International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Performance Testing: Methodology for Determining Scalability of Web Systems

Deep Manishkumar Dave[1], Amit Bhanushali[2]

[1]Independent Researcher, Massachusetts, United States of America

[2]Independent Researcher,West Virginia, United States of America

Abstract: ***This research focuses on the critical role of performance testing, particularly load testing, in evaluating web systems' robustness under varying loads. It highlights the methodology centered on request and response times, using https://mu.ac.in/ as an example. The paper recognizes the broader significance of performance testing in life-critical and mission-critical domains. It presents key findings from a college website load testing case study, demonstrating the impact of increased user loads on response time and scalability limitations. The study underscores the importance of systematic load testing in identifying performance bottlenecks and advocates its adoption for ensuring optimal web system scalability.***

Keywords: Performance testing, Application performance, Cloud computing, Web testing, Web application, Load testing, Request time, Response time

1. Introduction

In the dynamic realm of the web, user expectations for swift and reliable experiences are paramount. The modern user, faced with a deluge of online options, is quick to abandon a website that takes too long to load. The vitality of swift web applications is especially pronounced in the context of today's businesses, where online presence is integral to operations. Every organization relies on the internet to conduct its online business, emphasizing the critical importance of optimal web application performance [10][7].

Performance, in this context, refers to the ability of a system to deliver information accurately and rapidly, even amidst high multi-user interactions or constrained hardware resources [13]. Recognizing this, the software development life cycle (SDLC) places considerable importance on software testing, a process aimed at discovering and eliminating software errors [2]. The role of software testing extends beyond bug detection; it is pivotal in enhancing software reliability and overall quality [1]. For web applications, this specialized domain is termed web testing, encompassing various types as depicted in Figure 1.

The crux of a successful product lies in the harmony of two essential elements—functionality and performance. Functionality encapsulates what the application enables users to accomplish, while performance refers to the system's ability to execute transactions rapidly and accurately, irrespective of the challenges posed by high user interactions or limited hardware resources.

Unfortunately, application failures due to performance-related issues persist, often stemming from a lack of robust performance testing methodologies. Pre-deployment performance testing is pivotal in preventing these issues, ensuring the availability, reliability, and scalability of applications in the real-world scenarios they are destined for.

This paper delves into the critical realm of performance testing, with a specific focus on load testing—a technique instrumental in evaluating the scalability of web systems. The study undertakes an experimental approach, subjecting a college website to varying loads simulated by virtual users. Through meticulous analysis of key metrics such as response time and throughput, the research unravels the impact of escalating concurrent users on system performance.

The ensuing sections of this paper unfold a comprehensive exploration of load testing concepts, metrics, and tools in Section 2. Section 3 meticulously details the experimental load testing methodology, followed by a thorough analysis of results in Section 4. Conclusions and avenues for future research are presented in Section 5. This study fundamentally underscores the indispensability of systematic load testing in not only determining but also enhancing the scalability of web systems, ensuring their resilience under the anticipated peaks of user interactions.

Functional Testing:
Functional testing serves the primary purpose of examining all links on a webpage, encompassing testing internal connections, validating outgoing links from each page, and identifying any broken connections. Additionally, it involves cookie testing, where cookies - small files stored on user machines for login sessions - are scrutinized. This testing method evaluates browser options by enabling or disabling cookies. Forms, crucial for gathering information from users and maintaining communication, are also subjected to testing. This includes verifying the creation, viewing, deletion, or modification of forms and thoroughly examining authentication on all fronts. Furthermore, functional testing delves into database testing to ensure data integrity within the web application. This comprehensive approach to functional testing ensures the robustness and reliability of various elements within the web application.

Figure 1: Web testing methods

2. Performance Testing Concepts

Performance Testing, often interchangeably referred to as Load Testing, constitutes a strategic process wherein an application undergoes simulated user interactions facilitated by a load-generating tool. The primary objective is the identification of system bottlenecks, with a central focus on testing for scalability, availability, and performance, both in terms of hardware and software considerations.

In this comprehensive evaluation, various resource aspects are scrutinized, including CPU usage, memory utilization, cache coherence, data consistency (pertaining to main memory, virtual memory pages, and disk), power consumption, and network bandwidth usage. These elements are systematically monitored and reported as integral components of the performance testing regimen.

Furthermore, the evaluation extends to critical metrics such as response time and usage patterns associated with routers, web servers, and application servers (app server). An exhaustive analysis for performance is not a one-time endeavor; rather, it needs to be a recurring application at each stage of the product development lifecycle.

Collectively, system performance emerges as a comprehensive figure of merit. This includes an evaluation from various angles, encompassing response time, throughput, availability, reliability, security, scalability, and extensibility. By considering these multifaceted aspects, performance testing becomes an indispensable tool in ensuring the robustness and optimal functioning of an application across diverse operational scenarios.

2.1. The need for the Performance Testing

In the contemporary landscape of e-business and the escalating migration towards cloud platforms, the demand for competitive web services is more pressing than ever. For a website to thrive in this competitive milieu, it must meet certain criteria: pages should load instantaneously, web transactions must be efficient and accurate, and downtime should approach zero. The consequences of downtime are not merely inconvenient but can be exorbitantly expensive. A Gartner report indicates that the average cost of unplanned downtime for a mission-critical application is approximately $100,000 per hour.

In the evolving landscape of online consumer and B2B marketplaces, the competition is fierce. Web-based applications need to seamlessly handle multiple simultaneous users, engaging in various transactions or interactions. To ensure the delivery of such service levels, enterprises providing these services must employ application load-testing tools. Highlighting the significance of this, a Jupitor Media Metrix consumer survey suggests that more than 46% of users abandon websites due to technical and performance issues.

The imperative for performance testing is even more pronounced in life-critical situations, particularly in systems integral to procedures like heart surgery or angioplasty. In the critical moment of an angioplasty, for instance, where a balloon is inflated inside an artery, any malfunction during the next 60 seconds could lead to a fatal heart attack. Another example lies in genome-based applications, such as drug discovery systems, where a malfunction in the process could have far-reaching consequences across generations.

In essence, the significance of performance testing extends beyond the realm of competition and cost implications—it is a critical measure ensuring the reliability, efficiency, and safety of systems that play pivotal roles in life-critical scenarios and groundbreaking scientific endeavors.

2.2. The Benefits of Performance Testing

Performance testing yields a multitude of advantages across business, project, process, and product dimensions. The following highlights key benefits:

- **Better Reliability:** Performance testing mitigates deadlocks, enhances response time, and ensures scalability, fault tolerance, and recovery from failures. This contributes to overall system reliability.
- **Shorter Time to Market:** For large enterprise applications, performance testing significantly reduces the time to market. By treating performance requirements as high-priority, non-functional aspects, the time-to-market is improved, leading to a considerable reduction in the test cycle and defects.
- **Memory Problem Management:** Performance testing effectively identifies and addresses memory-related issues such as leaks, overflows, data inconsistencies, and byte order violations.
- **Secure Software:** By detecting memory overflows and other resource vulnerabilities, performance testing plays a crucial role in ensuring the security of both web-based and desktop applications.
- **Benchmarking:** Performance testing enables the benchmarking of various architectural options, allowing for a comprehensive assessment of the Quality of Service.
- **Future Expansion Facilitation:** Accurate prediction of required system and network capacity, facilitated by performance testing, aids in planning for future expansions with precision.
- **Service Level Tests:** Performance testing supports the

testing of different service levels to effectively address challenges post-deployment, contributing to acceptance testing.

Hence, performance testing emerges not only as a quality assurance measure but as a strategic asset that enhances reliability, security, and overall efficiency while facilitating streamlined processes and future scalability.

2.3. Factors enabling Successful Load Testing

The execution of performance testing is often intricate, influenced by several factors that can either streamline or prolong the process. The complexities arise from intricate tools, limited experience, and inflexible testing tools. Successful load testing hinges on adequately addressing these challenges, necessitating well-trained testers proficient in tools, versed in the application domain, and possessing a foundational understanding of design, architecture, technology influences, and performance testing methodologies.

Key steps in preparing for a performance test include script preparation, workload modeling and scheduling, and script execution. The efficacy and success of load testing are significantly impacted by the support provided by the chosen load-testing tool. The following factors contribute to the success of load testing:

- **Testing at Varied Speeds:** Evaluating performance across different connection speeds reveals resource utilization disparities. However, this might limit the number of virtual users accessing a website simultaneously.
- **Testing on Multiple Browsers:** Single-browser load testing is insufficient. Ensuring error-free performance necessitates load testing on various browsers to comprehend diverse user experiences.
- **Creating Complex Scenarios:** Simulating authentic user experiences requires the formulation of intricate scenarios. The load testing company must design scenarios closely mirroring transactions performed by real users.
- **Extensive Scripting Possibilities:** A comprehensive test scenario demands a multitude of scripts to ensure thorough exercise of the system.
- **Clear Reporting:** Robust reporting encompassing error logs, response times, throughput information, resource utilization, and network monitoring is essential for optimizing system performance.
- **User-Friendly Tools:** The developed tools should be user-friendly and intuitive, reducing the cost and effort associated with load testing while maintaining effectiveness.
- **Performance Prediction:** Advanced capacity planning allows modeling system behavior and forecasting workload characteristics. However, predicting real-world performance requires a high-scale factor for accuracy.

The success of load testing relies on a holistic approach, encompassing technical proficiency, comprehensive testing strategies, and the deployment of tools that align with user needs and system intricacies.

2.4. Factors enabling Successful Load Testing

A load-testing tool serves as a dynamic simulator, replicating the actions of actual users through the utilization of "virtual" users. This tool comprehensively captures and analyzes the site's performance under varying loads, providing crucial insights into the experiences of virtual users. Typically, load-testing software is distributive, deployed across multiple servers operating simultaneously, with each server emulating numerous virtual users.

Often, load-testing tools from vendor companies include proprietary browsers tailored to the specific testing needs of individual clients. Concurrently, ongoing records detailing virtual users' interactions with the test site, encompassing response times, and encountered errors, are meticulously preserved for post-production analysis.

To enhance diagnostic capabilities, many testing companies offer remote monitoring of client websites, recording actual error messages experienced by virtual users for subsequent review. Detailed logs are generated, chronicling each user's experience, facilitating later comparisons with CPU and database testing data obtained during the test to pinpoint and resolve issues effectively.

One notable feature of load-testing tools is the ability to externally assess web-based applications from multiple points of presence. This evaluation helps identify if the quality-of-service provider's connectivity is contributing to system slowdowns. For instance, consistent network slowdowns at 6 Mbps when an expected bandwidth of 10 Mbps is specified might indicate issues such as overloading, underutilization, or improper usage of the network.

An invaluable attribute of load/performance testing tools is their capability to furnish insights into the performance of the client network's infrastructure. Firewalls, routers, and load balancers, interconnected in a network, may inadvertently create bottlenecks. For instance, a firewall might lack sufficient throughput to handle simultaneous users adequately. By simulating real user activity, load-testing tools prioritize the assessment of application and database performance under stress, providing comprehensive evaluations of system capabilities.

3. Application Performance

Performance objectives, typically centered around response time and throughput under specific workloads and configurations, constitute critical benchmarks. Adopting a proactive approach to performance testing, starting early in the software development life cycle and validating at various stages, including software requirements, is an established best practice.

Considered under the umbrella of non-functional testing, performance testing demands meticulous attention during the requirement specification stage. Here, performance objectives undergo analysis to ensure completeness, feasibility, and testability. Feasibility is assured through prototyping, simulation, or other modeling approaches. Transitioning into the design phase, performance

requirements are meticulously applied to individual design components, with simulation, prototyping, and modeling guiding the analysis. Throughout development and deployment, performance analysis remains a focal point at every testing level, necessitating the creation of test data tailored to performance testing scenarios. Modern microprocessors often employ profile-guided performance optimizations for tuning.

Memory, a critical resource for both system and application, demands careful management to optimize overall performance. Reclaiming memory after usage significantly influences system efficiency. Technologies like COM/DCOM, J2EE, or .NET incorporate built-in garbage collector routines utilizing popular algorithms such as simple mark-and-sweep, generation-based, or advanced train algorithms [5]. Given that memory-related issues, including leaks, overflow, and byte order violations, can lead to major performance problems and even security breaches, constant monitoring and reporting through diagnostic reports are imperative.

In the era of distributed and concurrent systems, including commercial multi-cores and Network-On-Chip architectures, memory-related challenges such as cache coherence and consistency become pivotal performance parameters. Addressing these concerns requires strategic measures, and automated tracking using hardware-software co-design proves advantageous [2]. Application performance is the aggregate outcome of these multifaceted aspects, necessitating well-planned tests that consider the intricate interplay of various performance parameters.

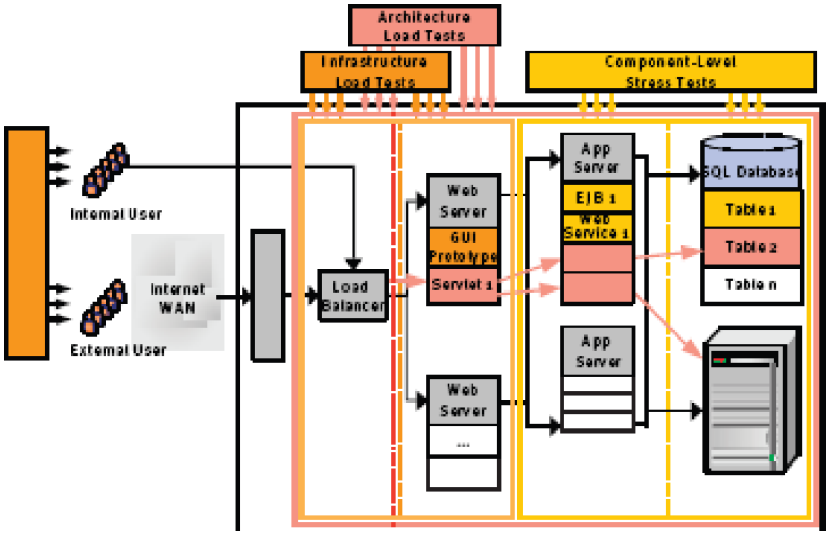

Figure 2: Stages of product development

3.1. Sources of Performance Issues - Concise Overview

Efficient testing necessitates a tester's awareness of primary sources of performance problems. The following outlines key sources of performance issues across various facets of a system and its architecture:

- **Technologies:** J2EE is esteemed for scalability; however, intensive threading, rich environments, and heavy transactions can induce performance overhead in Java/J2EE environments. Conversely, .NET and COM/DCOM face constraints from a heavy memory footprint, tightly coupled nature, and inadequate load balancing support.
- **XML Usage:** Widely used for interoperability, XML's storage, retrieval, and processing in the critical path can introduce delays. Navigation through XML using Document Object Model (DOM) incurs heavy memory use, while XPath navigator is lighter and faster for regular usage.
- **Database Considerations:** Placing a database server on a web server can lead to severe performance issues and security threats. The use of stored procedures can alleviate network traffic.
- **Language Impact:** While Java is considered performance-oriented, its use of synchronization statements between threads may cause deadlocks, a potential system breakdown. Servlets, comprising multiple threads, are susceptible to memory leaks and deadlocks.
- **Network Challenges:** Common performance problems include network traffic and communication delays. High Speed Ethernet (HSE) combined with high-speed H1 field bus protocol offers a comprehensive solution.
- **Protocol Awareness:** Intelligent leveraging of network protocols is crucial. For example, SOAP protocol may be slower and heavier than HTTP.
- **Wireless Protocols:** Bluetooth's short-distance, high-speed operation faces challenges like connection failures due to scalability issues or interference with other standards like Wi-Fi.
- **Batch Processing Consideration:** Utilizing batch processing, normalized data, and disconnected data objects like DataSet in .NET can significantly enhance network performance.
- **Internationalization Challenges:** Handling resource bundles in local languages may require more storage, and conversions may lead to memory issues. Careful design is needed for optimization in performance and scalability, particularly when dealing with internationalization considerations.

- **Security Impact:** Security features such as firewalls and data encryption/decryption can introduce performance problems, including increased access delays.
- **Server Platforms:** Efficient servers like DELL or HP integrity servers optimize performance, emphasizing the importance of their usage for enhanced system performance.
- **Algorithmic Impact:** Imaging algorithms, especially in medical imaging, demand careful consideration for performance and correct visualization effects. Striking a balance between performance and visualization effects is critical to avoid resource problems and system crashes.

3.2. Strategies for Performance Issue Remediation

To effectively address performance problems, consider the following strategic approaches:

- **Establishing a Comprehensive Test Lab:** Develop a test lab that replicates the actual deployment environment, complete with substantial data storage, and conduct thorough tests. While this approach closely simulates real scenarios, it may not perfectly emulate the geographically dispersed nature of applications like ATM banking, SAP, and wireless mobile network applications.
- **Infrastructure Enhancement:** Improve infrastructure by upgrading servers and transitioning operating systems, such as upgrading from Windows 2000 to Windows 2003. A case study demonstrated a 36% increase in server utilization for an average of 25-30 users, with no observed delays.
- **Leveraging Automation Tools:** Employ automation tools for performance, load, and stress testing. Utilizing standard performance/load test automation tools during the testing phase or earlier stages can significantly alleviate performance problems. Sometimes, functional tests...

These strategic measures provide practical and impactful ways to proactively manage and mitigate performance challenges across various applications and systems.

4. Related Work

Guo and Chen [5] proposed a performance testing model for web services, emphasizing testing capacity and automation. Their model includes a multi-machine joint testing approach for load balancing and a simulation approach for a realistic web services environment.

Büchler and Oudinet [3] introduced SPaCiTE, a security testing tool based on model checking for web applications. It identifies vulnerabilities and weaknesses in web applications by generating attacks and validating them against the system under test.

Vani and Deepalakshmi [17] discussed the importance of load testing in evaluating web application performance. Their work emphasized how load testing addresses quality of service (QoS) influences, response time, and throughput, meeting business requirements at different obligation stages.

Stoyanova [15] presented TASSA, an integrated testing framework for end-to-end testing of Business Process Execution Language (BPEL) orchestrations. TASSA supports automation of test condition generation, implementation, and supervision for web services defined with BPEL.

Sabharwal and Sibal [12] conducted a literature survey on web application testing methods from 2000 to 2014. Their comparative study focused on various testing techniques, with an emphasis on functional and performance testing.
Dhiman and Sharma [4] compared three performance testing tools—Apache JMeter, Grinder, and HttpRider-based on their response times. They highlighted the importance of performance testing for web services, considering factors like response time and scalability.

Lee and Chen [8] developed automated waiting mechanisms to enhance automated record-replay testing. These mechanisms dynamically specify waiting times, reducing errors and enhancing efficiency in artificial discrimination during test execution.

In summary, the related work showcases a variety of approaches, tools, and frameworks for automated, model-based testing of web applications, covering aspects such as performance, security, and functional testing.

5. Performance Testing

When a web application takes minutes to load, it becomes frustrating, especially when compared to other sites that download similar content in seconds. If someone attempts to log in to a web application and encounters a "server busy" message, it can be aggravating. Similarly, when a web application responds in some situations but goes into an infinite wait state in others, it becomes disconcerting. These issues are performance-related, and performance testing plays a crucial role in identifying and addressing such defects. Performance testing is instrumental in uncovering defects stemming from network bandwidth weaknesses, server-side limitations, operating system capabilities, and other software or hardware issues. Its purpose is to simulate real-world loading scenarios, such as an increase in the number of online transactions, data volume, or simultaneous users accessing the web application [11].

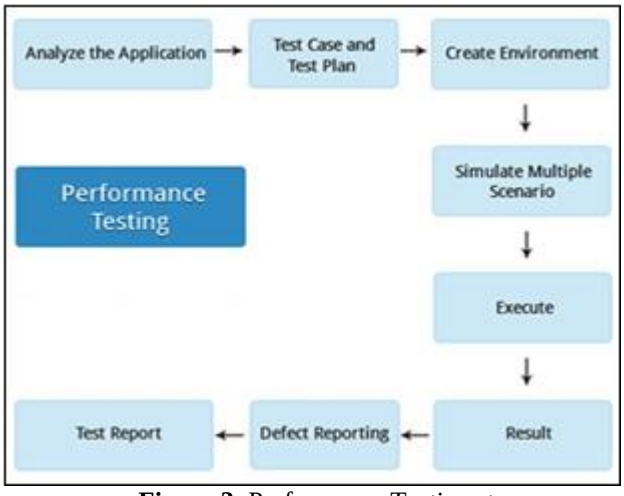

Figure 3: Performance Testing steps

The primary advantage of performance testing is its ability to govern the throughput and response time of a web application. Throughput measures the number of transactions per second an application can handle during the test, indicating the volume of transactions produced over time. It is influenced by factors such as the types of transactions being processed, specifications of the host computer, and processing overload in software [11]. Response time, on the other hand, is the time from sending a user request until the application acknowledges that the request has been completed, as detailed in the next section [14].

Performance testing tools are utilized to determine the time required for the system to execute a task [18]. It ensures that the non-functional requirements specified in the Software Requirement Specification (SRS) document are met by the system. Nowadays, website builders and developers recognize the essential nature of testing performance before deployment [19]. Performance testing encompasses load and stress testing, with load testing specifically focusing on response times. The main goal of load testing is to assess whether the application can withstand increasing loads on the server, a topic that will be explored in detail in the next section. Stress testing, while similar to load testing, goes further by pushing the server beyond its normal load limits. The primary objective of stress testing is to evaluate the system's behavior under extreme conditions. The following Figure 2 illustrates the main steps of performance testing [6].

6. Load Testing

Load testing stands out as one of the most widely used techniques for assessing performance. Its primary objective is to define the anticipated peak load conditions and the behavior of web applications. This testing method reveals crucial insights into system behavior while managing the specific load imposed by users [10]. The process of load testing involves a gradual increase in resources. Initially, the test loads the web application with a limited number of virtual users and progressively scales up from normal to peak, as illustrated below:

Load testing is instrumental in measuring the Quality of Service (QOS) performance of the application, relying on the actual behavior of customers. Interaction scripts are constructed using a script recorder based on customers' requirements. These scripts are then replayed by the load generator module, potentially adjusted via test parameters in contrast to the actual website.

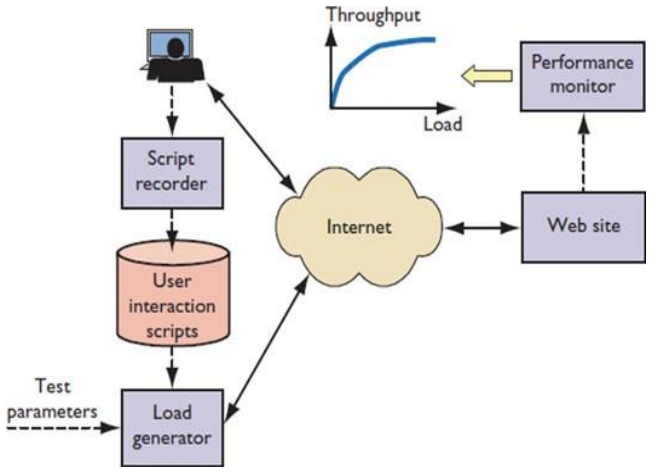

Figure 4: Load Testing steps

The primary key to Quality of Service (QOS) is the response time, which must be measured to understand how customers perceive different aspects such as keyword search times and page downloads. When defining response time, differences between the base HTML pages' download time and other components like images and ad banners should be considered. Response time for web applications varies based on factors such as the customer's internet service provider, the testing site IPS, bandwidth, and the network route for packet transmission from the customer to the testing website. Measuring response time within specific time windows and geographical locations provides a comprehensive understanding, acknowledging that response time is dependent on both time and space [10].

7. Conclusion

The landscape of performance testing has evolved significantly, especially with the proliferation of applications in critical domains like medical, healthcare, real-time, and mission-critical sectors. While metrics like response time and throughput may appear routine in typical scenarios, they present formidable challenges in the context of mission-critical applications and sophisticated technologies like .NET, J2EE, and XML. Ensuring quality demands adherence to standards, such as CMMi, FDA (Food and Drug Administration)—a requisite for medical and health applications—or industry-specific standards like MISRA, tailored for communication, automotive, aerospace, and real-time applications. The imperative is to initiate performance testing early in the product development cycle to enhance overall product quality. The rigorous demands of contemporary applications necessitate vigilance, standardized processes, and the incorporation of third-party tools that align with established quality benchmarks. As we navigate this evolving landscape, the importance of robust performance testing practices becomes increasingly evident, ensuring that applications not only meet but exceed the performance expectations of diverse and demanding sectors.

References

[1] OpenGL (2012), “OpenGL - The Industry Standard for High Performance Graphics”, www.opengl.org.
[2] H. Sarojadevi and S. K. Nandy (2011), "Processor-Directed Cache Coherence Mechanism – A

Performance Study", International Journal on Computer Science and Engineering, Volume 3, issue 9, 3202-3206.
[3] HP(2010),"HPBringsAffordablePerformanceTestingtotheCloud",HPwhitepaper,online-http://www.hp.com/go/loadrunnercloud.
[4] Borland (2006), "Choosing a Load Testing Strategy" , A Borland whitepaper. Tom Kelley (2006), "The Art of Innovation".
[5] R.L. Hudson and E.B. Moss (1992), "Incremental collection of Mature Objects", Proceedings of the International Workshop on Memory Management, Springer-Verlag, 388-403
[6] James S. Collofello (1988), "Introduction to software verification and validation", SEI curriculum module, CMU.
[7] GlenfordMyers(1969),"TheArtofSoftwareTesting",JohnWiley.
[8] Bathla R, Bathla S (2009) Innovative approaches of automated tools in software testing and current technology as compared to manual testing. Glob J Enterp Inf Syst 127–131
[9] Barr ET, Harman M, McMinn P, Shahbaz M, Yoo S (2015) The oracle problem in software testing: a survey. IEEE Trans Softw Eng 507–525
[10] B¨uchler M, Oudinet J (2012) SPaCiTE—web application testing engine. In: IEEE Fifth international conference on software testing, verification and validation
[11] Dhiman S, Sharma P (2016) Performance testing: a comparative study and analysis of web service testing tools. Int J Comput Sci Mobile Comput 507–512
[12] Guo X, Chen Y (2010) Design and implementation of performance testing model for web services. In: 2nd International Asia conference on informatics in control, automation and robotics
[13] Kaushik M, Fageria P (2014) A comparative study of performance testing tools. Int J Adv Res Comput Sci Softw Eng 1300–1307
[14] Khan R, Amjad M (2016) Performance testing (load) of web applications based on test case management. Perspect Sci 355–357
[15] Lee S, Chen Y (2018) Test command auto-wait mechanisms for record and playback-style web application testing. In: 42nd IEEE international conference on computer software and applications
[16] Maheshwari S, Jain DC, Maheshwari MS (2012) A comparative analysis of different types of models in software development life cycle. Int J Adv Res Comput Sci Softw Eng 285–290
[17] Menascé DA (2002) Load testing of web sites. IEEE Internet Comput 70–74
[18] Pressman R, Lowe D (2008) Web engineering: a practioner's approach. McGraw-Hill, Inc
[19] Sabharwal S, Sibal R (2015). A survey of testing techniques for testing web based applications. Int J Web Appl
[20] Sarojadevi H (2011) Performance testing: methodologies and tools. J Inf Eng Appl 5–13
[21] Sharmila MS, Ramadevi E (2014) Analysis of performance testing on web application. Int J Adv Res ComputCommun Eng 2319–5940
[22] Stoyanova V (2013) Automation of test case generation and execution for testing web service orchestrations. In: IEEE Seventh international symposium on service-oriented system engineering
[23] Teoh SH, Ibrahim H (2017) Median filtering frameworks for reducing impulse noise from grayscale digital images: a literature survey. Int J Future ComputCommun
[24] Vani B, Deepalakshmi R (2013) Web based testing—an optimal solution to handle peak load. In: International conference on pattern recognition, informatics and mobile engineering (PRIME)
[25] https://loadfocus.com/blog/2013/07/04/what-is-throughput-in-performance-testing. Have been visited at 29 Nov18
[26] https://loadfocus.com/blog/2013/07/03/what-is-response-time-in-performance-testing. Have been visited at 02 Dec 18

Author Profile

Deep Manishkumar Dave, IEEE Member
OCR ID 0009-0000-3946-8944

Deep Manishkumar Dave, currently at LTIMindtree and is involved with Johnson & Johnson's Depuy Synthes Orthopedics Division. He is an acclaimed Digital Transformation and IT System Reliability Expert. With an impressive track record in Industrial IoT (IIoT) and digital strategy, Deep has significantly enhanced operational processes and patient outcomes in the medical device manufacturing industry. Deep's mentorship and training of junior IIoT engineers highlight his commitment to future talent development. His accolades, including the Young Achievers Award and a Global Recognition Award, reflect his impact and dedication to the field. Deep's influence extends beyond his immediate professional sphere, evidenced by his contributions to global forums and sustainable manufacturing practices, along with advocacy for diversity and inclusion in tech. Deep holds a Bachelor's in Mechatronics Engineering and a Masters in Engineering Management, providing him with a robust foundation in both technical and managerial aspects of technology. His passion for knowledge extends to research and academia, having authored multiple research papers on pivotal topics such as digital transformation, neural manufacturing, Industry 4.0 and 5.0, IT system reliability, and human-robot collaboration. Deep's expertise is further recognized through his role as a reviewer for various scientific journals. He has also contributed numerous technical articles to platforms like Dzone and other prominent publications. His role as a judge in the Globee Business and Leadership Awards showcases his expertise and recognition in the field. As a senior coach and approved mentor on the ADPlist organization, Deep demonstrates his commitment to guiding and nurturing future talent in the technology sector.

Amit Bhanushali, IEEE Senior Member
OCR ID 0009-0005-3358-1299.

Amit Bhanushali is a highly accomplished software quality assurance professional with over 22 years of experience in the IT industry. He earned his Master's in Business Data Analytics from West Virginia University in 2017. Based in West Virginia, USA, Mr. Bhanushali is a Senior IEEE Member and has significantly contributed to software testing research and practice. His expertise spans automation testing, performance testing, DevOps, and CI/CD implementation. He has also led testing efforts in complex cloud environments. In addition to testing, Mr. Bhanushali has authored several articles exploring cutting-edge topics like artificial intelligence and machine learning. His published research demonstrates his thought leadership and impact on software quality engineering. Mr. Bhanushali's accomplishments have been recognized through prestigious

appointments. He serves as a reviewer for the Elsevier journal and has been a hackathon judge. His contributions were further honored in 2023 when he received the International Achievers' Award. With his sustained record of excellence across software development, testing, and research, Mr. Bhanushali continues to be an influential leader in his field.