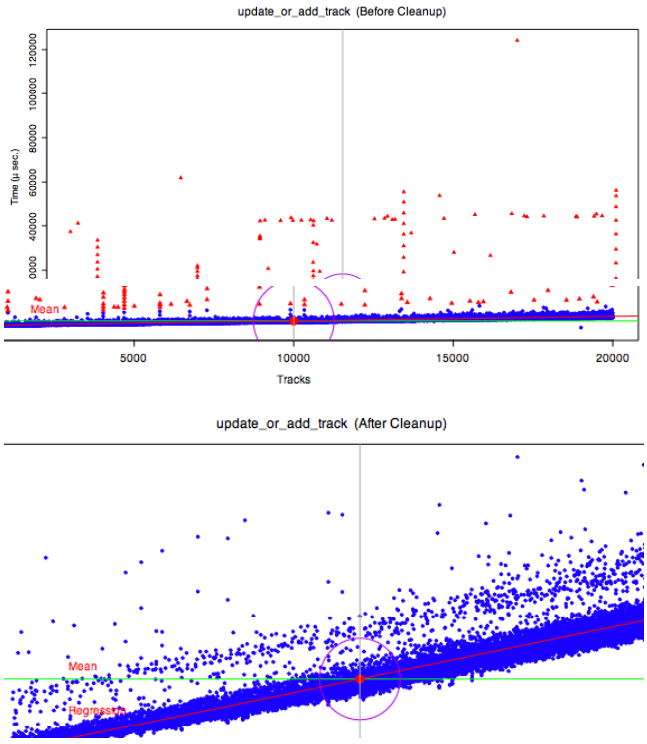

Figure 3 Observations of Scenario 2 Experiment 3 with 20,000 Writer tracks

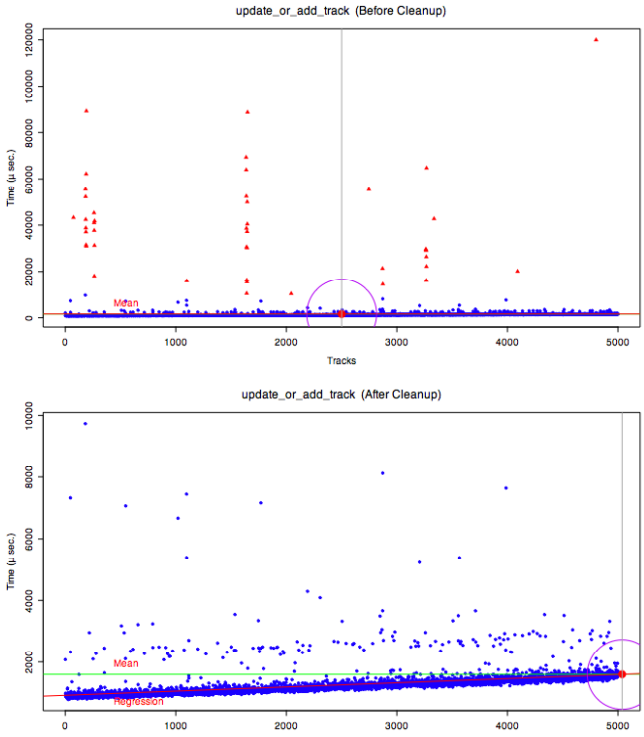

Figure 4 Observations of Scenario 2 Experiment 1 with 5,000 UpdateOrAdd tracks

7. Discussion

The techniques facilitate the analysis of the empirical results in the study in two aspects. Firstly, it helps to identify the outliers, which indicate abnormal responses of the system that may be caused by external factors, allowing system engineers to focus on the reduction of these abnormal behaviors in the enhancement of the system. Secondly, it provides a statistical indication of the equilibrium point, showing the capacity with which the system can handle required tasks within a reasonable time

We selected a sample output result in Section 6 to show the methods proposed in the study. Other results derived are not illustrated in this paper due to space limitations. However, they have very similar characteristics to those of the samples illustrated in Section 6. A certain data pattern has been found across different experiments:

- The density of the outliers increases as the number of tracks increases; or, equivalently, as the rate of track updates increases.
- The time based metrics follow linear trends.

These observations provide some indications of performance and scalability:

- The performance and scalability of the systems are sensitive to the workload beyond a certain threshold. Estimation and prediction of the system's capacity in handling the workload is very useful to optimize performance and scalability.
- Outliers not only indicate the sensitivity of the system under test but also affect the expected average value of a metric.

The novel statistical approach we devised in Sections 3 and 4 are very useful in analyzing outliers' patterns and their impact. As shown in Figure 3 and Figure 4 the crossing of two lines mean and regression implies that, given the expected value of a metric, the x-axis value at the crossing point is the capacity of the system. For example, in the bottom graph of Figure 3, the crossing point means given an expected mean value of latency of 2,600 microseconds, the maximum capacity of the system is 10,950 tracks, while before abnormal data points removal (top graph of Figure 3), the maximum capacity of the systems falls to 10,080 tracks with a mean value of latency of 5,010 microseconds.

The difference between the capacity levels with and without outliers establishes the threshold boundary of the testing scenario. In this example, the capacity of the systems is approximately [10,080, 10,950] tracks with average latency of approximately [2,600, 5,010] microseconds.

The equilibrium point is also useful and practical for determining when the system is overloaded. When an analysis diagram can be produced, the equilibrium point is obtained by moving the line of expected value against the regression line. When two lines intersect, the equilibrium points can be identified.

Based on the system stability metric K (Equation 3), we are able to statistically determine the overall performance of the system. In our experiment, we demonstrated that it is possible to use K as an indication to show that the software component under evaluation is stable at 99.36% of the time.

8. Conclusion

The proposed approach is a set of comprehensive metrics that leverages statistical techniques and is integrated with the testing experiment design, measurements and results analysis. Our analysis method provides insights in the design and development of mission critical systems to satisfy high dependability requirements. It enables to accurately pinpoint bottleneck components in the architecture built from COTS middleware framework. The rigorous statistical analysis provided also demonstrates the trends of the system under different workload conditions.

The basic assumption underlying our approach is that based on the observed data from the experiment, the actual observed data points should not deviate significantly from their respective expected values. If they do, it is likely that this is due to unforeseeable events or factors, such as resources sharing and other external factors. These influential points cause delay in response time of the system and will distort the stability of the dataset and hence the model for predicting its performance behaviors. Therefore these points must be isolated using our sensitivity analysis or spurious statistics may be introduced which will distort the result. We have successfully demonstrated these effects in the experiments.

In summary, the contributions of this study are the development of the following statistical method for the measurement and analysis stage of MEMS:

- A Leverage Metric to remove abnormal data points
- A System Stability Metric K to evaluate overall stability of the system
- An Equilibrium Point to capture the capacity of the system under normal condition.

The integration of these metrics for the analysis is thus a robust solution that provides a sound statistical basis for MEMS measurement and analysis on the large amount of runtime performance data. This is a major advancement and completes the missing measurement analysis required for the MEMS approach for evaluating middleware architectures.

9. Acknowledgement

NICTA (National ICT Australia Ltd.) is funded through the Australian Government's Backing Australia's Ability Initiative, in part through the Australian Research Council.

10. References

[1] Y. Liu, I. Gorton, C. Hoang, and S. Abanmi, "MEMS: A method for evaluating middle architectures," in *International Conference on Quality of Software Architecture (QoSA)*: IEEE Computer Society, 2006, pp. 9-26.

[2] Y. Liu, K. Foster, T. Nguyen, and J. W. Keung, "Quality Assessment of Mission Critical Middleware System Using MEMS," in *International Conference on Quality Software (QSIC)*, Jeju, Korea, 2009.

[3] M. A. Barbar and I. Gorton, "Comparison of scenario-based software architecture evaluation methods," in *Asia Pacific Software Engineering Conference (APSEC)*, Washington DC USA, 2004, pp. 600-607.

[4] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*: Addison-Wesley, 2003.

[5] I. Gorton, A. Liu, and P. Brebner, "Rigorous evaluation of COTS middleware technology," *Computer,* 36 (3) pp. 50-55, 2003.

[6] W. A. Shewhart, *Economic control of quality of manufactured product*: D. Van Nostrand Company, 1931.

[7] J. S. Oakland, *Statistical Process Control*. Oxford: Butterworth-Heinemann, 2003.

[8] A. Selvanatha, S. Selvanatha, G. Keller, B. Warrack, and H. Bartel, *Australia Business Statistics*. Sydney: Thomson Publishing, 1994.

[9] J. Keung, B. Kitchenham, and R. Jeffery, "Analogy-X: Providing Statistical Inferences to Analogy-based Software Cost Estimation," *IEEE Transactions on Software Engineering,* 34 (4) pp. 471-484, 2008.

[10] R. Project, "The R Project for Statistical Computing," 2009.