End to End Performance Monitoring Approach

Salah Hussein, Abdel Salam Sayyad
Joint Master Program in Software Engineering (JMSE)
Faculty of Engineering and Technology
Birzeit University
Birzeit, Palestine
E-mail: salah.hussein@gmail.com, asayyad@birzeit.edu

Abstract

Performance monitoring is one of the most important aspects to ensure that software systems operate well. Performance administrators struggle to precisely monitor huge and complex systems with software and hardware components. Achieving this task in a simple and precise way is the goal of this research. To achieve this goal, we need to jump over internal system complexity, and must automate an intensive monitoring process. A simple and accurate approach was developed to monitor systems, all hardware and software components are managed as one abstract box, and then front-end services can be monitored by terms from end to end. The implemented case studies show and confirm that we can emulate manual performance monitoring by simulating the role of real subscriber exactly without any systematic restriction, hence complex algorithms, processes for status prediction and faults diagnosis aren’t farther needed in live system monitoring.

Keywords: performance; monitoring; end-to-end.

1. INTRODUCTION

Performance monitoring in different disciplines is one of the most important roles, and this importance refers to the assessment of objectives and goals. Therefore, business leaders continuously look to monitor main performance indicators, especially ones that affect the top goals.

Performance monitoring in software engineering is one of the most important aspects, because of the vast involvement of software in all aspects nowadays. The performance of software systems affects the goals of other systems directly or indirectly due to high dependency among them.

Performance approaches [1] are divided into early and late cycles, model-based and measurement-based approaches, respectively. On top of those two approaches, we can monitor performance continuously or periodically in discrete periods, may be also in case of acceptance tests or validity test. Alive and continuous monitoring for performance is essential issue for service providers to insure service availability and perceived quality for their customers.

When we talk about software system, we mean huge and complex systems that are based on huge hardware systems such as scalable clusters, load balancers, shared storage, and redundant infrastructure. To monitor a certain business service we may need to monitor many infrastructure services that are interdependent within certain hierarchy, which means very complex algorithms to diagnosis service status at certain moment by using raised faults [13][14][15].

The tested algorithm to investigate the status of such that complex system was through looking from outside these systems, exactly like a client who interacts from end to end, which abstract internal complexity by looking from the end front as a subscriber. This approach can be followed periodically and continuously through suitable and dedicated simulator to interact system or service and then record its status.

We have done two empirical case studies based on this approach for more than two years, with better results than traditional approach [1]. After comparing results with data of standard fault managers and measurement systems for known network management systems (NMS), we found that end-to-end performance monitoring is more accurate, it covers all scenarios that are impossible to be covered by traditional approach [1].

The first case study was on mobile radio service status, the end-to-end methodology was applied, and a manual procedure was followed to check service status from its front end APIs. The results were compared with NMS deducted results, especially from fault manager, measurements, and also from customer complaints, new method introduced more precise and comprehensive results, but with tiny side effects on the running resources.

The second case study was on short message system center (SMSC), it is a very huge system based on a very complex infrastructure and redundant platforms, a dedicated simulator was built to interact with system services from its top interfaces similar to usual subscribers. Collected results reflected a very clear figure about service availability and efficiency for more than one year, these end-to-end results of integrated systems were more comprehensive and realistic than results of synthesis measurements for these systems.

2. RELATED WORK

2.1 Early-cycle (Model-Based) approach

The "Software Performance Engineering" definition is the umbrella of model-based approach, it was pioneered under the name of SPE by Smith [3] [4] (see also [5] for a survey of modelling approaches), which creates performance models early in the development cycle, and then uses quantitative results from these models to adjust its architecture and design to meet purpose of performance requirements.

SPE is a common approach, it is used as predefined model in system design and architecture, it is integrated with measurement-model to fetch data that form adequate indicators about performance, it is usually applied as fault manager subsystem, or software and hardware counters that can be collected and aggregated later as good performance indicators, these counters can be combined with other measurements from outside systems to present obvious figure about system performance.

SPE approach is essential, but it is more general, it can't draw enough figure about behaviour of certain system from inside that system only, and in general it needs human enrolment for monitoring, so it should be integrated in a precise way with other approach from outside the system.

2.2 Late-cycle (Measurement-Based) approach

The commonest approach is purely measurement-based approach, it applies testing, diagnosis and tuning late in the development cycle, when the system under development can be run and measured (see, e.g. [6], [7], [8], and [9]).

The SPE definition is also the umbrella of measurement-based approach, it is mostly integrated with model-based approach to predefine and implement certain dynamic indicators that can be collected and integrated with measurements.

This approach is the most common, but it is designed to be based on model-based approach to be comprehensive, hence model-based is not enough as mentioned above.

Acceptance tests usually depend on both approaches in general, an example of this was mentioned in [2], and research goal was to facilitate acceptance test and performance test, which involved Load Generators that could be deployed for testing systems such as an SMSC or MMSC. The Load Generators enable us to measure how well these servers can handle large numbers of concurrent users, scalability issues, such as response time and processing bottlenecks.

Value-Added Services (VAS) and Content Platforms were carried out within a group of same name at Tele2 AB in Kista, Stockholm. That group was responsible for network design, capacity planning, dimensioning, Acceptance testing (ATP test), and introduces new functionality in Tele2s VAS platforms.

Acceptance testing was performed on new devices (servers and other network components) in order to verify its capacity and performance that guaranteed by their manufactures. Every platform has a guaranteed upper bound performance (based on license of buyer), it can be measured by different approaches. For instance for Short Message Service Center (SMSC) platforms, the measurement was based on the maximum number of SMS messages processed per second (SMS/sec), for Multimedia Messaging Service Center (MMSC) platforms, the metric was the maximum number of MMS messages processed per second (MMS/sec), and for WAP Gateways it was the maximum number of WAP Transactions Per Second (TPS).

The above related [2] work was the nearest to this research, but it was designed to be run for discrete time periods at acceptance test stage only and wasn't designed for continuous monitoring.

3. CASE STUDIES

3.1 Approach: End-to-End monitoring

In this article, case studies are based on the second approach, measurement-based, but with main deference in the way of fetching these measurements about systems from outside system itself, using its front end interfaces, to treat as real client, it exceeds complex diagnosis for deep dependencies and ambiguous faults, and it monitors simply by test and fetching measurements all the time based on predefined pattern. For future work, a dedicated APIs can be designed and included, it will be more suitable for solid monitoring by transients the full life cycle and checks all system processes.

New methodology simulates manual work from the top of service interfaces with continuous pattern to be exactly like human approach, it processes inputs and store results into organized database to be reported in well-designed reports. Yes, this methodology seems trivial and simple to be implemented, but it is very risky in practical mode; this risk appears as side effects on service ability to be continuous all the time, hence case study is important to pilot these issues and confirm method validity.

3.2 First case: Monitoring Mobile Radio Services

One of the most important and essential services in our live nowadays is the mobile services that based on wireless radio signals, all people need this service all the time, any interruption on this service means hazard and big risk, many of the institutions and enterprises are fully depend on that service in different sectors like rescue, emergency, etc. Therefore performance monitoring for such service is very important, and it needs very accurate techniques to notify early and take proper action.

The mentioned approaches [1] are used in the performance monitoring process, early and late approaches, and usually human resource is essential aspect of this process to monitor and follow up model-based outputs, and also to analyze measurements-based outcomes for each day.

Although human resource is essential, but tools and instruments are more important to help in monitoring process and to efficient the work. Engineers monitor alarm viewers all the time and also can analyze measurements outcome after each collection period to investigate service status. In Ericsson environment, Fault Manager Subsystem is used to raise alarms via monitoring dashboard, each alarm has predefined severity level with detailed problem description, NOC engineers receive alarms, and based on its severity and defined description, they decide the new situation for the service, and mostly diagnosis the situation by manual check through dedicated APIs, level of the service should be announced and documented for reporting issues. On the other hand, measurements based on predefined and predesigned software counters, results are fetched periodically by performance engineers though special systems, these counters, in Ericsson environment, check service status and count up in case of down service result and then aggregate results as number of down times per quarter (15 minutes) to be used as good performance indicators about service in late-cycle approach.

Above methodology needs huge efforts to analyze and diagnosis service situation based on raised faults, and some time it’s impossible to fit faults’ set into known scenario to decide service situation automatically, hence manual check is mandatory and very hard at the same time, especially in case of many cells degradation due certain incident like storms. Late-cycle approach (measurements) needs long time to be collected and fetched, so this methodology is poor and needs significant development as shown in the above section.

This case study was designed to monitor more than 450 sites in Palestine, Wataniya Mobile operates a mobile network on basis of Ericsson platform technology. Developed process fetches status for all sites through certain APIs on main controllers, every 5 seconds, and then process results to analyze radio service status for all sites and record outages of affected sites into log file, and finally logs are archived in a database, as shown in Figure 1.

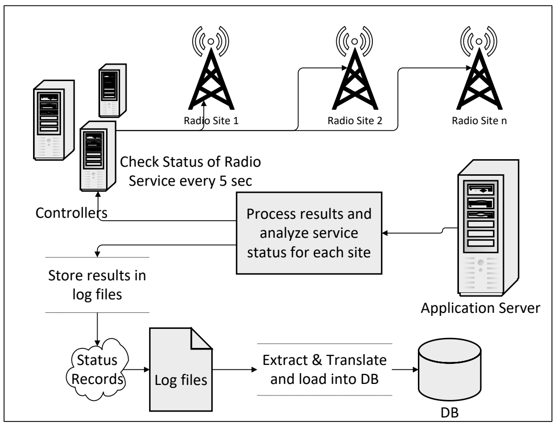
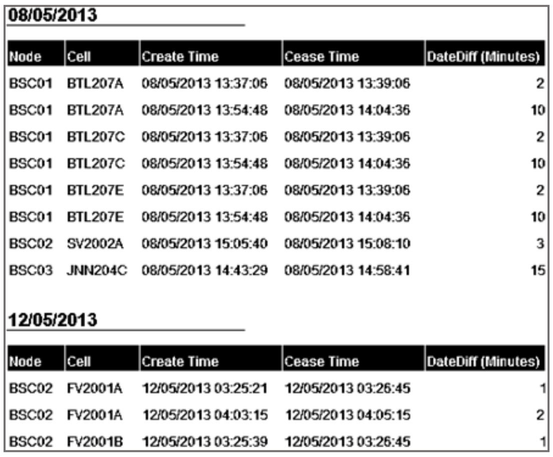

Radio Site 1
Radio Site 2
Radio Site n
Check Status of Radio Service every 5 sec
Controllers
Process results and analyze service status for each site
Application Server
Store results in log files
Status Records
Log files
Extract & Translate and load into DB
DB

Fig. 1: E2E Performance Monitoring System

All these processes are implemented based on java platform, java packages are developed for all needed jobs like fetching service status on all sites, analyzing status with a very fast speed in milliseconds to save high monitoring accuracy, and recording logs into system files, all these tasks are done by java process, which is run over Unix OS on the application server.

After archiving outages’ data of mobile radio services into well-organized database, more than 80000 records are stored within one year, hence rich and well-defined reports can be fetched and presented, as shown in Figure 2, and also it can be correlated with other related data (e.g. revenue) to conclude significant analysis result, as shown in Figure 3.

By comparing new approach with old one we found that there were many interruptions detected on service after using new approach. A brief summary of differences are listed as shown in Figure 4. On the other hand there were some of side effects after applying new approach, but really it was very lite and doesn't cause any major effect on system, even though that systems are loaded, the worst side effect due to running processes of new approach was CPU utilization increment on controllers platform by (2-4%) at most which means very normal situation.



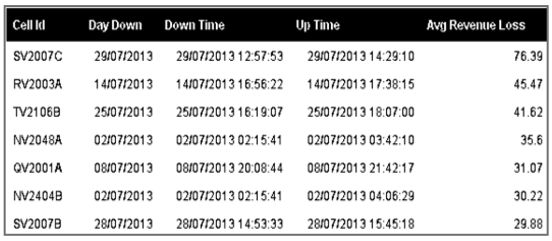
08/05/2013

Node	Cell	Create Time	Cease Time	DateDiff (Minutes)
BSC01	BTL207A	08/05/2013 13:37:06	08/05/2013 13:39:06	2
BSC01	BTL207A	08/05/2013 13:54:48	08/05/2013 14:04:36	10
BSC01	BTL207C	08/05/2013 13:37:06	08/05/2013 13:39:06	2
BSC01	BTL207C	08/05/2013 13:54:48	08/05/2013 14:04:36	10
BSC01	BTL207E	08/05/2013 13:37:06	08/05/2013 13:39:06	2
BSC01	BTL207E	08/05/2013 13:54:48	08/05/2013 14:04:36	10
BSC02	SV2002A	08/05/2013 15:05:40	08/05/2013 15:08:10	3
BSC03	JNN204C	08/05/2013 14:43:29	08/05/2013 14:58:41	15

12/05/2013

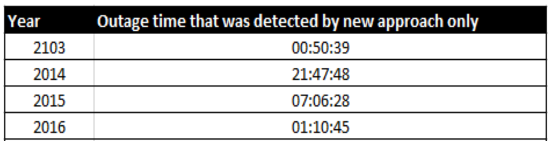
Node	Cell	Create Time	Cease Time	DateDiff (Minutes)
BSC02	FV2001A	12/05/2013 03:25:21	12/05/2013 03:26:45	1
BSC02	FV2001A	12/05/2013 04:03:15	12/05/2013 04:05:15	2
BSC02	FV2001B	12/05/2013 03:25:39	12/05/2013 03:26:45	1

Fig. 2: Outage data of mobile service



Cell Id	Day Down	Down Time	Up Time	Avg Revenue Loss
SV2007C	29/07/2013	29/07/2013 12:57:53	29/07/2013 14:29:10	76.39
RV2003A	14/07/2013	14/07/2013 16:56:22	14/07/2013 17:38:15	45.47
TV2106B	25/07/2013	25/07/2013 16:19:07	25/07/2013 18:07:00	41.62
NV2048A	02/07/2013	02/07/2013 02:15:41	02/07/2013 03:42:10	35.6
QV2001A	08/07/2013	08/07/2013 20:08:44	08/07/2013 21:42:17	31.07
NV2404B	02/07/2013	02/07/2013 02:15:41	02/07/2013 04:06:29	30.22
SV2007B	28/07/2013	28/07/2013 14:53:33	28/07/2013 15:45:18	29.88

Fig. 3: Revenue loss due to outage



Year	Outage time that was detected by new approach only
2103	00:50:39
2014	21:47:48
2015	07:06:28
2016	01:10:45

Fig. 4: Outage times detected by new approach

3.3 Second case: Monitoring SMS Services

SMS service is one of most important services in mobile systems, yes after the advanced generations of mobile systems, we faced less attraction for this service, but really it is still forms the based platform and prerequisite service for other main services that used intensively nowadays like MCN, MMS, MNN, USSD. SMS business services usually forms more than 10% of total revenue which means a very big concern from top management, so performance monitoring is very important issue to keep high service availability and high level of quality.

SMSC is the system that is responsible about SMS delivery from sender to receiver, sender and receiver can be simulated via mobile station or computer application, SMSC is integrated with mobile switching center to be accessible through mobile station, and also it can be accessed directly form computer application by using SMPP protocol which is usually used to send and receive SMS.

Traditional approaches are usually used to monitor performance level of SMS service, wherein engineers monitor faults and alarms via Fault Handler Subsystem which is part of SMSC system and also they can follow other performance issues by using statistics of measurement subsystem. These two subsystems are main components for SMSC systems and both of them forms model and measurement based approaches respectively.

SMSC systems are based on very complex infrastructure and platforms to keep its services available, scalable, and reliable all the time, so SMS service depends on many other infrastructure services that should be running well to get valid SMS service, therefore while monitoring, many deferent types of alarms and faults among deferent platforms that may be raised, so engineers shall know enough about all scenarios that may interrupt the service, it is a very hard job, and sometime it is being uncertain until receiving complaints from customers.

Furthermore, SMS service availability depends on mobile radio and core backbones, so it is impossible to evaluate service status based on model or measurements approaches by monitoring faults, alarms, and statistics only as shown in Figure 5. End-to-end testing is a simpler approach to monitor service.

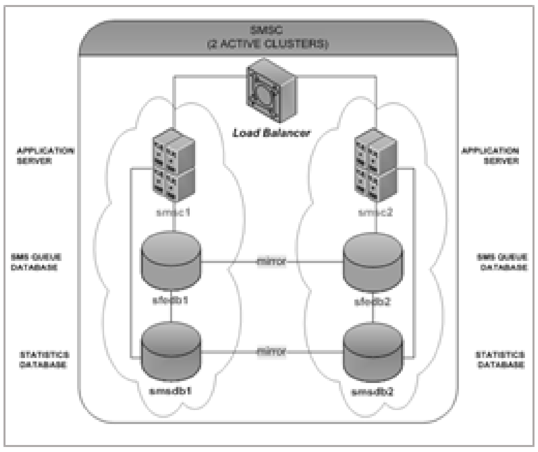


Fig. 5: SMSC system node architecture

To test SMS service from end to end we need similar simulator to check service availably exactly like manual test, so SMPP application have been developed based on java platform to handle SMS message via SMSC over SMPP protocol on IP backbone.

Two main test scenarios are implemented, first one was used to check SMSC system availability with all its related systems and components to check complete process life cycle on the system including delivery and charging process, and the second scenario was used to check round trip delivery time over radio network to certain mobile station, and then reply back from that mobile station.

To implement first test scenario, SMPP client application is essential to send test message to SMSC, every 30 seconds, by using SMPP protocol over TCP, SMSC was configured to receive that message and resend it back to the sender, after sending message from application side, it waits up to 30 seconds until receiving the response message from the SMSC side, if response isn't received, SMSC will be assumed as unavailable, outage period will be documented, and continue in test message every 30 seconds.

Transient SMS must practise all subsystems and related systems on SMSC (including charging), which insures that all processes and components are running well, as shown in Figure 6. Radio accessibility is not checked in this scenario because it is already checked in the other scenario.

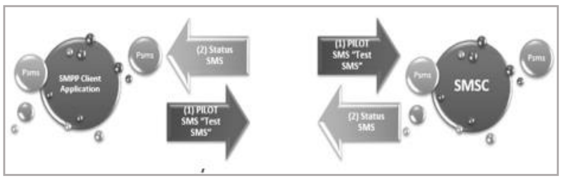


Fig. 6: SMSC monitoring first scenario

In the second scenario, message delivery time will be measured for its round trip from-to mobile station over radio network access. Other instance of the same SMPP client application can be used again, but with deferent message routing criteria, this time message will be forwarded to certain mobile station over radio access network, and then the same message will be replied back automatically to the client application by using dedicated real time application on that mobile station, as shown in Figure 7.

SMPP client application will repeat it by sending test message to the SMSC every 2 minutes, and then waits up to 30 seconds to receive the response from mobile station, else it will record an outage in logs. While waiting, it continues in sending test messages. Client application records spent delivery time into organized data logs on system file, this test has been run for more than one year and already tested in real interruption occurrence, delivery time results are being recorded in logs, and also loaded into database very well, hence it can be correlated easily with other valuable data to analyze incidents and low performance issues. Figure 8 illustrate a sample of statistical data.

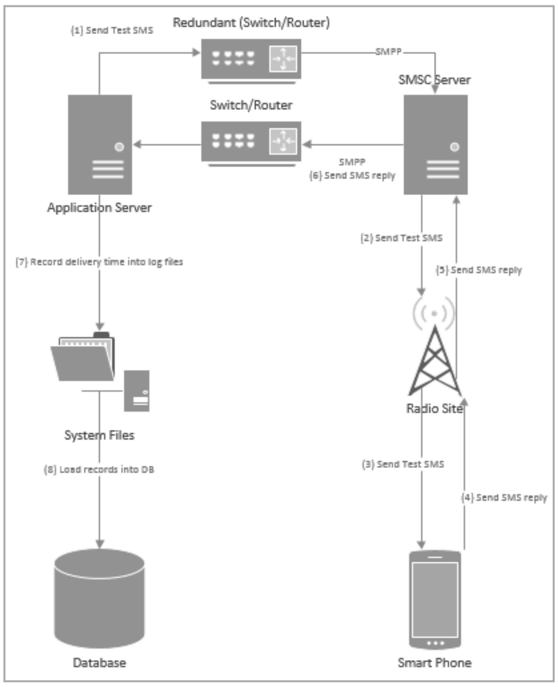
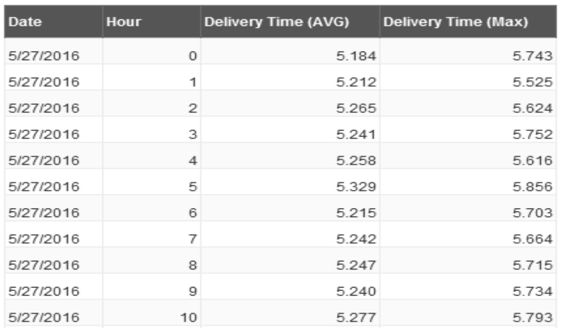


Fig. 7: SMSC monitoring second scenario

By using this approach, we can confirm that performance of short message service is monitored all the time from end to end, and so no need to go deep inside complex system and infrastructure to analyze effect of certain alarms on the quality of service, and we will not wait until receiving complaints from customers to take action which means high loss.



Date	Hour	Delivery Time (AVG)	Delivery Time (Max)
5/27/2016	0	5.184	5.743
5/27/2016	1	5.212	5.525
5/27/2016	2	5.265	5.624
5/27/2016	3	5.241	5.752
5/27/2016	4	5.258	5.616
5/27/2016	5	5.329	5.856
5/27/2016	6	5.215	5.703
5/27/2016	7	5.242	5.664
5/27/2016	8	5.247	5.715
5/27/2016	9	5.240	5.734
5/27/2016	10	5.277	5.793

Fig. 8: Sample of SMS data

4. DISCUSSION OF RESULTS

This approach is mostly mentioned in the context of manual procedure only, but no one has tried to automate this approach within certain accuracy. In general, researchers implicitly assume that this approach is for manual resources only, and they can't imagine that it could be automated. This may due to theoretical assumptions like bad side effects on the resources of the systems. After applying these two case studies, we proved that this approach is practical, precise and can be used with most critical systems with a very light side effects on system's resources. On the contrary, it provides an accurate monitoring for the services all the time which must offer real figure about trends of system status all the time.

Therefore practitioners can use this approach by developing dedicated applications to provide real performance monitoring for their systems and services all the time, and also they may use developed tools and applications that are based on this approach to be as an auxiliary tool for human resources.

And in parallel, researchers and students can use this contribution to build and include an integrated model architecture, based on the performance early-cycle approach, to offer more suitable interfaces and protocols for this new approach.

5. THREATS TO VALIDITY

Internal validity makes sure that relationship between treatment and outcome is a causal relationship, and then the result of the study can be generalized outside the scope of our study. Instrumentation validity considers the effect caused by the artefacts used for experiment execution, such as data collection forms, document to be inspected in an inspection experiment, and so SMPP connection between application server and SMSC server forms instrumentation internal threat and it was mitigated by human monitoring and by considering it in the check list in case of analysis process for certain degradation or interruption, and also TCP connection between application server and radio controllers forms instrumentation threat and it was mitigated by automatic health check and notification to alert in case of any interruption.

External validity makes sure that relationship between cause and effect is a causal relationship, and it is not a result of a factor of which we have no control or have not measured. And confirms that the treatment causes the outcome. Instruction of setting and treatments considers the effect of not having the experimental setting or material representative of, therefore selection of service or function that should be handled via simulator forms "Instruction of setting and treatments" external threat, and it was mitigated by interacting with certain service that absolutely will pass the complete life cycle through system components and processes to insure that all parts and process of the system are running well, e.g. asking SMSC to send message to certain mobile station means that all processes with all related systems will be executed like routing, charging, radio signalling, and data delivery.

Construct validity concerns with the relation between theory and observation when the relationship between cause and effect is causal. It must make sure that the treatment reflects the construct of the cause well and the outcome reflects the construct of the effect well. Inadequate preoperational explication of constructs conceders that constructs are not sufficiently defined, before they are translated into measures or treatments, hence design of accuracy level for monitoring forms a design construction threat, and it was mitigated by using very high accurate level of design, especially at first stage of monitoring until make tuning for optimum check period, e.g. check radio service availability every 5 seconds, and test message was being sent every 30 seconds in the first stage.

Conclusion validity makes sure that there is a statistical relationship between treatment and outcome, and then we can draw a correct conclusion about relations between the treatment and the outcome of an experiment. Power of a statistical test is the ability of the test to reveal a true pattern in the data, so low statistical power forms conclusion threat, and it was mitigated by using very intensive data sampling to draw an accurate trends, e.g. radio service status was being checked every 5 seconds, and also test message was being sent every 5 seconds in the first stage before it was tuned to 30 seconds to check availability and 120 seconds to check delivery time.

6. CONCLUSION

This paper introduced a simple and practical methodology to monitor systems among deferent complex components of software and hardware systems, all these aspects are wrapped and encapsulated as one abstract box and matched by terms from end to end.

After completing both case studies on SMS center and on mobile radio service for more than one year, it is verified that the approach is practical and accurate, and by comparing it with most applied approaches, we found that it is more accurate and simpler than common ones.

The implemented case studies show and confirm that we can simulate manual performance monitoring exactly like real human behaviour without any systematical restriction, hence complex algorithms and processes for status diagnosis aren't farther needed in live system monitoring.

While this approach was mostly mentioned in context of manual procedure only, and no one has tried to automate this approach within certain accuracy, and researchers assume implicitly that this approach is for manual resources only, and they can't imagine that it could be automated, this may refer to theoretical assumptions like bad side effects on the resources of the systems. While all the above issues, we applied these two case studies, and approved that this approach is practical, precise and can be used with most critical systems with a very lite side effects on system's resources, and on the contrary it provides an accurate monitory for the services all the time which must offer real figure about trends of system status all the times precisely.

Therefore practitioners can use this approach by developing dedicated applications to provide real performance monitoring for their systems and services all the time, and also they may use developed tools and applications that are based on this approach to be as an auxiliary tools for human resources.

As for future work, researchers and students can build an integrated model architecture, based on performance early-cycle approach, to offer more suitable interfaces and protocols for this new approach.

7. ACKNOWLEDGMENT

We would like to thank Wataniya Mobile Company in Palestine, a member of Ooredoo Group, for this opportunity to perform the case study and for their support and helpfulness.

8. GLOSARY

API	Application Programming Interface
ATP	Acceptance Test Procedure
CPU	Central Processing Unit
E2E	End to End
MCN	Missed Call Notification
MMS	Multimedia Messaging Service
MMSC	Multimedia Messaging Service Center
MNN	My New Number, as VAS service
NMS	Network Management System
NOC	Network Operations Center
OS	Operating System
SPE	Software Performance Engineering
SMPP	Short Message Peer to Peer
SMS	Short Messaging Service
SMSC	Short Messaging Service Center
TCP	Transmission Control Protocol
USSD	Unstructured Supplementary Service Data
VAS	Value Added Services
WAP	Wireless Access Protocol

9. REFERENCES

[1] Woodside, Murray, Greg Franks, and Dorina C. Petriu. "The future of software performance engineering." Future of Software Engineering, 2007. FOSE'07. IEEE, 2007.

[2] Mahdavi, Adrian. Value Added Services and Content Platforms. Diss. Royal Institute of Technology, Stockholm, Sweden, 2003.

[3] C.U. Smith. Performance Engineering of Software Systems. Addison Wesley, 1990.

[4] C.U. Smith, "Software Performance Engineering", Encyclopedia of Software Engineering, Wiley, 2002.

[5] S. Balsamo, A. DiMarco, P. Inverardi, and M. Simeoni, "Model-based Performance Prediction in Software Development", IEEE Trans. on Software Eng., vol. 30, 2004, pp. 295-310.

[6] M. Arlitt, D. Krishnamurthy, J. Rolia, "Characterizing the Scalability of a Large Web-based Shopping System", ACM Trans. on Internet Technology, v 1, 2001, pp. 44-69.

[7] A. Avritzer, J. Kondek, D. Liu, and E. J. Weyuker, "Software performance testing based on workload characterization," in Proc. WOSP'2002, Rome, , pp. 17-24.

[8] S. Barber, "Creating Effective Load Models for Performance Testing with Incomplete Empirical Data", in Proc. 6th IEEE Int. Workshop on Web Site Evolution, 2004, pp. 51-59.

[9] S. Barber, "Beyond performance testing", parts 1-14, IBM DeveloperWorks, Rational Technical Library, 2004, www-128.ibm.com/developerworks/rational/library/4169.html.

[10] Krogmann, Klaus, et al. "Improved feedback for architectural performance prediction using software cartography visualizations." International Conference on the Quality of Software Architectures. Springer Berlin Heidelberg, 2009.

[11] Mabrouk, Nebil Ben, Nikolaos Georgantas, and Valerie Issarny. "A semantic end-to-end QoS model for dynamic service oriented environments." Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems. IEEE Computer Society, 2009.

[12] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck, " A Close Examination of Performance and Power Characteristics of 4G LTE Networks,". University of Michigan, AT&T Labs - Research.K. Elissa, "Title of paper if known," unpublished.

[13] A. Korodi and T. L. Dragomir, "Mobile fault detection and diagnosis module for automatic systems," 2007 Mediterranean Conference on Control & Automation, Athens, 2007, pp. 1-6.doi: 10.1109/MED.2007.4433942

[14] C. M. Vong, P. K. Wong and W. F. Ip, "A New Framework of Simultaneous-Fault Diagnosis Using Pairwise Probabilistic Multi-Label Classification for Time-Dependent Patterns," in IEEE Transactions on Industrial Electronics, vol. 60, no. 8, pp. 3372-3385, Aug. 2013.

[15] L. Hou and N. W. Bergmann, "Novel Industrial Wireless Sensor Networks for Machine Condition Monitoring and Fault Diagnosis," in IEEE Transactions on Instrumentation and Measurement, vol. 61, no. 10, pp. 2787-2798, Oct. 2012.