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

Selecting Effective Metrics for Evaluating and Measuring Software Application Performance

Krishna Mohan Pitchikala

Abstract: ***The success of a software application depends on how well it measures up against Key Performance Indicators (KPIs). User Acceptance Testing (UAT) is the final testing stage where end - users assess the software to confirm that it meets their needs and is ready for release. While UAT plays a crucial role in determining whether the software application meets user expectations, it alone is not sufficient either to guarantee a software's success or to evaluate the application's performance. To ensure ongoing success, it is essential to have metrics that continuously monitor and evaluate the application's performance throughout its lifecycle. This continuous measurement helps proactively identify and fix issues, ensuring that the software consistently meets the required standards. Performance evaluation in software development is complex due to the lack of universal guidelines for selecting performance metrics. These metrics vary depending on the specific characteristics of the application and the business goals it aims to achieve. Choosing the right set of metrics is crucial for accurately assessing how well the software performs. These metrics should be tailored to the unique aspects of the application and aligned with the objectives of the business. This paper explores the fundamental principles and approaches that guide the selection of relevant metrics for evaluating and measuring software performance. By understanding these principles, organizations can better ensure that their software applications not only meet functional requirements but also deliver a reliable and optimal user experience.***

Keywords: software performance, key performance indicators, user acceptance testing, performance metrics, software evaluation

1. Introduction

Metrics in the context of software application refers to a quantifiable measure that is used to evaluate the effectiveness of both the software development process and the software itself. They provide insights into various aspects such as quality, performance, reliability, and efficiency. They are crucial for developers, managers, and other stakeholders to make informed decisions, or to identify areas that need improvement during the software development life cycle (SDLC).

Metrics can be collected at every stage of the SDLC, with each type serving its own specific purpose. When the right metrics are identified, implemented, and monitored, they support data - driven decision - making. In software applications, metrics can be broadly classified into five types: Security, Process, Performance, Code Quality, and Usability, as shown in the figure 1.

In this paper, we will focus on performance metrics, which are used to evaluate the efficiency and effectiveness of software. Performance metrics help us understand how software behaves in different situations, allowing us to make improvements. As applications evolve, these metrics give developers and stakeholders the information they need to make smart decisions about optimizing performance, managing resources, and maintaining system health. The challenge, however, is selecting the right metrics that capture the most crucial aspects of performance, since different metrics highlight different areas.

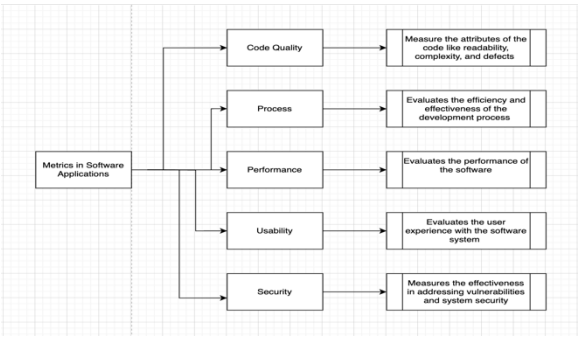

Figure 1: Metrics in Software Applications

Selecting performance metrics begins with a clear understanding of the software's purpose, its users, and the business goals it aims to achieve. The metrics must align with these goals to provide meaningful insights into the software's performance. Performance metrics are generally divided into two types: operational metrics and business - oriented metrics. The main challenge is finding the right balance between these two, which depends on the type of software application. Balancing operational metrics (like response time, availability, and error rates) with business - oriented metrics (like user satisfaction, retention rates, and business impact) is crucial for getting a complete view of the software's performance.

2. Performance Metrics

2.1 Operational Metrics

Operational metrics are the key performance indicators of any software applications. These metrics are used in measuring the internal, system - level performance of the software. These metrics are primarily concerned with the efficiency, speed, stability, and resource utilization of the software. They help developers and IT teams optimize the software’s technical performance. The following are the main operations metrics that every software application developer should consider monitoring and improve the application efficiency

2.1.1 Availability

Availability is the amount of time a system is accessible and usable by its users. In other words, it tells us how often the system is up and running. Critical systems often boast availability rates as high as 99.9999%, meaning they are almost always available. Tracking availability is crucial because it shows how long an application is accessible or not to users. There are different ways to measure this. One can assess the availability of an entire system or just a single API.

A simple way to measure availability is by calculating the uptime, which is the time the system is working, as a percentage of the total time. In basic terms, availability is the amount of time the system is running divided by the total time, which includes both the time it's running (uptime) and the time it's not (downtime) [6].

$$Availability = \frac{\text{Uptime}}{\text{Uptime} + \text{Downtime}}$$

Another approach is to use "canaries, " which are small groups of test users who simulate real user activity. These canaries regularly interact with the critical parts of the software to ensure everything is functioning as expected. If they encounter issues, it helps catch problems early.

For a single API, availability can be measured by comparing the number of successful requests to the total number of requests made [5].

2.1.2 Latency

Latency in software systems is the time it takes to send information between a client and a server. It's usually referred as the is the duration that a request is waiting to be handled. Keeping the latency low is important because high latency can make applications slow to respond. High latency can cause communication issues, making applications slower and less responsive.

By measuring latency, developers and network architects can identify where delays are happening, whether in data transmission, or somewhere else. This helps them fix specific problems and improve overall performance. In applications like online games, stock trading, and live broadcasts, where quick response times are crucial, managing and reducing latency is essential to ensure a smooth user experience.

2.1.3 Response times

Often people assume response time and latency are same, but they are not. Latency refers to the time it takes for information to travel between a client and a server. Response time, on the other hand, is the total time it takes from the moment a client sends a request until it receives a response back. Response time as a combination of two factors:

- **Latency**: This is the time the message spends in transit between two points, such as traveling across a network or passing through different gateways.
- **Processing Time**: This is the time it takes for the server to process the request, which might involve translating data, adding extra information, or performing other tasks.

Response Time = Latency + Processing Time.

If the processing time is very short, which is usually the case in well - designed systems, the response time might feel almost the same as the latency. However, it's important to remember that they are technically different. For accuracy, it's best to use the correct terms and not mix them up.

2.1.4 Error rates

Error rates in software applications refer to how often errors, bugs, or failures happen when the software is being used. These issues can include the application crashing, giving incorrect results, responding slowly, or having security weaknesses. Tracking these error rates is important because it helps ensure the software is of good quality and works well. High error rates often mean there are problems in the code or design that need fixing. By monitoring and fixing errors, developers can improve the user experience, make the software more reliable, and ensure it meets security and performance standards. Regularly tracking error rates also helps in preventing future issues and continuously improving the software. It can be calculates using the following formula:

Error Rate = (Number of requests with errors / Total number of requests) * 100%

In some cases, we can directly monitor the HTTP - 500 status codes returned by APIs to users instead of the percentage. This helps us in identifying all the cases where the server couldn't process a request and return a response, often due to improper error handling. By tracking these failures, we can fix them and improve the system's reliability.

2.1.5 Resource Utilization

Resource utilization in a software application refers to how much of the system's resources—such as CPU power, memory and storage—the application uses while it's running. Tracking this utilization is essential because it allows developers to identify inefficiencies, optimize performance, and ensure the application runs smoothly under various conditions. It also plays a critical role in scalability, helping to plan for future growth by ensuring that the necessary resources are available to handle increased traffic without compromising performance.

When running software on our own servers, monitoring CPU and memory usage on each server is crucial to ensure sufficient space and smooth operation without exhausting available resources. Tracking resource usage also helps in planning and estimating new resources that might be needed or cutting costs if we have allocated more resources than necessary. In cloud environments, where costs are based on usage, keeping an eye on resource consumption helps in managing expenses by avoiding waste and making informed decisions about scaling resources up or down. Monitoring resource usage also contributes to system stability, preventing slowdowns or crashes caused by overuse. This leads to a better user experience, as applications that efficiently use resources tend to be faster and more reliable. Overall, tracking resource usage is essential for improving performance, managing costs, maintaining system stability, and providing a good user experience.

2.2 Business oriented metrics

While operational metrics provide insights into the technical performance of a software application, offering developers the information they need to enhance the application and enabling business stakeholders to make informed decisions. Business - oriented metrics help business owners to understand how well the software is supporting their overall goals. These metrics focus on the impact the software has on the business, including the number of users, its contribution to revenue, customer satisfaction, and overall efficiency.

Business - oriented metrics can vary depending on the type of business, but in this discussion, we will focus on two commonly used metrics that apply across all types of software applications. These metrics are crucial for evaluating how effectively the software aligns with business objectives and drives success.

2.2.1 Request Volume

The Request Volume metric measures how many requests are made to a software application over a specific period. A "request" is any action that a user or another system asks the application to perform, such as loading a page, submitting a form, or making an API call to retrieve data from a server. This metric is important because it helps us understand how much the application is being used. It can be used in several ways:

- **Monitoring Performance**: By tracking request volume, we can see if there are any unusual spikes or drops in activity. For example, if the application suddenly becomes slow or crashes, we can check the request volume to see if a high number of requests might have caused the issue.
- **Capacity Planning**: As the application grows and attracts more users, the request volume will likely increase. By monitoring this metric, we can plan and make sure we have enough resources (like servers or bandwidth) to handle the extra load. If the request volume decreases, we might consider scaling back resources to save money.
- **Understanding User Behavior**: Request volume can also give insights into how users interact with the application. For instance, we might see higher request volumes during certain times of the day or after releasing a new feature.

The request volume is typically calculated by counting the number of requests received by the application over a specific period, such as per second, minute, hour, or day. This counting is usually done automatically by the application's server or by using monitoring tools that track and record every request made to the application.

2.2.2 Support Ticket Volume and Resolution Time

Another important metric for software applications is the Support Ticket Volume and its Resolution Time. Support Ticket Volume refers to the total number of support requests and outgoing emails handled by the customer support team within a specific time frame. These requests could be issues, questions, or help users need with the software. Resolution Time is the time it takes to resolve a support ticket, starting from when the ticket is created until it’s marked as resolved. Tracking these metrics is helpful because if there are many tickets or if it takes a long time to resolve them, it could mean there are problems with the software or that customer support isn't efficient.

These metrics are also useful for making important business decisions about the software, like deciding which new features to prioritize or finding resources to fix recurring issues that generate a significant number of support tickets.

These metrics can be tracked using dashboards, which allow teams to monitor the number of tickets and how quickly they are resolved.

3. Effective Strategies for Performance Measurement

When evaluating the software application performance, it's important to use metrics that match the goals of the project. Key areas to focus on include performance, quality, usability, and maintainability, which together give a full picture of how well the software is working. Tracking these metrics helps identify problems in the software and make informed decisions to fix them, ensuring the best possible experience for the user. Here are a few strategies that one can use to measure the performance of a software application.

1) **Align Metrics with Business Needs:** Start by choosing performance metrics that align with the business's goals. For example, in an e - commerce site, response time and uptime are crucial for user engagement and sales, while backend systems might prioritize resource use.
2) **Set Clear Guidelines and Limits:** Once the important metrics are identified, set clear benchmarks based on best practices, past performance, and user expectations. This helps in detecting any performance issues before users experience them.
3) **Monitor and Analyze Continuously:** Performance evaluation should be ongoing. Use monitoring tools to track how the application performs over time and address issues as they arise. Regular analysis of performance data is crucial for application success and can reveal patterns and areas for improvement.
4) **Regularly Review and Update Metrics:** As the application evolves, so should the performance metrics. Regularly reviewing and updating these metrics ensures they remain relevant and effective, involving both technical and business teams to balance different needs.

4. Conclusion

To accurately determine if software applications are meeting their goals and business targets, it's crucial to choose the right performance metrics. By using a mix of these metrics, organizations can get a full picture of how their software is performing and make informed decisions on any necessary changes. The key is to select metrics that match the specific needs of the software and business, and then regularly measure and track them. This approach allows software teams to spot and fix issues early, improve user experience, and increase the software's positive impact on the business.

Performance evaluation isn't a one - time task; it's an ongoing process that requires teamwork between technical and business teams. Regular analysis is essential for building a

strong performance measurement system that supports data - driven decisions and continuous software improvement.

References

[1] https: //www.browserstack. com/guide/what - is - software - metrics
[2] https: //stackify. com/track - software - metrics/
[3] https: //en. wikipedia. org/wiki/Software_metric
[4] https: //www.researchgate. net/publication/333505554_A_Study_on_Software_Metrics_and_its_Impact_on_Software_Quality
[5] https: //docs. aws. amazon. com/whitepapers/latest/availability - and - beyond - improving - resilience/measuring - availability. html
[6] https: //www.ni. com/en/shop/electronic - test - instrumentation/add - ons - for - electronic - test - and - instrumentation/what - is - systemlink - tdm - datafinder - module/what - is - rasm/what - is - availability - . html?srsltid=AfmBOoodulPM - T6Ik65mGMQRLxYtM4m10YloTh - R69CzqHQFYESkDswn
[7] https: //stackoverflow. com/questions/58082389/what - is - the - difference - between - latency - and - response - time
[8] https: //adapty. io/glossary/error - rate/
[9] https: //www.globallogic. com/uki/insights/blogs/which - software - metrics - to - choose - and - why/
[10] https: //www.researchgate. net/publication/331080288_Software_Performance_Testing_Measures