A Large Scale Empirical Study on User-Centric Performance Analysis

Shahed Zaman, Bram Adams, Ahmed E. Hassan
Software Analysis & Intelligence Lab, School of Computing
Queen's University, Canada
{*zaman,bram,ahmed*}*@cs.queensu.ca*

***Abstract*—Measuring the software performance under load is an important task in both test and production of a software development. In large scale systems, a large amount of metrics and usage logs are analyzed to measure the performance of the software. Most of these metrics are analyzed by aggregating across all users to get general results for the scenario, i.e., how individual users have perceived the performance is typically not considered in software performance research and practice. To analyze a software's performance, user's perception of software performance metrics should be considered along with the scenario-centric perspective of system tester or operator.**

In our empirical study, we analyzed the impact of performance on individual users to see if performance analysis results based on the user's perception is really different from the scenario-centric (aggregated) one. Case studies on common use case scenarios in two commercial large telecommunication systems and one open source performance benchmark show scenarios where user-centric software performance analysis was able to identify performance issues that would be invisible in a scenario-centric analysis. We find that the user-centric approach does not replace the existing scenario-centric performance analysis approaches, but complements them by identifying more performance issues.

***Keywords*-Software performance, user, metric, load test.**

I. Introduction

Measuring the software performance perceived by the user during test or production of a software allows an organization to adjust the deployment of a software system in order to meet the performance expectations of its users [1], [2]. Since performance measurements gather gigabytes of log and performance counter data, performance analysts typically aggregate this data across all user, or clusters of users with the same quality of service (QoS), to get more manageable information. For example, a system is analyzed to see if 99% of the response time for a test scenario was under 2 milliseconds.

Aggregation of performance data loses knowledge about how individual users are affected by the performance. A large software system may have a low average system response time, which suggests that everyone is happy, while from the user perspective, it may be possible that some thousands (out of millions) of users are always unhappy because of persistent high response time (i.e., because of software compatibility issues at the user's end). In a competitive market with other options for the user, user perceived quality is important for user satisfaction and user loyalty to the service provider [3]. Moreover, if these unlucky users are the major users, it can be disastrous for a business.

User perceived quality has been studied in the field of software system design, usability analysis, product and process quality [4], [5], [6], [7], [8]. The user's perspective of quality can be different from the developer or manager's aggregated perspective. Performance can be viewed as one of the characteristics of software quality [9].

We compared individual user performance data (user-centric) with the aggregated performance data (scenario-centric) to see whether or not the user-centric performance analysis data provide the same information as the scenario-centric data. In our controlled case study of two large enterprise systems, and one small open-source e-commerce system's test data (with simulated load), we calculate scenario-centric and user-centric metrics to address the following three research question:

Q1) *How does the user-centric performance experience differ from the scenario-centric one?*
One of the enterprise systems showed major performance improvements for 2 (out of 4) use case scenarios while the scenario-centric comparison could not identify these improvements. In the open source e-commerce system, user-centric analysis in 3 (out of 3) scenarios showed that 40% of the users in each scenario had a bad performance in at least 30% of their requested operations, while the scenario-centric analysis showed that on average, 13.11% to 20.80% of these operations behaved badly in these three scenarios.

Q2) *How does the user and scenario-centric performance evolve over time?*
When a system is running for a long time, it may encounter performance problems that could not be visible if it was running for a short amount of time. 2 (out of 4) and 5 (out of 6) scenarios in the two enterprise software systems and 1 (out of 3) scenarios in the open source system showed a different time trend from the scenario and user perspective. These differences suggest that scenario-centric performance trends do not necessarily reflect the performance trends perceived by user.

Q3) *How consistent are the user and scenario-centric performance characteristics?*
A system may sometimes perform outstandingly good

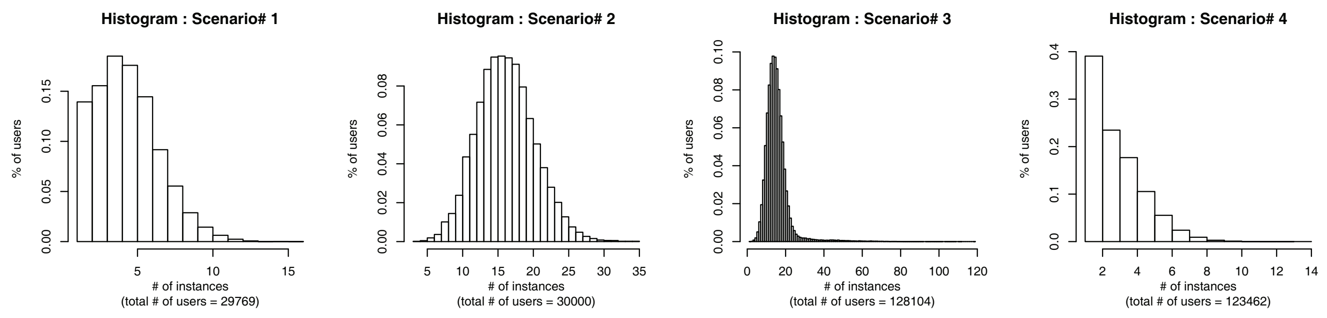

Figure 1. Enterprise System 1 - Histogram of number of scenario instances per user

and sometimes very poorly while the users in the system may expect a moderate and consistent performance from the system all of the time. From user-centric analysis, we found scenarios where a large percentage of users had an inconsistent performance while the system's performance looked consistent in scenario-centric analysis (on average).

The paper is organized as follows. Section II, discusses the study design and data source for our study. Section III explains the motivation, approach and results for our three research questions. Section IV summarizes the related work. Section V discusses the threats to validity. Section VI presents the conclusion of the paper.

II. Study Design

In this section, we discuss our three case studies in which we performed both user-centric and scenario-centric performance analysis. User-centric analysis measures the metrics from the individual user's perspective and scenario-centric analysis measures the metrics for a scenario across all users. Table I shows an overview of the three projects in our study, i.e., two commercial enterprise systems (ES) and one open source e-commerce system, the Dell DVD store (DS2). For each system, we considered the most popular use cases. For each instance of such a use case, we collected the response time (i.e., the time between start and end) and user id. All the metrics used in our analysis are derived from this response time and user id data.

Table I
Properties of three case studies

Factor	Enterprise System 1	Enterprise System 2	Dell DVD store
Functionality	Telecommunications		E-commerce
Vendor's Business Model	Commercial		Open-source
Size	Ultra Large	Large	Small
Complexity	Complex	Complex	Simple

Enterprise System 1 (ES 1): The first system for our case study is an ultra large scale distributed system for which we had access to the performance regression test log data. A test was run on two versions of the system with the same test operational profile (i.e., a realistic distribution of user input requests). Any performance deviation found in the new version's performance is reported by the tester to the developer. One hour of test log data was collected. Log lines contain log messages that can be grouped into sequences per user session and mapped to a use case using log sequence patterns known to the system experts [10], [11], [12]. We compare the old and the new version of the ES 1 to verify if the scenario-centric performance analysis differs from the user-centric performance analysis. Out of around 100 scenarios, we considered the four most occurring ones as the four major use-cases of this system. Other scenarios had a too low frequency of occurrence to comment on their performance. Every scenario's first and last log line's time-stamp was used to calculate the response time for that scenario. Ideally, every user in these tests are expected to perceive a similar response time.

Enterprise System 2 (ES 2): The second system in our case study is a large scale enterprise system of which we could use the test log data. The test was run on one version under development and 10 hours of log data was collected and analyzed. We used a similar log sequence analysis approach as we did for ES 1. Out of 1092 scenarios, we considered the top 6 most popular log sequences.

Dell DVD store (DS2): The third case study was performed on the Dell DVD store (DS2). DS2 is an open-source three tier simulation of an e-commerce website [13]. We set up the DS2 test server in a lab environment using two Pentium 3 servers running Windows 2008 and Windows XP with 512MB of RAM. The first machine is the Apache Tomcat web application server [14] and the second machine is the MYSQL 5.5 database server [15]. The load for this test was generated using the DS2 web load driver running on a Pentium 4 machine with 2GB of RAM. The load generator simulates multiple users by sending HTTP requests to the web server for each user, the web server processes the request by communicating to the database server, then replies to the user (load generator) again.

We ran the test for more than 11 hours and monitored the server resource utilization logs to identify the initialization period of the system. We removed the initial 1 hour log data to remove the system initialization performance from our study and concentrated on the normal operation of the system. Details of the Dell DVD store (DS2) load generator configuration can be found in our replication package online [16]. We instrumented the DS2 driver program that generates the load to the web application to create log files containing one line for each user's HTTP request+reply (i.e., for each instance of every use case scenario). As a simple e-commerce system, DS2 has four scenarios: Create new user, Login to the system, Browse products and Purchase. As the scenario 'Create new user' occurred only once for each user, we dropped it from our study.

In total, our study contains 4 use-case scenarios from ES 1 (2 versions), 6 use-case scenarios from ES 2 and 3 use-case scenarios from the Dell DVD store e-commerce system.

III. Case Study Results

For each question, we present the motivation behind the research question, the approach and a discussion comparing the results of the user-centric approach with those of the scenario-centric approach for analysis.

A. How does the user-centric performance experience differ from the scenario-centric one?

Motivation: A user's initial experience to the system (i.e., the experience of the first few operations performed by a user) is very important to the user's confidence in the system's performance and to keep him using the system. In a competitive market, bad experience of usage may cause the user to ask for a refund, look for another new service provider, repeatedly call to system support lines or ask for technical support.

A user's overall experience (i.e., the experience across all operations by a user during our test analysis time frame) explains whether the initial experience was representative for the full experience. For example, if there are some users who have a persistent problem in their network or system configuration, they will always perceive a bad experience. In case of a relatively small number of such unlucky users, the scenario-centric analysis of the system's performance may miss such information, although the performance problem of those users can be important. A good run from scenario-centric perspective may actually be a bad run from the user-centric perspective.

Approach: **a)** For the initial experience, we considered the response time for the first "n" instances of a scenario for each user. The idea behind a metric for initial experience is to see how many users had a bad experience initially. In this paper, we measure bad experience or bad instance of a

Table II
Enterprise system 1 - scenario-centric Comparison of performance

Scenario #		Old	New
1	Mean	227.34	226.1
	Median	204	203
	% of bad instances	12.65	11.32
2	Mean	410.62	425.89
	Median	406	406
	% of bad instances	16.48	0.21
3	Mean	44.82	42.25
	Median	46	32
	% of bad instances	7.19	0.37
4	Mean	8.24	7.14
	Median	0	0
	% of bad instances	5.03	4.64

Table III
DS2 - scenario-centric Comparison of performance

Scenario #	Mean	Median	% of bad instances
Login	545.31	531.25	13.11
Browse	517.05	515.63	12.87
Purchase	6062.84	5984.38	20.8

scenario as an instance with the following delay or response time criteria:

$delay > median(scenario) + standard\ deviation(scenario)$

This threshold uses the median and standard deviation of response time for each scenario.

Out of the "n" initial instances, we measured the percentage of bad instances. We choose the value of "n" to be the maximum of the 10th percentile and 5. For example, in Figure 1, we can see the histogram of the total number of instances of a user for each scenario in ES 1. For scenario 1 and 4, the 10th percentile was 1 and using only the first instance to measure the initial experience would give us only two levels of initial experience, either 0% or 100%. Hence, for these two scenarios, we considered the first 5 instances to measure the initial experience and for scenarios 2 and 3, we considered the first 7 instances (as the 10th percentile value was 7). In ES 2, we considered n=11,11,9,5,5,5 for six scenarios and in DS2, n=5,7,5 for the 3 scenarios.

b) For the user-centric performance measure of overall experience, "n"' represents all instances for a user during the whole period of analysis. For example, if a user had 35 instances during one hour of analysis, we identified the number of bad instances in these 35 instances using the same threshold $delay > median(scenario) + standard\ deviation(scenario)$, then calculated the percentage of bad instances of that user.

c) For the scenario-centric performance analysis, we do

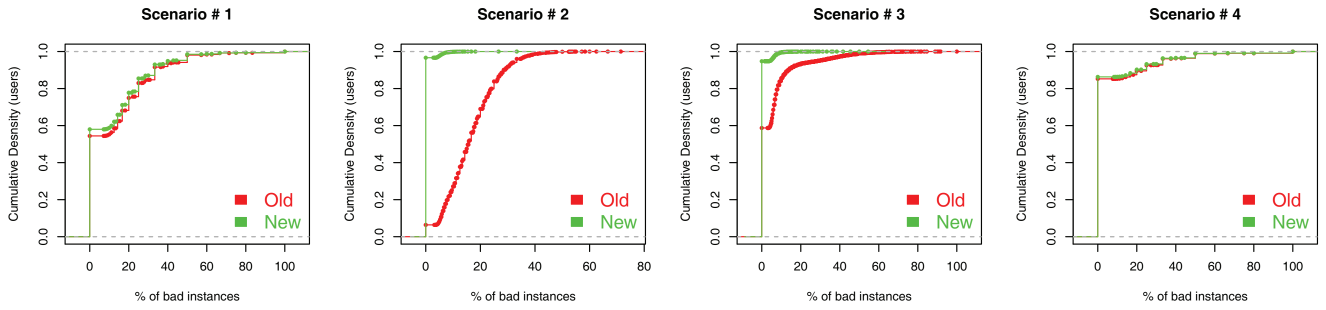

Figure 2. Enterprise System 1 - Cumulative Density Plot of user Overall Experience in % of bad instance

not need to define “n”, since performance trend curves (discussed in subsection III-B) over time can show the scenario-centric initial performance.

d) For the overall scenario-centric performance measure of a scenario, we considered the mean, median and density of the response time, as well as the percentage of bad instances across all users. For example, if a scenario had 50 users and they had 500 instances of a scenario during one hour of analysis, we calculated the mean, median and standard deviation of these 500 instances across all users. The mean value gives us the average across all observation data including outliers, while the median quantifies the central point of the dataset, neutralizing the outliers. We graphically compare the distribution of response time with kernel density plots (regular and cumulative).

With median and standard deviation values, we used the same threshold $delay > median(scenario) + standard\ deviation(scenario)$ to define a bad instance. Finally, we measured the scenario-centric % of bad instances in each scenario (shown in Table II for ES 1 and in Table III for DS2). For scenario-centric comparison of mean response time values between old and new ES1 version, we also used Welch’s statistical t-test to check whether that the mean difference is statistically significant.

Findings - Enterprise System 1: User-centric performance analysis is showing users with serious performance problems while the aggregated scenario-centric performance metrics (mean, median) are reporting no performance problems. The user-centric performance analysis showed major performance improvements for 2 (out of 4) use case scenarios while the scenario-centric analysis showed major performance improvements for 1 use case scenario.

Figure 2 shows the visible difference (between old and new version) in cumulative density plot of the user-specific response time for enterprise system. We are not showing here the initial experience curve for the users as it is similar to Figure 2.

In scenario #2, using median values (scenario-centric, across all users) we find that most of the instances in both versions had the same response time of 406 units indicating no performance deviation between these two versions. Using mean values, we find approximately 3.72% performance degradation (statistically significant) on average in the newer version, while from user-centric analysis, (Figure 2) we found that the newer version of the system performs better than the older version. We found that more than 90% of the users using the older system had at least 1 bad instance (% of bad instances > 0 in Figure 2), while less than 5% users had such bad experience in the new system.

In scenario #3, we can see from Table II that on average and for most of the instances (median), the newer version had a better overall performance (5.73% better on average, statistically significant and 30.43% using median). The user-specific overall experience curve for this scenario also confirms that performance improvement (Figure 2).

Scenario-centric analysis (shown in Table II) shows that, depending on the scenario, the difference between the old and new version was within the range of 0.39% to 16.27%.

Findings - Enterprise System 2: In ES 2, user-centric analysis in 5 (out of 6) scenarios showed that 50% of the users in each scenario had a bad performance in at least 20% of their instances, while the scenario-centric analysis showed 2.09% to 18.35% bad instances across all users. Different perspectives of the system performance are shown by these two approaches. We are not showing the data and curves for ES 2 as these looked similar to the ones shown for ES 1.

Findings - DS2: In DS2, a user-centric analysis for the 3 (out of 3) scenarios showed that 40% of the users in each scenario had a bad performance at least in 30% of their instances (Figure 3), while scenario-centric analysis shows 13.11% to 20.8% bad instances across all users (Table III). Having less than 20% bad instances on average may be acceptable for a scenario, but having 40% bad instances across 30% of the customers may be too high from a user-centric perspective.

Figure 3 shows the cumulative density function (CDF) plot of the user’s overall experience (user-centric analysis result) and Table III shows the mean, median and % of bad instances across all users (scenario-centric analysis result) .

We can see that on average, a login or browse scenario in DS2 takes around 500 milliseconds or 0.5 seconds and

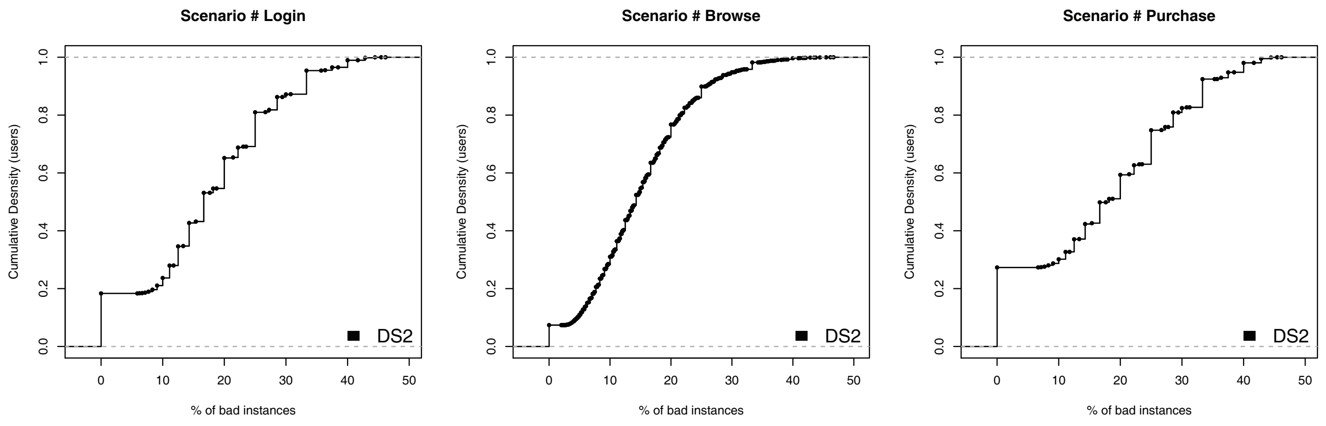

Figure 3. DS2-Cumulative Density Plot of user Overall Experience in % of bad instances

around 12-13% of instances had a response time higher than median + standard deviation. From a scenario-centric perspective, the performance may look good, but when we look at the user-centric view in Figure 3, we find that more than 80% of the users using DS2 had at least one performance deviation. Moreover, we found around 10% of the users with more than 30% of the requests having a performance deviation.

Response time (Table III) and CDF plot (Figure 3) of the purchase scenario show that on average across users, the response time was more than 6 seconds (which is surprisingly high) and similar to the login and browse scenarios, 50% of the users perceived a bad performance in more than 25% of their instances.

User-centric performance analysis shows users with serious performance problems in 10 scenarios (out of 13) that would be invisible from the scenario-centric performance metrics.

B. How does the user and scenario-centric performance evolve over time?

Motivation: Progressive degradation of software performance is a common phenomenon called “software aging” [17]. The measurement, analysis and prediction of system resource usages, response time is done to counteract software aging [18]. Software system performance trends can show how the system's performance changes over time while the system is running. Typically, aggregated summaries of the data (e.g., per minute or week) are plotted over time for system performance analysis. Again, the user aspect is missing in such time trends. A software tester may be interested to know how system performance changes for an individual user with time. For example, it may be possible that in the beginning, the average response time for a user may be higher than later on (because of the initial registration process for each new user or due to cache not being filled up yet), and the tester wants to investigate these. Moreover, it may be possible that from scenario-centric performance trends, a software's performance is steady over time (from mean or median in scenario-centric analysis), while from the user-centric view, as he/she uses the system more and more, the performance actually degrades. The scenario and user-centric perspective of experience over time may give different performance trends over time.

Approach: From the scenario-centric perspective, the time trend of a system's performance is measured by collecting the instance response time and plotting this response time over the instance start/end time. For example, for the duration of our test analysis period, we collected the response time and instance start time for each scenario instance in the system. In a period of 1 hour for ES 1 and 10 hours for ES 2 and DS2, we used “loess” (locally weighted scatter plot smoothing) [19] to get the time trend curve for the system's performance.

For the user-centric time trend, for each scenario, we consider the first instance of each user as the user's time 1, the second instance as time 2, and so on. At each time X, we calculate the mean and median of the response time of all Xth instances. Since in most of the scenarios, only a few users (usually less than 20) have the highest number of scenario instances (for example in Figure 1 for ES 1), we drop data points with less than 100 users in our plots.

Findings - Enterprise System 1: In ES 1, 2 scenarios (out of 4) showed a different time trend when we considered the time from the scenario-centric and user-centric perspectives. Both time trends showed useful information and one cannot be replaced by another.

Figure 4 shows the time trend of scenario-centric system performance and Figure 5 shows the user-centric performance trend. The scenario-centric trend for all scenarios shows that most of the time, the older version had a higher response time, although in the middle of the test run for scenario #1, the newer version had a very large response

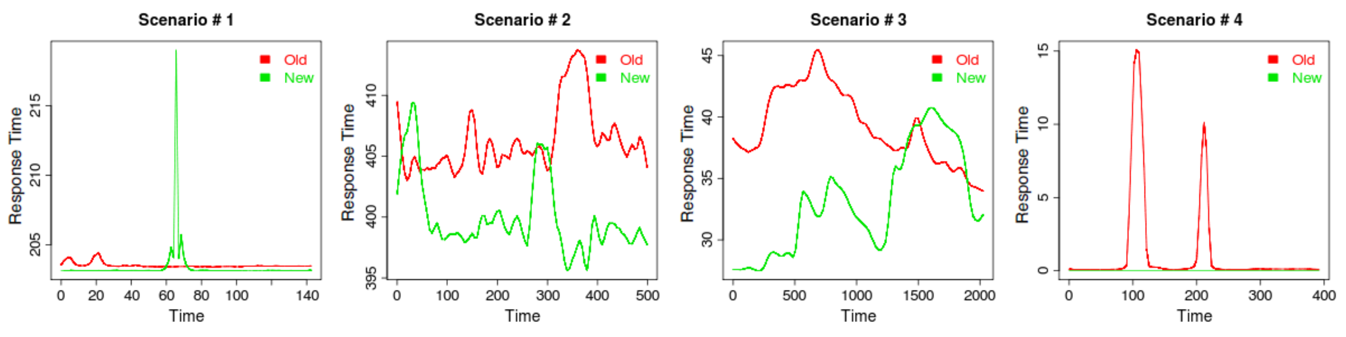

Figure 4. Enterprise System 1 - scenario-centric time trend of the system performance

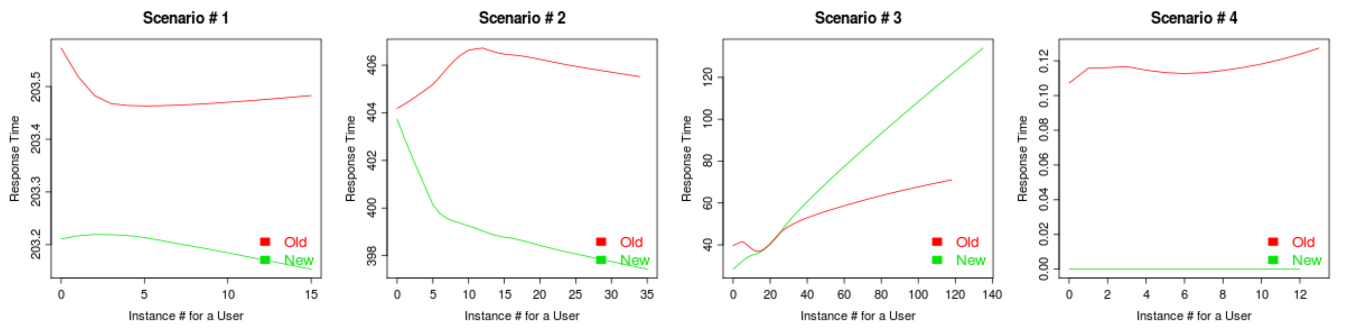

Figure 5. Enterprise System 1 - Time trend showing the user-centric trend of system performance (After using LOWESS smoother)

Table IV
NUMBER OF USERS AND SCENARIO INSTANCES IN EACH SCENARIO

Scenario # in ES 1 (old version)	# of scenario instances	# of users
1	142,528	29,769
2	498,838	30,000
3	2,025,645	128,104
4	391,959	123,462
Scenario # in ES 2	**# of scenario instances**	**# of users**
1	8,387	500
2	8,287	500
3	7,051	500
4	4,616	500
5	4,310	487
6	3,908	500
Scenario in DS2	**# of scenario instances**	**# of users**
Login	138,318	19,984
Browse	418,395	21,385
Purchase	139,733	21,384

peak. From the scenario-centric trend lines, we can identify the time periods during the test data analysis hour when the system performed poorly for each scenario and version.

From the user-centric performance trend lines, we can see that the older version had a higher response time (similar to the scenario-centric trend line), although we could not see any peak as we found in the scenario-centric trend lines of scenario #1.

Moreover, in scenario #3, in the user-centric view (Figure 5) , the users who are involved in more than 30 instances

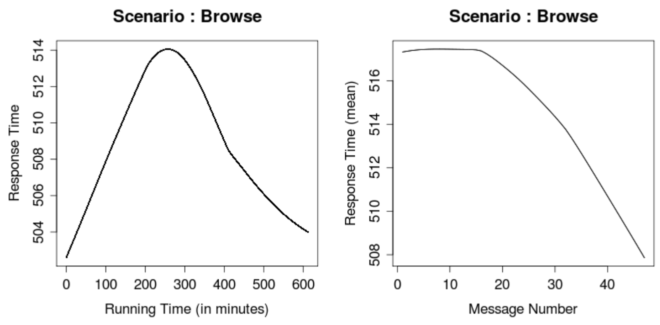

Figure 6. DS2 - Scenario-centric (left) and user-centric (right) time trend of system performance for "browse" response time

(3,409 or 2.66% users), perceived a very bad performance (in comparison to the older version) for those instances after the 30th one. This information could not be observed in scenario-centric analysis.

In scenario #2, for the older version the performance got worse too for users over time, but in the newer version (in green) this problem seems to be fixed. Such a conclusion cannot be made from the scenario # 1 and # 4 curves (Figure 5) as the Y axis scale suggests that the fluctuation was very small in comparison to the median value.

Findings - Enterprise System 2: In ES 2, 5 scenarios (out of 6) showed a completely different time trend when we considered the time from the scenario and user perspective. However, user-centric and scenario-centric trends were complementary both showing useful trends for performance analysis.

Findings - DS2: In DS2, 1 scenario (out of 3) showed a different time trend when we considered the time from scenario-centric and user-centric perspective. Figure 6 shows how the DS2 performance for the browse scenario changes over time. From the scenario-centric trend curve, we can see that the response time increased first, then decreased in this six hour period, while from the user's perspective, we found that as a user browses the DVD store more, his performance to the system improves. The other two scenarios, login and purchase showed similar performance trends in both scenario-centric and user-centric analysis.

The time trends drawn from the scenario and user-centric perspective show different aspects in 8 scenarios (out of 13) of our case study.

C. How consistent are the user-centric and scenario-centric performance characteristics?

Motivation: Performance consistency is an important factor for a user. A user using a system regularly with a slow response time may be used to that slow performance, but if the system performs sometimes very fast and sometimes very slowly, this would show to the user that a better experience is possible and he/she is likely to start demanding such level of services. On the other hand, there may be a case where consistent poor performance may be more expected than inconsistent poor performance. For example, a network service provider may have some users with specific Quality of Service (QoS) requirements that specify that the user needs a consistent but poor performance (relative to the system's performance capability). Preference of consistency is dependent on the system operator's and the user's performance expectation.

Figure 7 shows the simplest possible combination of performance and consistency. For overall experience, a user may have either Good (G) or Bad (B) experience while for performance consistency, a user may have either Consistent (C) or Inconsistent (I) performance. The first square, which represents the users with consistently good experience perceived from the system, is desired by everyone (user and system operator). If it is not possible to have most of the users in square #1, the system operator, depending on his preference, may prefer to have most of the users in one of the other three squares. Intuitively, it seems that any user in the third block (marked as 3), is the most unlucky one. Depending on the QoS offered to the user (if the QoS offered is a slow but steady performance), this can also be the desired performance by the system operator. Research in the field of marketing shows that there may be some rational users who may want to rationally choose a lower quality [20]. For example, some users may be interested in a cheaper service that may not give them the best overall performance (i.e., response time), but still they would want this cheaper performance to be consistent. Such performance and consistency analysis (shown in Figure 7) can identify the users in each of these performance regions.

Approach: To measure the performance consistency over time (scenario-centric), we calculated the median and standard deviation of scenario instances in every one minute interval and plotted the median $\pm$ standard deviation values in every such interval using "loess". For the performance consistency trend over user-perspective of time, we plotted the median $\pm$ standard deviation values for the 1st, 2nd, 3rd, ... nth scenario instances across all users (similar to research question 2). Median $\pm$ standard deviation plots create two lines forming the standard deviation band in which most of the scenario instance delay varies.

To measure the performance consistency, we used variance (which is the square of standard deviation). For scenario-centric variance, we calculated the standard deviation of all instances across all users. For user-centric variance, we calculated the variance of scenario instance response time for each user. To observe the relation between user-centric performance consistency and the user's overall performance experience as described in Figure 7, we plotted the scatter-plot of variance vs overall performance for each user (Figure 10). In the same plot, we added the scenario-centric variance values by using horizontal lines.

Findings - Enterprise System 1: Drawing the performance consistency trends from user perspective or using the overall performance vs consistency scatter plot and dividing it into four boxes, these approaches cannot replace the scenario-centric analysis approach, but can be helpful to better assess the performance of a software system.

Figure 8 shows the scenario-centric performance consistency trend of the system for scenario #2 and #3. We used green for the new version and red for the older version to show the area of standard deviation band. In scenario #2, we see that in the beginning (from X=0 to X=100) the newer version performed more inconsistently than the older version (green region of newer version showing beyond the overlapped region). In scenario #3, we see that the older version performed more inconsistently than the newer version. From the user-centric performance consistency trend in Figure 9, we see the inconsistent performance during initial scenario instances of the newer version. We are not showing in the paper the other two scenarios (#1 and #4) as they did not show any visibly significant difference between scenario-centric and user-centric performance consistency error bands. Both user-centric and scenario-centric consistency trends show useful information and the purpose of one cannot be achieved by the other one.

Figure 10 shows the variance and performance experience of each user as a single semi-transparent dot. In this Figure, darker (green or red) dots and dotted regions corresponding to over-plotting of several semi-transparent dots represent

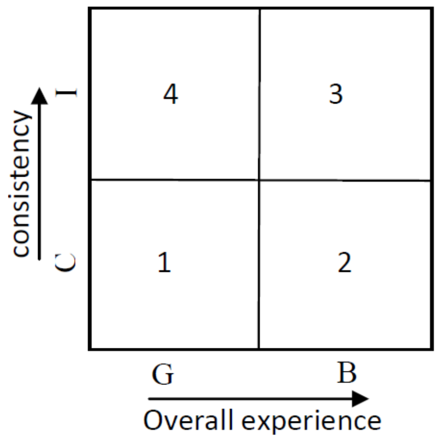

Figure 7. Four options to choose between performance and consistency

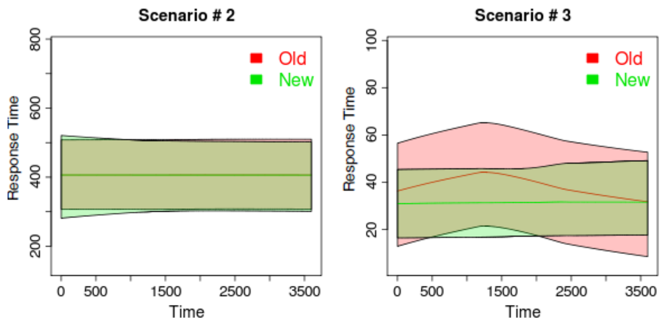

Figure 8. Enterprise System 1 - Scenario-centric performance consistency - showing the error band after using LOWESS smoother

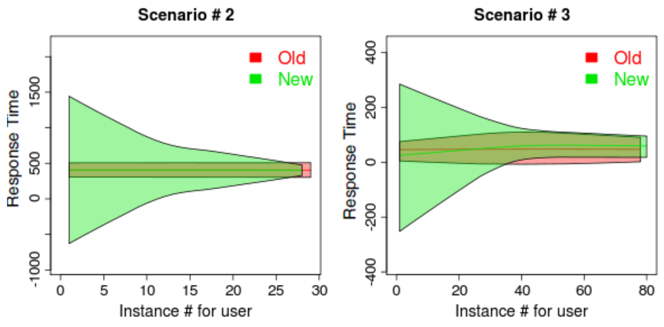

Figure 9. Enterprise System 1 - User-centric performance consistency - showing the error band after using LOWESS smoother

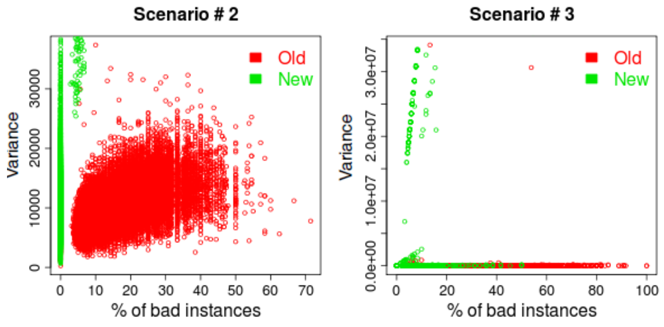

Figure 10. Enterprise System 1 - User's Experience Variance vs Overall Experience scatter plot

the existence of many users in that region. Clearly visible scattered green dots with higher values of variance in scenario #2 and #3 show many users in the newer version with performance inconsistencies. Moreover, most of the green dots (new version) to the left of the plot indicate the lower % of bad instances experienced by the users in the new version.

Depending on the threshold set by the system operator to define good/bad performance and consistent/inconsistent performance, each plot in Figure 10 can be divided and mapped into the four regions as shown in Figure 7. For example, in scenario #2, most of the green dots along the Y axis indicate that most of the users in the newer version are either in square #1 or #4 (most of them had a good experience depending on the threshold). In scenario #3, most of the users using both versions were in either region #1 or #2 (most of them had a consistent experience depending on the threshold).

Findings - Enterprise System 2: The scenario-centric and user-centric time trend of performance inconsistency did not show any difference for ES2. 6 scenarios (out of 6) had a similar error band curve from the scenario and user-centric perspective. More investigation by variance vs % of bad instances showed that most of the users had a consistent performance, although they were scattered into different levels of performance experience (scattered along X axis).

Findings - DS2: The scenario-centric consistency plot for DS2, similar to Figure 8, showed us that the DS2 overall performance variance was consistent over time (two almost straight and parallel lines formed by median $\pm$ standard deviation lines). The user-centric performance consistency plot also showed a similar curve.

However, user-centric analysis of performance variance vs experience (% of bad scenario instance) in Figure 11 shows that for the login and browsing scenarios, the users faced less performance variance than for purchasing (relatively high Y axis value of variance in the purchase scenario than login and browse scenario). In the purchase scenario, a straight dark line along the Y axis represents a large number of users with good but inconsistent performance experience (region #2 in Figure 7).

User-centric analysis showed us performance consistency trends that offered a complementary useful and different view from scenario-centric performance consistency trends.

IV. Related Work

We discuss related work in the areas of software engineering and user-centric analysis. A user oriented approach

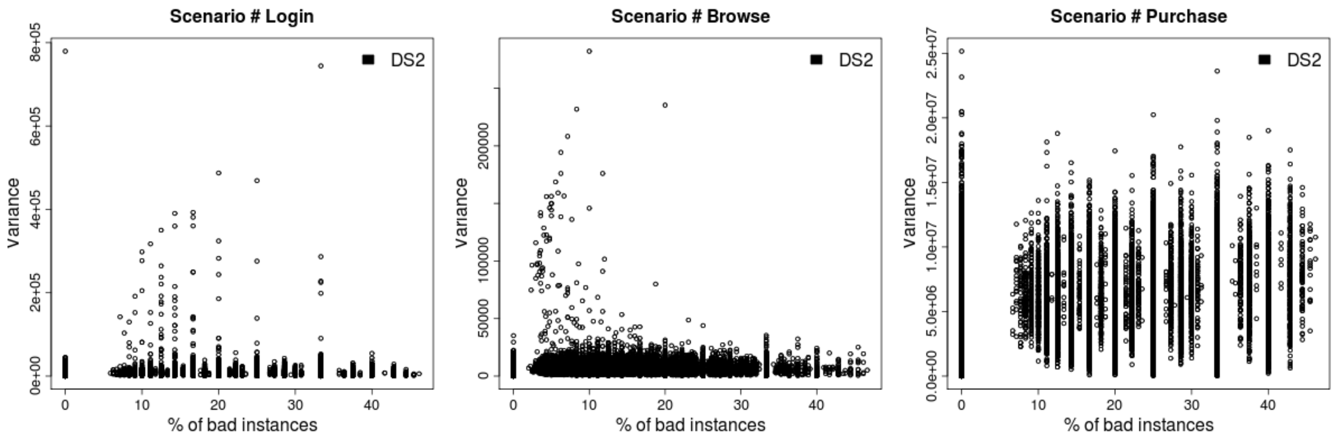

Figure 11. DS2 - User's Performance Experience Variance vs Overall Performance Experience scatter plot

has been considered before for different characteristics of software quality. In the field of software reliability modeling, Cheung described the software reliability from the point of view of a user [21] as "the probability that the program will give the desired output with a typical set of input data from that user environment". He showed that the reliability model can be formulated depending on the user profile (frequency distribution of the use of different features in the system) .

User-centric analysis has also been considered in the field of usability analysis. Terry et al. used end-user oriented analysis on instrumented software to analyze the real-world practices of the users of that software [4]. They collected and analyzed different types of usage data to measure the usability of a software system. Calongne worked on web site and other usability goals [5]. One of those goals for a designer is "high task performance", i.e., the quantifiable speed in which the web page should load and display the requested information given a particular system hardware and software configuration.

The ISO 9126 standard for software product quality has six characteristics that describe product quality [22]. "Efficiency" is the closest characteristic to the definition of performance [9]. This standard describes three set of metrics for software efficiency evaluation, time behavior metrics, resource utilization metrics and efficiency compliance metrics. ISO 9126 also mentions the view of software quality from three perspectives, user, developer and manager. Our empirical study results show the importance of considering the user's perspective along with the other two perspectives for the analysis of these performance related metrics.

Chulani et al. derived a software quality view from customer satisfaction and service data to obtain a better understanding of customer view of software quality [7]. Mockus et al. worked on finding the predictors of customer-perceived software quality [1]. From their study, they identified the factors like software defect reports, requests for assistance and field technician dispatches as the predictor of customer perceived quality for a large telecommunications software system. Mockus et al. also worked on relating the customer-perceived quality to process quality [8]. They developed and evaluated a practical measure of customer perceived quality based on the probability that a customer will observe a problem. The measure is calculated from software problem tracking and customer support systems data. Our analysis shows the importance of such user-centric analysis for software performance, which is a sub-characteristic of software quality.

Gould et al. theoretically and empirically recommended three principles of system design that should be followed to design a useful and easy to use software system [6]. The three recommended principles were, to focus on users and task early in the development process, empirical measurement, and iterative design. They also emphasized on the designers understanding about the users of the system by studying the users directly (by studying their cognitive, behavioral characteristic) or in part (by studying the nature of the work expected to be accomplished). Their research emphasizes on considering the user's perspective from early stages of development life cycle.

V. Threats to validity

Our study was done on three software systems of different scale (ultra large, large and small) and different type (open-source and commercial). To generalize our user-centric performance analysis results, more projects from different software domains should be studied.

This study is based on virtual users that are simulated by software load generators. Although no real users were involved in this study, the load generators used in test environments follow realistic loads, typically sampled from real users.

We used median $\pm$ standard deviation (1 standard deviation) threshold to define a scenario instance response time to be affected by bad performance. Depending on the performance criteria of the system used, this threshold should be adjusted. However, we also checked the median $\pm$ 3 * standard deviation threshold and found similar findings (overall observation, although the percentages and values were different) showing that the user-centric analysis can give us different and complementary useful findings from scenario-centric analysis.

VI. Conclusion

Our goal in this study is to show the importance of user-centric software performance analysis. Researchers in many fields such as marketing, software usability, software reliability have a broad knowledge on how to do user-centric analysis in their field. By the use of these analysis techniques in software performance analysis, we were able to detect performance problems in a software system that could not be detected from existing scenario-centric performance analysis techniques. User-centric analysis could track the percentage of users with very bad experience and identify users with good but very inconsistent performance. However, we also found scenarios where performance peaks were detected by existing scenario-centric analysis, but not by user-centric analysis. Hence, we find the user-centric analysis to be complementary to the existing scenario-centric performance analysis approaches.

VII. Acknowledgement

We would like to thank Research in Motion (RIM) for providing the enterprise data used for this study. The findings and opinions expressed in this paper are those of the authors and do not necessarily represent or reflect those of RIM and/or its subsidiaries and affiliates. Moreover, our results do not in any way reflect the quality of RIM's products.

References

[1] A. Mockus, P. Zhang, and P. L. Li, "Predictors of customer perceived software quality," in *Proceedings of the 27th International Conference on Software engineering, ICSE*, 2005, pp. 225–233.

[2] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, 1st ed. Addison-Wesley Professional, 2010.

[3] P. Chang and H. Chong, "Customer satisfaction and loyalty on service provided by malaysian telecommunication companies," in *Proceedings of the 3rd International Conference on Electrical Engineering and Informatics (ICEEI)*, july 2011, pp. 1 –6.

[4] M. Terry, M. Kay, B. Van Vugt, B. Slack, and T. Park, "Ingimp: introducing instrumentation to an end-user open source application," in *Proceeding of the 26th annual SIGCHI Conference on Human factors in computing systems*, 2008, pp. 607–616.

[5] C. M. Calongne, "Designing for web site usability," *Journal of Computing Sciences in Colleges*, vol. 16, pp. 39–45, March 2001.

[6] J. D. Gould and C. Lewis, "Human-computer interaction," R. M. Baecker, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987, ch. Designing for usability: Key principles and what designers think, pp. 528–539.

[7] S. Chulani, P. Santhanam, D. Moore, B. Leszkowicz, and G. Davidson, "Deriving a software quality view from customer satisfaction and service data," in *Proceedings of European Conference on Metrics and Measurement*, 2001.

[8] A. Mockus and D. Weiss, "Interval quality: relating customer-perceived quality to process quality," in *Proceedings of the 30th International Conference on Software engineering*, ser. ICSE '08, 2008, pp. 723–732.

[9] S. Becker, "Dependability metrics." Springer-Verlag, 2008, ch. Performance-related metrics in the ISO 9126 Standard, pp. 204–206.

[10] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier, "Using magpie for request extraction and workload modelling," in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation*, 2004, pp. 259–272.

[11] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: problem determination in large, dynamic internet services," in *Proceedings of the International Conference on Dependable Systems and Networks*, 2002, pp. 595 – 604.

[12] Z. M. Jiang, A. E. Hassan, G. Hamann, and P. Flora, "Automatic identification of load testing problems," in *Proceedings of the 24th IEEE International Conference on Software Maintenance*, 2008, pp. 307–316.

[13] Dell Inc., "Dell dvd store database test suite," Version 2.1.

[14] The Apache Software Foundation, "Tomcat, 2010," Version 5.5.

[15] MySQL AB, "Mysql community server, 2011," Version 5.5.

[16] "Replication package," http://research.cs.queensu.ca/~zaman/cust-centric-perf-analysis/, November 2011.

[17] D. L. Parnas, "Software aging," in *Proceedings of the 16th International Conference on Software engineering*, ser. ICSE, 1994, pp. 279–287.

[18] M. Grottke, L. Li, K. Vaidyanathan, and K. Trivedi, "Analysis of software aging in a web server," *IEEE Transactions on Reliability*, vol. 55, no. 3, pp. 411 –420, sept. 2006.

[19] W. S. Cleveland, S. J. Devlin, and J. B. Wagenaar, "Locally-weighted regression: An approach to regression analysis by local fitting," *Journal of The American Statistical Association*, 1988.

[20] R. T. Rust, J. J. Inman, J. Jia, and A. Zahorik, "What you don't know about customer-perceived quality: The role of customer expectation distributions," *Marketing Science*, vol. 18, pp. 77–92, 1999.

[21] R. Cheung, "A user-oriented software reliability model," *IEEE Transactions on Software Engineering*, vol. SE-6, no. 2, pp. 118 – 125, march 1980.

[22] ISO 9126:2003, *Software engineering – Product quality*. ISO, Geneva, Switzerland, 2003.