Automated Software Performance Improvement: Best Practices for Evaluation

Velmourougan Suburayan
Scientist, Centre for Reliability, Dr.VSI Estate, Thiruvanmiyur, Chennai -600041

Dhavachelvan Ponnurangam
Phd, Professor, School of Computer Science and Engineering, University of Pondicherry- Puducherry India Pin- 605008

Baskaran Ramachandran
Assistant professor, School of Computer Science and Engineering, Anna University, Chennai, India, Pin- 600021

ABSTRACT
Automated software has different dynamic behaviour during use when compared to general software application. Mostly these dynamic characteristics degrade the application performance during operation due to lack of understanding of performance requirement during testing. Mis-understanding on requirement for testing by the test engineer lead to loss of reputation, financial and operational loss to the community using the automated application. Software-performance issues are not only to be patched up by coding routines and pumping testing effort but, can be easily eliminated by inculcating the best practices during testing phase of the software development life cycle. Software-performance is the effective attribute to improve the maintainability and reliability of the software application. This paper outlines the various activities to be carried out during the testing phase of the lifecycle while developing the automated software to eliminate performance related issues during the software operations and maintenance. Moreover this paper also presents a set of performance testing taxonomy to inculcate efficient performance into the automated applications developed.

Keywords
Performance testing, Automated Software, Load testing.

1. INTRODUCTION
Automatic Software (AS) has a significant role in today's industry for conducting process monitoring, testing, measurement and diagnostics of various components, sub-assemblies, termed as "unit under monitoring" (UUM) to ensure that they meet their required performance characteristics [1]. As the evolution of testing technology improves the measurement performance capabilities of legacy test systems need to be analyzed to ensure that both the system and the testing software still perform according to the UUM. The performance of the automated software plays a critical role in determining the financial implication and reputation of the process [2]. Performance degradation is due to improper coding, improper design, inadequate testing due to stringent delivery schedules, and inadequate maintenance [3]. As far as the automated software is concerned it has an additional constraint on the dynamic characteristics of the software and the associated interfaces/UUM. The performance degradation triggers the deviations in the response time, abrupt delay between functions [2, 3] causing disruptions to real time objectives etc. In software engineering, performance testing is performed, to determine how fast the key functional aspects of a system perform under a particular workload. It can also serve to validate and verify other quality attributes of the system, such as scalability, reliability and resource usage. Performance testing is a subset of Performance engineering, an emerging computer science practice which strives to build performance into the design and architecture of a system, prior to the onset of actual coding effort [4][5].

Evaluation process namely has four stages; Evaluation plan, Establish acceptance criteria, design evaluation and execution of evaluation process [6] and its associated activities as shown Figure 1.

1.1 Evaluation plan

Identify the physical test environment and the production environment as well as the tools and resources available to the test team. The physical environment includes hardware, software, and network configurations. Having a thorough understanding of the entire test environment at the outset enables more efficient test design and planning and helps you identify testing challenges early in the project [7]. In some situations, this process must be revisited periodically throughout the project's life cycle. Figure 2 depicts the model of the schedule for the evaluation of automated software. This schedule is approximately planned for five week covering evaluation plan, establishing the evaluation criteria, design of evaluation scenario and execution of evaluation plan.

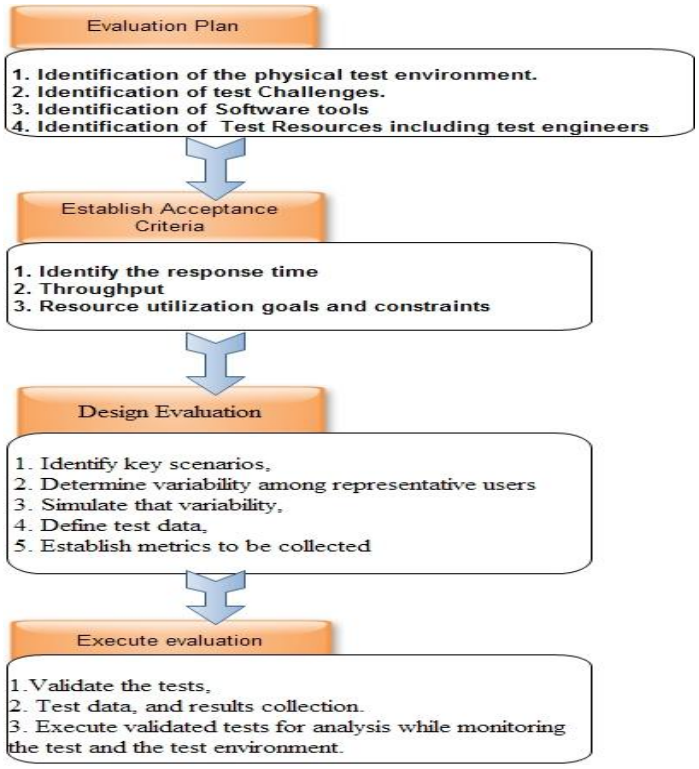

Fig 1: Life cycle of the Evaluation process

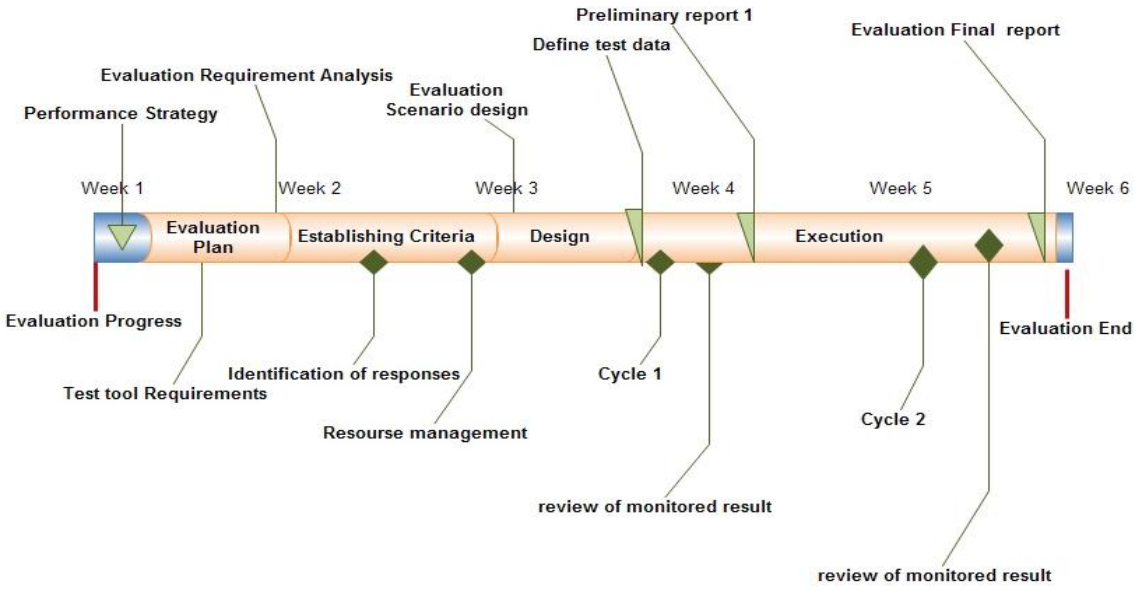

Fig 2: Schedule for the Evaluation of Automated software performance

1.2 Establish Acceptance Criteria.

Identify the response time, throughput, and resource utilization goals and constraints. In general, response time is a user concern, throughput is a business concern, and resource utilization is a system concern. Additionally, identify project success criteria that may not be captured by those goals and constraints[8]; for example, using performance tests to evaluate what combination of configuration settings will result in the most desirable performance characteristics.

1.3 Designing evaluation plan:

Identify key scenarios, determine variability among representative users and how to simulate that variability, define test data, and establish metrics to be collected. Consolidate this information into one or more models of system usage to be implemented, executed, and analyzed. Configure the Test Environment to prepare the test environment-tools and resources necessary to execute each strategy as essential features and components that is available for test [9]. Ensure that the test environment is instrumented for resource monitoring as necessary.

1.4 Execution of Evaluation process:

Implement the test design by developing the performance tests in accordance with the test design. Execute the Test and monitor your tests. Validate the tests, test data, and results collection. Execute validated tests for analysis while monitoring the test and the test environment. Analyze Results, Tune, and Retest. Analyze, Consolidate and share the results fine tune the relevant change and retest. Ensure whether there is Improvement or degradation. Each improvement made will return smaller improvement in performance than the previous results. Determine, when do you stop? When do you reach a CPU bottleneck? The choices then are either improve the code or add multiple processors (CPU).

In this paper authors have segregated the performance testing as an evaluation process to find out the possible shortcomings of the automated software. What are the features that the automated software must have to ensure better performance? What are the critical codes of best practices to be identified during the evaluation of the product?

2. THE CHALLENGES TO THE AUTOMATED SOFTWARE

Unlike software application the automated software has the additional responsibilities to understand and react to the dynamic processes to ensure better performance;

1. Under which load does the application encounter error?

2. Are the system failures reproducible? Or repetitive in nature,

3. Whether the bad performance reproducible?

4. Is it a system, application or configuration failure?

5. Where is the bottleneck in my infrastructure?

To satisfy the above queries automated software must have built-in non functional features such as Auto recovery, Diagnostics, auto logging, auto response to exceptions, mechanism to monitor network health and recovery, early warning system etc.,

The performance influencing indicators can be subdivided into system, application related and load related and the self inducing issues.

2.1 The system related parameters

1. Number of hardware components
2. Configuration of related hardware
3. Network components
4. Configuration settings (HW)
5. Operating system
6. Standard and custom software components
7. External devices that access the system

The Figure 3 below depicts the overall logical and architectural requirements of the automated software to deliver the desired performance.

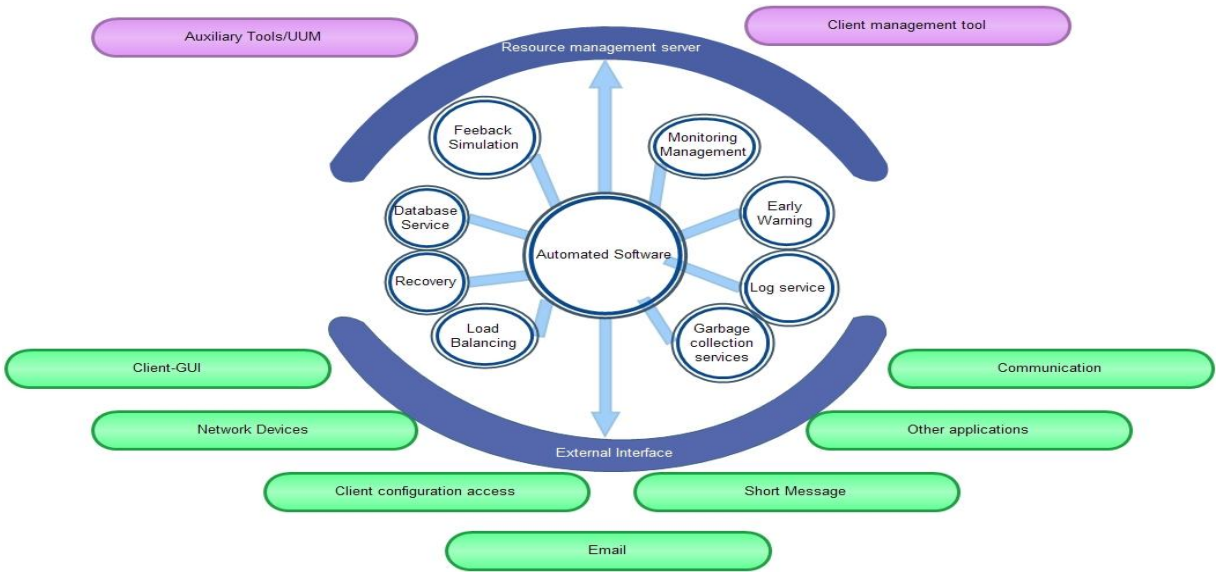

Fig. 3: Structure (Logical) of the automated software with performance requirements.

2.2 The Application related issues are;

1. Test cases used for the test
2. Type of application
3. Database structure and data quantities
4. Configuration and architecture of the environment
5. Protocols for data exchange

2.3 The load related issues are;

1. Number of users
2. Number of sessions

2.4 The self inducing issues are;

1. Cache/memory leak
2. Log-read –write delay
3. Data capture delay
4. Intermittent interface with UUM
5. Interoperability/protocol mismatches etc.

3.0 THE CHALLENGES TO THE EVALUATOR

Since the automated software has to be built with the additional non-functional features as discussed the evaluator must have the capability to check the functionality of the automated software to ensure whether the software has the capability to demonstrate the required performance [10,11,12]. The evaluation plan must specifically address the scenarios to demonstrate the capability of the automated software to manage the desirable performance during critical performance issues.

3.1 Evaluation plan:

The real engineering challenge during evaluation is not only being able to meet your business requirements, but also achieving them on time and the evaluator must first determine those business requirements. For example, the need to determine the budget for new hardware, and what existing hardware is available and also need to know how rigid the target delivery date is, whether an initial release to a limited number of users is acceptable, and what project aspects take priority. Some areas worthy of significant consideration in a performance plan include:

- Throughput and latency (e.g., do you need to ensure that deploying this application will not adversely affect other applications that use the same network?)
- Reaction (e.g., are there components of your application that need to interact in a timely manner, such as a load balancer that skips a particular Web server if it does not respond in <200 milliseconds?)
- Capacity planning (e.g., can you afford the infrastructure to support up to 500 users under standard conditions?)
- Entry cost (e.g., is it viable to achieve the end-user requirements with existing hardware?)

3.2 Establishing acceptance criteria:

Determining performance-testing objectives are not easy, unless the system characteristics are understood and especially it difficult task for the automated software. The challenge is that the performance tester does not always have easy access to either explicit or implied objectives, and therefore frequently must conduct a systematic search for these objectives. Determining performance-testing objectives generally involves the following tasks: Determining the overall objectives for the performance testing effort, such as detect bottlenecks that need tuning, assist the development team in determining the performance characteristics for various configuration options, providing input data for scalability and capacity-planning efforts, reviewing the project plan with individual team members or small groups.

The performance of a system is described by the following aspects;

1. CPU / Memory usage of the related systems
2. System Calls / Context switches per time period
3. Running / blocking processes per time period
4. Network throughput
5. Paging rates
6. Database calls per timeframe
7. Response times for the end-user
8. Error rate
9. Transactions per time period

All these issues are common to all type of software but the automated software has a factor of influences in the above nine aspects. E.g. Usage of CPU in automated software.

Best practices to address the performance related issues are as listed below and the evaluator must consider every point as discussed below before deriving the acceptance criteria:

- What functionality, architecture, and/or hardware will be changing between the last iteration and this iteration?
- Is tuning likely to be required as a result of this change? Are there any metrics that I can collect to help you with the tuning?
- Is this change likely to impact other areas for which we have previously tested/collected metrics?
- Reviewing both the physical and logical architecture with individual team members or small groups. As you review the architecture, ask questions such as:

    1. Have you ever done this/used this before?
    2. How can we determine early in the process if this is performing within acceptable parameters?
    3. Is this likely to need tuning? What tests can I run or what metrics can I collect to assist in making this determination?
    4. Asking individual team members about their biggest performance-related concern(s) for the project and how you could detect those problems as early as possible.

3.3 Designing Evaluation

3.3.1 Design Tests.

Identify all key scenarios in addition to the general performance issues especially scenarios which are influenced by the inclusion of automated software, determine variability among representative users. Ensure that automated software do not have any randomly varying influence in the routine function of the application and how to simulate that

variability. Define test data and establish procedure on metrics to be collected. Consolidate this information into one or more models of system usage to be implemented, executed, and analyzed.

3.3.2 Configure the Test Environment.

Prepare the test environment, tools, and resources necessary to execute each strategy as features and components become available for test. Ensure that the test environment is instrumented for resource monitoring as necessary.

3.3.3 Test Data

Test data analysis & design is critical for performance testing, mainly due to the volumes of data required and the need to apply load to all parts of the system. This section describes the plan for data provision and loading.

Three categories of data are required to be produced:

1. **Reference data** – data that must exist on the database prior to test execution as it will be referenced by the transactions executed against the database during testing; for example, brokers details, valid banks.
2. **Transaction data** – the parameterized details of transactions that are to be executed during testing. For example, a purchase transaction would need data of the type buyer's name and address, purchase details, etc.
3. **Bulk data** – this is data that must exist on the database but is unchanged throughout testing. This data is either not at all involved or only indirectly involved in the testing, often for reasons of providing bulk, realistic searching or sorting, etc.

It is important that all these data types are specified for the required testing.

The key questions are: What types of data do we need? How much data do we need? How will it be provided?

a) Data Analysis

1. Describe types and volumes of data needed:
2. Transactional data – for test scripts
3. Reference data – data in database referred to by test scripts
4. Bulk data – data not directly used by test but necessary to create a realistic database size

b) Describe relevant data flows:

1. End-to-end across the system
2. Upstream – data flows received from other systems
3. Downstream – data flows sent to other systems

c) Data Provision

1. Describe how data will be generated or sourced, e.g.
2. Copy of data from production – does it need to be sanitized to comply with the data Protection Act?
3. Generated from scratch – using what tool/process?

d) Implement the Test Design.

Develop the performance tests in accordance with the test design.

The best practices are;

i) Definition of test cases, business processes?

1. Which Application should be tested?
2. What are the business processes?
3. Measuring points?
4. Test data?

ii) Definition of set of test cases (Multiple Application Tests)

1. Which test cases are needed?
2. What is the correct ratio?
3. Which Load generators to use?

iii) Modeling of the scenario

1. What are the performance influencing parameters?
2. Single Application Test or how many number of virtual users
3. Multiple Application Test
4. Increase number of applications
5. Increase load on one of the applications (change ratio)
6. Constant Load (and start other processes)

iv) Positioning of Load Generators

1. How many Load generators are needed?
2. If it is relevant, where to position the load generators?
3. Are all the results interpretable by someone?

3.4 Execute Evaluation

Run and monitor your tests. Validate the tests, test data, and results collection. Execute validated tests for analysis while monitoring the test and the test environment.

Monitoring of the environment shall cover which components should be monitored? What parameters are relevant? Which software is needed for the monitoring? How often to measure means of the frequency of monitoring, and optimum analyzes, not too much as to overload the system.

Statistical analysis of all test series need to be analyzed statistically and in each interval of change (variation of the "load parameter", usually the number of users), the Error rate, the Mean value, the Confidence interval (Reliability/scattering of the mean value),the test series are used to calculate parameters that have not been measured. E.g. Signature on Application, Also Error analysis, what is the error rate (correct measurements versus Errors), what kind of errors occurred whether Application errors or Protocol errors, what's the reason for the errors? (Load, System, Data- related), Correlation of test series whether all the test series must be viewed in a context. Next step is rating of the results, does the system reach the expected performance ?, were all performance requirements reached ?, What is the allowance of the system regarding higher load ?, How are the components

are effected be changes in the load ?,What are the performance restricting values or parameters?, Does the applications Interfere ?

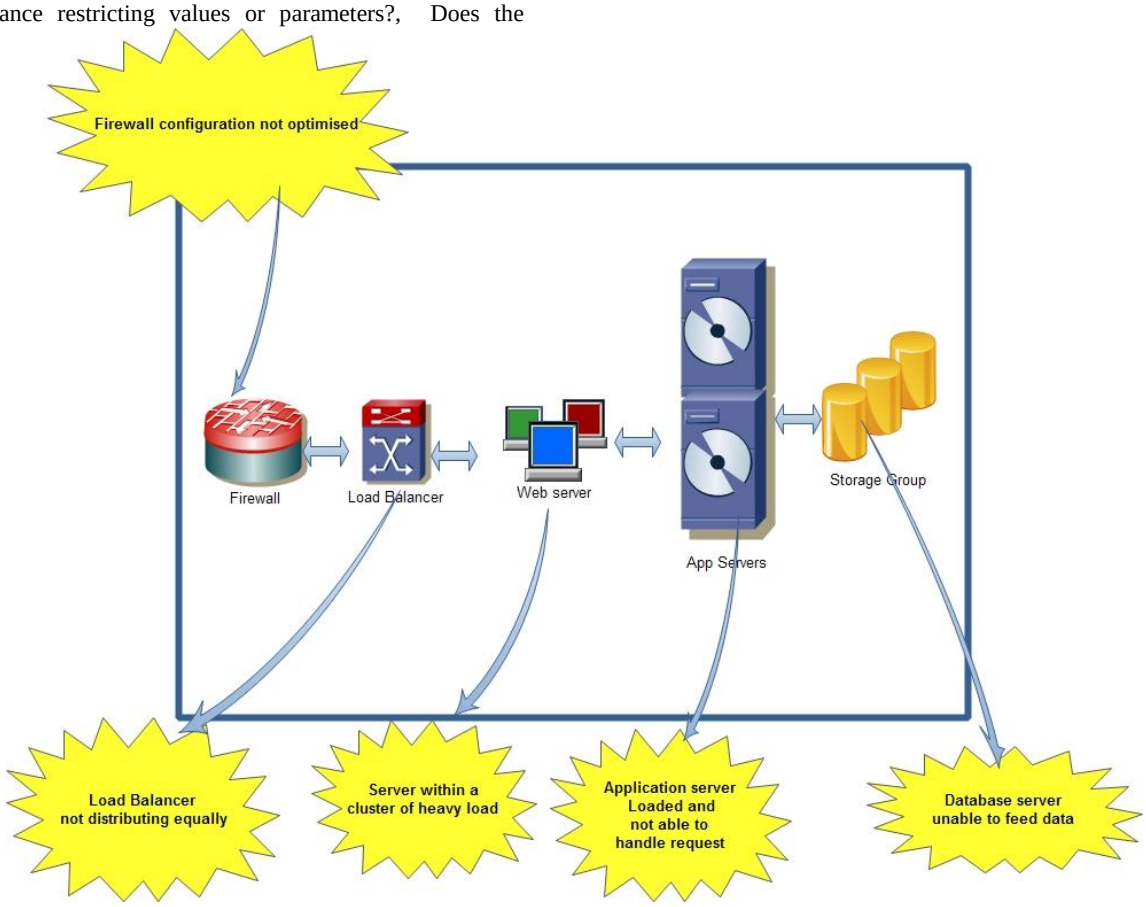

Fig 4: Representation of bottlenecks in the system monitoring architecture.

Based on the analysis a re-run might be required to verify the evaluation results, test time was not long enough to gather enough statistical data. Once the system environment changed the test cases need to be changed or enhanced because of changed environmental requirements then optimize the system functions. Finally ensure the expected performance was reached and troubleshooter (problem was not found until now). The Figure 4 represents the possible bottleneck in the overall automated system deployed in monitoring. The key issues are; 1) load balancer not distributed evenly, 2) improper optimization of firewall, 3) Server stuck with a cluster of heavy load, 4) Application server unable to handle requests, 5) Low performance of database to feed data to the application. Faulty transactions usually have different response times as compared to correct ones. So they should be analyzed separately considering, 1) error rate per interval, 2) Occurrence of the first errors, 3) Interval between the errors (any regularity or script errors?),4) Un-usable or wrong test data?

The Average Transaction Response Time shows the response time for all measurements, Response time is shown against the elapsed time and whether it is possible to see when the response time exceeds the target value. The Statistical values are graphed as minimum, maximum, average with Standard deviation to understand its behaviors. Mostly the tools calculate these statistics for the entire test and not for particular "load steps" but this is not very useful when the load changed during the run. Figure 5 represents the average response time of the task of a key function with other monitoring function running. This delay is generally termed as "The Cost" or in other terms it is called the consumption of bandwidth or resources. The evaluation report has to be submitted after the completion of the evaluation

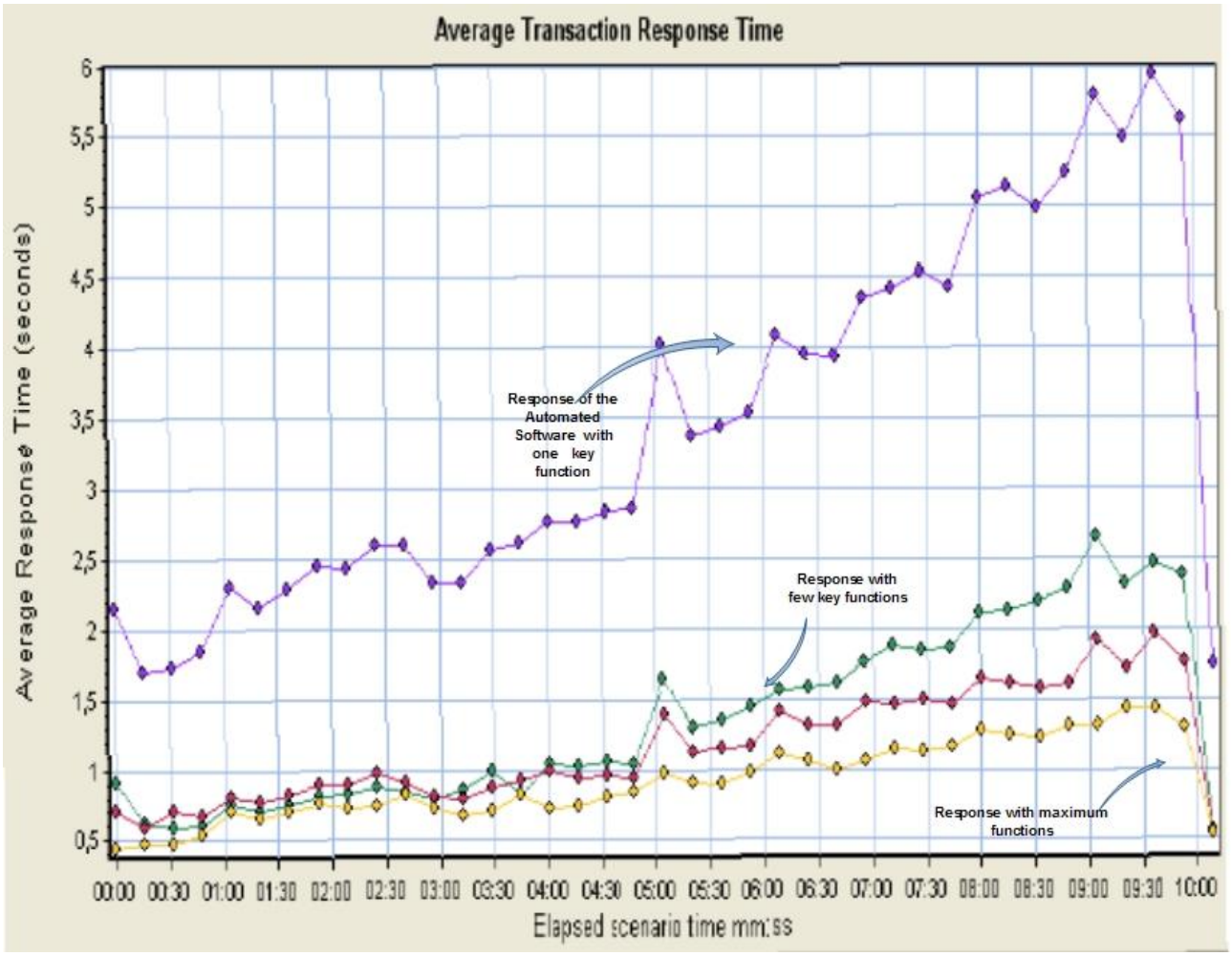

Fig 5: Average response time versus the elapsed scenario during the operation of automated software. (Sample of performance testing result)

Automatic reports, customized to contain the most relevant information, will be made available in a shared directory after each test scenario execution. This will contain:

1. Detailed results of test scenarios
2. Summaries of response time data collected during test run
3. Graphs of transaction throughput and all online monitors collected during test
4. Error breakdown

The Daily status reports will be sent to test management, the performance test team, business analysts and the performance test technical support team during test execution. These will detail:

1. Progress against plan and test objectives
2. Summary of results & findings so far
3. Planned next steps

The Flash reports will be provided after each test cycle, containing:

1. Summary of progress against project plan – are we on course to complete on time? Has the expected amount of performance testing been done?
2. Summary of results & analysis so far with conclusions & recommendations – Is the system meeting the performance requirements? How can the performance be improved?

The Full results report will be created once the planned tests are completed, detailing:

1. Summary of tests run, results & system tuning conducted
2. Key findings, conclusions & recommendations
3. RAG metrics reporting on the Measurable Success Criteria of each test – were the key test objectives met?
4. Statement of Readiness – concise statement answering the key questions asked during the performance testing. Is the application ready to go live?

4. CONCLUSION

The performance evaluation taxonomy has been developed based on the best practices and practical testing and is presented in the Appendix [A.1]

The evaluation of automated software is not an easy task and they have to be given with additional effort to plan, define, design and execute. It is an additional testing effort at the end of the product development to ensure that the automated software meets the performance requirement in terms of reputation, financial or other critical aspects. Because the

automated software consumes the "The cost" of the actual process in terms of transaction response, read/write delay or the delay due to mapping with other UUMs the automated software must be evaluated beyond its functional requirement addressing the non-functional requirements such as Auto recovery, self-diagnostics, auto logging, auto response to exceptions, mechanism to monitor network health and recovery, early warning system etc as shown in Figure 3 of the section 2.1 of this paper. Authors have taken effort to compile the performance evaluation taxonomy to ease out the evaluator problems in derive and define the performance requirement of the automated software deployed for the monitoring. Further these best practices and taxonomy can be tailored to meet application of different natures like semi-automated, standalone, web-based projects.

5. REFERENCES

[1] K. Kumar, "Post Implementation Evaluation of Computer-Based Information System: Current Practices," Comm. ACM, Vol. 33, No. 2, Feb. 1990, pp. 203- 212.

[2] J.S. Chandler, "A Multiple Criteria Approach for Evaluating Information Systems," MIS Quarterly, Vol. 6, No. 1, Mar. 1982, pp. 61-74.

[3] V.R. Basili and H.D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," IEEE Trans. Software Eng., Vol. 14, No. 6, June 1988, pp. 758-773.

[4] J. Mylopoulus, "Conceptual Modeling and Telos," in Conceptual Modeling, Databases and CASE, P. Loucopoulos and R. Zicari, eds., Wiley & Sons, New York, 1992.

[5] G. Boloix and P.N. Robillard, "Comprehensive Software Metrics Framework," Tech. Report EPM/RT-94/07, Dept. Electrical and Computer Eng.,École Polytechnique de Montréal, Montréal, 1994.

[6] L.L. Constantine, "Work Organization: Paradigms for Project Management and Organization," Comm. ACM, Vol. 36, No. 10, Oct. 1993, pp. 34-43.

[7] Fuggetta, "A Classification of CASE Technology," Computer, Vol. 26, No. 12, Dec. 1993, pp. 25-38.

[8] J. Nielsen, Usability Engineering, Academic Press, London, 1993.

[9] E. Brynjolfsson, "The Productivity Paradox of Information Technology," Comm. ACM, Vol. 36, No. 12, Dec. 1993, pp. 66-77.

[10] T.L. Saaty, The Analytic Hierarchy Process,McGraw-Hill, New York, 1980.

[11] Clement, "Computing at Work: Empowering Action by Low-level Users," Comm. ACM, Vol. 37, No. 1, Jan. 1994, pp. 53-63.

[12] Brown, A.W. Wallnau, K.C, A framework for evaluating software technology. IEEE Software, Vol. 13, No. 5, pp.39–49

Appendix [A.1]

Table 1 Performance Evaluation Taxonomy

SlNo	Scenarios	Evaluation/Type	Description
T.1	What is the best end-to-end performance for each of the transactions in the automated application?. What is the Common performance degradation due to automated software itself?	Performance benchmarking	Measure the performance (response latencies, resource usage etc) of an application and determine whether it meets the performance criteria of the business
T.2	How can we improve performance without upgrading infrastructure? How without disabling other core functions of the automated application. What is the time taken for diagnostics?	Performance diagnostics tuning	Identify the cause of a performance issue - find the bottleneck & suggest remedial actions. Performance timelines - breakdown of response times into sub-components - comparison at various load levels to understand scalability of contributing components
T.3	Will the delivered automated application meet the performance requirements?	Performance assurance	Test whether an application provided by a third party vendor meets agreed SLAs and is acceptable to the client
T.4	Will the newly added functions of the application continue to meet our performance requirements?	Performance regression	Test to determine whether a new version of an application performs at least as well as the previous version
T.5	How will the application perform after it goes live? What is the Maximum client it can connect?	Load testing	Test the performance of an application under conditions of predicted peak load and ensure that business processes still work correctly
T.6	How to improve a company's in-house performance testing skills?	Training	Mercury public course, Mercury courses hosted at SQS / Client sites, bespoke training in the arts of performance testing.

T.7	At what point will we need new hardware to replace our current hardware? How much do we need to spend to ensure continued performance in the coming years?	Capacity Planning/Sizing/Scalability	Use performance tests simulating predicted usage of an application to inform the client's infrastructure purchasing plans. Scalability assessment - horizontal (better components) versus vertical (additional servers)
T.8	At what point does the application or infrastructure suffer catastrophic failure, or specified unacceptable performance?	Stress Testing	Find limits of application performance by applying load to the breaking point of the system and observing the root cause of failure
T.9	Will the application operate reliably under prolonged continuous load?	Stability/Soak testing	Test ability of an application to withstand long periods of load - likely to find problems with capacity, memory usage etc that do not occur with shorter tests
T.10	How will the application respond if everyone logs in at once?	Spike Testing	Test ability of an application to withstand sudden, short period of extremely high load. Likely to find problems with queue management, resource usage and integration of components
T.11	Will the product we are considering using in our system give the desired performance combined with all the other components?	Product Evaluation	Test whether a new system can scale to meet user demand. Proof of concept that a new technology is scalable
T.12	Does the application function correctly under real world conditions?	Functionality Under Load	Apply load and test all the business processes of an application
T.13	How can we simulate internet traffic coming from different locations worldwide?	Remote testing	Test execution based at a remote location to allow more realistic internet traffic simulation. This could include: Remote hosting of load generators, enabling a performance test to apply load from one or more remote locations. Penetration/security testing, where testers based remotely try to illegally access a system.
T.14	How do we reduce costs of performance testing whilst still mitigating our risks?	Offshore	Each of the above services can be wholly or partly performed at remote locations, allowing the benefits of an off shoring model to be gained
T.15	Which performance test tool should we buy to test the automated software?	Test tool recommendation/ evaluation	Analyze the system under test and decide the possible test tools that could be used to performance test given the technologies involved. Using proof of concept and knowledge of test tools, recommend to the customer the tool that best fits their needs
T.16	Which architecture should we use to ensure system performance? What are the different probes built to ensure performance?	System architecture & planning	Design & architecture of system components to ensure performance. Recommendations on choice of system components. Web page design/composition assessments - optimization of design to improve performance
T.17	How do we know that response times are acceptable, and when to upgrade infrastructure?	Production monitoring	Monitoring production systems to ensure that performance of live system stays within the required boundaries.

6. AUTHORS PROFILE

S. Velmourougan is presently working as Scientist at STQCIT, Chennai, STQC Directorate, Ministry of Information & Communication Technology, Govt. of India. He has obtained his B.E in Electronics and Communication Engineering and M.S in Software Reliability from Anna University, Chennai, India. He is an ISMS-Lead Auditor, Certified Ethical Hacker (CEH), Certified Information Security Professional (CISP), Certified Reliability Professional (CRP) and Certified Software Test Manager (CSTM). He is working in the field of application Security Testing and Information Security Management System. His experience comprises of Software Reliability Estimation, Reliability Allocation, System Reliability Analysis, Failure Analysis, and Reliability development/Growth testing. He has developed various reliability engineering tools to assess the reliability of software and hardware. He has published various research and technical papers in reputed journals, conferences and Magazines.

Dr. Dhavachelvan Ponnurangam is working as professor, Department of Computer Science, Pondicherry University, India. He has obtained his M.E. and Ph.D. in the field of Computer Science and Engineering in Anna University, Chennai, India. He is having around a decade of experience as an academician and his research areas include Software Engineering and Standards, Software Agents and Distributed Systems. He has published around 50 research papers in National and International Journals and Conferences. He is heading two research groups working towards to develop the standards for Attributes Specific SDLC Models & Evaluation and Business Intelligent Logic Based Management and Evaluation of Web Services.

Dr. Baskaran Ramachandran is working as the Assistant professor in Department of computer science, Anna University, Chennai. He has obtained his M.E. and Ph.D. in the field of Computer Science and Engineering in Anna University, Chennai, India. He is having around a decade of experience as an academician and his research areas include Multimedia and principles, Software quality engineering, Software Agents and Distributed networking. He has published around 50 research papers in National and International Journals and Conferences. He is the member of various forums. He is the editor and the reviewer in various journals. He is guiding research scholars working in area of software standards for Attributes Specific SDLC Models & Evaluation and Metric Based Efficient Traffic Management and A Multi-object Image Retrieval systems.