International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 05 July 2023 **Revised:** 25 August 2023 **Accepted:** 15 September 2023*

Streamlining Software Development in Enterprises: The Power of Metrics-Driven Development

B Gomathi[1], Manimegalai R[2], Srivatsan Santhanam[3], Ravikiran Krishnaprasad[4]
[1]Department of Computer Science and Engineering
PSG Institute of Technology and Applied Research
Coimbatore, India
e-mail: gomathi.babu@gmail.com
Department of Computer Science and Engineering
PSG Institute of Technology and Applied Research
Coimbatore, India
e-mail: drrm@psgitech.ac.in
SAP Labs India Pvt Ltd
Bengaluru, India
e-mail: srivatsan.santhanam@sap.com
SAP Labs India Pvt Ltd
Bengaluru, India
e-mail: ravikiran.vidhyaranya.krishnaprasad@sap.com

Abstract— In today's fiercely competitive market, software companies must prioritize the activities in the Software Development Life Cycle (SDLC) to produce high-quality software and stay ahead of competitors. The exponential growth of Cloud computing enabled technologies has prompted companies to adapt their software development processes based on Cloud, as it offers instant access to essential resources. However, traditional software metrics-based approaches fall short when applied to Cloud computing-based software development. In order to address this challenge head-on, this paper presents an approach called Metrics-Driven Development (MDD) tailored specifically for enterprise Cloud development which is also known as full-stack development. In order to support informed decision-making in the age of Cloud computing, the main goal of this research is to evaluate the functionality and quality of software using several metrics. Furthermore, MDD plays a pivotal role in improving the software quality, performance, and efficiency within the realm of Cloud computing. Based on the empirical experiments and observation, it is evident that Metrics-Driven Development is an invaluable approach for enhancing efficiency and effectiveness in enterprise software development.

Keywords- Cloud Computing, Software Metrics, Architectural Metrics, User Metrics, Cloud-based Software Development Life Cycle.

I. INTRODUCTION

Cloud services provide virtualized and dynamically scalable resources over the Internet, and their adoption in software applications is on the rise. The Cloud environment significantly impacts software development practices, as applications may require extensive maintenance due to poor software quality or neglect. Therefore, recognizing the importance of software quality measurement throughout all phases of software development is crucial. Metrics play a vital role in measuring software quality with the primary goal of enhancing the performance and maintainability of the software. However, the migration of legacy software to Cloud computing models, which offer flexible pay-per-use options, is gaining popularity. Traditional metrics mainly focus on conventional software applications, while semantic metrics pose challenges in determining actual performance since they rely on data gathered during the development stage. It is suggested to gather attributes from the design, implementation, monitoring, and maintenance phases in order to overcome the above-mentioned challenges and for better understanding during upgrades.

The collaboration of numerous teams and departments is necessary for managing large-scale corporate software development projects. Enterprise Metrics Driven Development (EMDD) uses metrics to assess how well software development processes are working, track those metrics, and use them to make informed decisions and streamline procedures [1, 2]. Identifying and measuring the key effects of the development process is a significant challenge for enterprises. This work aims to address this problem by identifying the most relevant Key Performance Indicators (KPIs) for enterprise software development projects done across organizations. It provides guidance on how to measure and interpret these metrics to improve project management and decision-making. Business metrics are deployed by organizations and Cloud service providers to facilitate informed decision-making and operational evaluation. Business metrics are quantitative measurements used to monitor business operations and analyze performance. For Cloud-based full stack developers, managing software development projects in Cloud computing world can be daunting, especially when faced with limited time and resources [3]. Prioritizing tasks and making informed decisions becomes challenging. Furthermore, identifying key metrics to track project progress and areas for improvement can be difficult. This research work aims to identify the most relevant metrics for developers to track during software development projects and provide insights on interpreting and utilizing the metric data to enhance project outcomes. An SLA is an agreement between an enterprise and its clients that defines the level of service to be provided. Technical metrics are crucial components of Service Level Agreements (SLAs) in Cloud environments to evaluate and quantify the primary technical attributes of offered services. These metrics can be categorized into (i) high-level metrics, that are easily understandable for end users, and (ii) low-level metrics, that focus on fundamental technology characteristics. Cost metrics play a significant role in measuring profitability and inform pricing strategies based on resource usage and time. Predicting future Cloud expenses based on usage patterns facilitates cost analysis. Infrastructure metrics provide insights into the behavior of platform-managed services, ensuring reliability, availability, and performance. Service Level Indicators (SLIs) are specific metrics used to measure service performance, while Service Level Objectives (SLO) set the target goals for those metrics to ensure the agreed-upon service levels are met. This research centres around metrics as a means of measuring Service Level Indicators (SLIs), which are essential for establishing Service Level Objectives (SLOs) to uphold Service Level Agreements (SLAs) [4, 5] with clients. These metrics aid both service providers and customers in impartially assessing the level of service and identifying areas for improvement. The Rest of the paper is organized as follows: review and analysis of several software metrics are presented in Section 2. Section 3 describes the proposed Metrics-Driven architecture in relation to Cloud services, and Section 4 covers the implementation and testing of the proposed architecture, including results and evaluation. Section 5 summarizes the paper by indicating directions for future work.

II. LITERATURE SURVEY

The shift from traditional to Cloud-based software development has introduced numerous challenges in ensuring software quality. The increased utilization of Cloud computing resources has led to a higher probability of failures affecting applications running in the Cloud. To address these challenges, developers and software companies have focused on analyzing metrics to effectively detect and synthesize issues, in order to produce high-quality software. It is crucial to identify and assess relevant metrics that can help identify and evaluate important outcomes in order to improve project management and decision-making in Cloud-based software development. Researchers have been actively involved in software metrics research for several decades, and the most notable approach [1] is Enterprise Metrics Driven Development. This approach aims to define measures that enhance the production process and product development. Various techniques on extended frameworks and paradigms using the EMDD model are presented in this Section. Numerous studies have been done recently on the application of metrics to assess and enhance software development processes. They have demonstrated the benefits of employing metrics, including enhanced software quality, reduced development time, and increased transparency and accountability. For instance, metrics such as code complexity, test coverage, and defect density have been proposed and utilized to evaluate software quality, enabling the identification of areas for improvement and tracking progress over time [3]. Additionally, various approaches have been developed to integrate metrics into the software development process, such as agile development, DevOps, and continuous integration and delivery. These methods seek to enhance collaboration and communication among development teams, speed up and improve the delivery of software, and lower the possibility of mistakes and faults.

Losup et al [3] have analyzed the performance of Cloud computing services for loosely coupled scientific computing workloads. The performance evaluation of the commercial Cloud environment is tested against scientific workload and concluded that the performance of the commercial Cloud is low, but it is the better solution for scientists who require computation resources for instant usage temporarily. Several end users can access the same service using multi-tenant SaaS, however they may have access to various levels of Quality-of-Service (QoS). Also, they must meet SaaS optimisation objectives such as lowest resource costs and optimal system performance. A QoS-driven approach that enables service selection is proposed in [6] by using Multi-tenant SaaS Optimiser (MSS) for multi-tenant software applications deployed in the Cloud. Integer programming is used to assist SaaS developers in identifying the optimal services for a multi-tenant SaaS that meet various degrees of end-user QoS requirements while achieving the optimization goals of SaaS providers. In order to cater to the needs of customer requirements, a statistical approach is suggested to evaluate the relationship between Cloud platforms, software development procedures, and big data applications. The processing and storing capacity of the Cloud environment is taken into account while examining the anomalies and flaws in the Cloud environment in order to avoid performance deterioration of big data applications and Cloud computing. Resource planning and SLA design both rely on the outcomes of the work proposed in [7]

It is important to think about the infrastructure that hosts the service as well as the service's performance, dependability, security, and availability. Consumers of Cloud services should be concerned about QoS since they depend on the Cloud service provider to fulfill the promised level of service quality. QoS aids Cloud service companies in determining the ideal balance between operational expenses and QoS standards. With the aid of QoS, service providers can strike a balance between providing services at a level that is acceptable to customers and keeping operating expenses for the delivery of the service within acceptable levels [8]. The efforts required to produce new software and the quality of Cloud services can be greatly reduced by an automated software defect detection model that can evaluate the overall quality of software and identify the components that can cause problems. The proposed approach in [9] discovers meaningful metrics by combining various filters and wrapper methods. One of the important features of the proposed technique is the parallel architecture of a hybrid software defect predictor, which is created and tested in order to process massive amounts of software metric data efficiently in a Cloud context. By combining Fisher and Maximum Relevance (FMR) filters with an Artificial Neural Network (ANN)-based wrapper, two unique hybrids are created in the parallel architecture. For all parallel models, evaluations are performed using actual datasets of software that are prone to defects. The use of logistics services in the area of Cloud powered manufacturing has become the new normal. The authors have examined the qualities of logistics services and created a mathematical framework for choosing them. Simulated tests were run to determine the suggested model's functioning and outcomes. It was asserted that the suggested methodology in [10] would optimize task delivery time in a Cloud manufacturing setting. The scope of this model's use, however, is constrained to logistics domain.

The customer-related metrics are such as downtime, cost, and reliability based on the suitable Cloud infrastructure selected [11]. The stochastic model is proposed to compute the dependability-related metrics between various Cloud infrastructures. In [11], infrastructure-related metrics alone are considered. It is better to evaluate the metrics of time, throughput, and CPU usage to know about the performance of various applications in a Cloud environment. Using a Cloud-based solution that focuses on QoS aspects such as defect detection capabilities, resource consumption, and pricing may make software unit testing simpler. A new framework for Hadoop and cluster customization is developed to offer customers the most accurate and suitable configuration for improved performance in terms of time and cost [12]. In addition, a clear pricing strategy that matches customer expectations and boosts service providers' net profits is also proposed in [12]. The suggested architecture includes a security mechanism for role-based access control that safeguards the data and code of various clients. Experiments conducted in a Cloud environment clearly show the effectiveness of the suggested architecture.

Self-adaptive resource allocation for Cloud-based software applications is recommended, to provide great service quality and low resource costs. The machine learning-based model proposed in [13] can be trained using historical data. By using workload and resource allocation data as inputs, it can forecast QoS values. Genetic algorithms can be used to automate online resource allocation decision-making in order to discover a feasible resource allocation strategy leveraging the QoS model. Using the Rubis benchmark, it was determined that QoS model accuracy was over 90% and resource consumption has improved by 10% to 30%. By combining an iterative QoS prediction model with a runtime decision method based on Particle Swarm Optimisation (PSO), a self-adaptive resource allocation system for Cloud-based software applications is created. Three successive QoS models are trained using Support Vector Machine (SVM), Nonlinear Regression Analysis Program (NLREG), and Classification and Regression Trees (CART) [14]. It has been demonstrated through testing that the iterative QoS model forecasts service levels more precisely than the traditional QoS model. In a later stage, an iterative QoS model and the PSO algorithm are merged to perform online automatic resource allocation. An improved optimisation approach for service composition that takes Service Level Agreement (SLA) constraints in the Cloud environment into consideration is proposed by Yaghoubi et al [15]. The objective function is defined by the weighted sum of QoS attributes based on preferences given by users. This optimisation method selects the suitable service to meet user requirements, while Cloud services with identical functionality may be available in a variety of service levels.

III. THE PROPOSED CLOUD BASED METRICS DRIVEN DEVELOPMENT

In this work, a Cloud-based metrics-driven development architecture is proposed as shown in Figure 1. The proposed approach can be used in enterprises to generate real time, continuous monitoring metrics from software models. This architecture leverages the capabilities of Amazon Web Services (AWS), an extensive Cloud computing platform that
provides a range of services to help businesses with their software development. The architecture contains with a front-end service built using React JS, running on an EC2 instance. This front-end service allows customers to interact with UI and perform the activities. It communicates with the backend services through API calls. The API Gateway, a managed service provided by AWS, acts as the proxy between the customer and the backend service and facilitates secure and efficient communication. Customers can access their applications through the presentation layer, facilitated by a

Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

REST API created in Amazon API Gateway. API Gateway resources are programmable elements that enable the routing of API calls to the appropriate backend service endpoints. This ensures seamless integration and interaction between the front-end and backend components of the architecture. The API Gateway also offers additional features such as authentication, rate limiting, caching, versioning, monitoring, and documentation, enhancing the overall management and performance of the API.

A service mesh is employed in the proposed architecture to manage internal service-to-service communication. The service mesh acts as a dedicated infrastructure layer, providing consistent and scalable solutions for traffic management, service discovery, load balancing, security, and observability. The backend service interfaces with the front-end service, API Gateway, and other components by utilizing certificate-based authentication. This ensures that only legitimate requests are processed, enhancing security and maintaining the integrity of service-to-service communication [16]. Data persistence is handled by AWS RDS, a reliable and scalable Relational Database Service. The backend service utilizes MySQL to persist data, ensuring data availability even in the event of backend service downtime. Amazon RDS simplifies the setup and management of relational databases, offering a cost-effective managed service with features such as software patching, backups, recovery, and automatic failure detection. Access to database instances is restricted to maintain a controlled customer experience, and backups can be automated or manually initiated through backup snapshots. The proposed architecture combines various AWS services to create a Cloud-based metrics-driven development approach in enterprises. It uses front-end and backend services, API Gateway, service mesh, and AWS RDS to facilitate efficient and secure software development processes. Enterprises can gain valuable insights for enhancing the development process, improving software quality, and making informed project management decisions by generating metrics from software models using the proposed architecture.

A. Dashboards and Metrics

In the context of streamlining software development in enterprises, leveraging metrics-driven development can significantly improve service reliability and meet Service Level Agreements (SLAs), Service Level Indicators (SLIs), and Service Level Objectives (SLOs). The proposed architecture collects the required metrics through Kibana, Sysdig, and New Relic dashboards collectively and helps in addressing SLA, SLI, and SLO challenges.

B. Data Collection

The proposed architecture for metrics-driven software development collects various metrics from different sources:

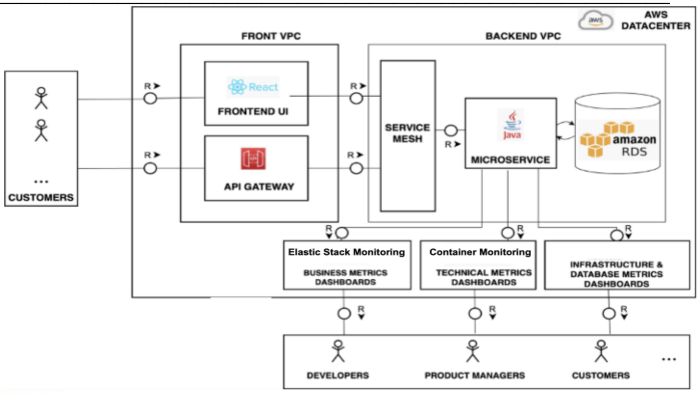

Figure1. Architecture of the Proposed Cloud based Metric Driven Development

(i) Business Metrics with Kibana: Kibana or its equivalent tool for data visualization of collected operational data is used to collect business metrics related to service performance, user behavior, and overall system health. These metrics can include response time, error rates, latency, throughput, user satisfaction scores, and other relevant business-specific metrics. *(ii)* Technical Metrics with Sysdig: Sysdig or its equivalent tool for system monitoring is employed to collect technical metrics related to operating system, network performance, containerized environments, and other infrastructure aspects. These metrics include CPU utilization, memory usage, network latency, disk I/O, and container performance, among others. *(iii)* Database and Infrastructure Metrics with New Relic: New Relic or its equivalent tool for performance monitoring is used to collect metrics specific to databases and infrastructure components. These metrics can include database response time, query throughput, database errors, server load, and infrastructure resource utilization [17].

C. Data Processing and Analysis

Once the metrics are collected, they need to be processed and analyzed to derive valuable insights. Each tool such as Kibana, Sysdig, and New Relic offers built-in capabilities for data processing and analysis. *(i)* Kibana: With Kibana, data can be transformed, aggregated, and statistically analyzed to identify patterns, trends, and anomalies in business metrics. This helps in understanding the overall service performance, user behavior, and adherence to SLAs. *(ii)* Sysdig: Sysdig provides powerful analytical features to process technical metrics, enabling the identification of performance bottlenecks, resource utilization issues, and potential infrastructure-related problems that may impact SLAs and SLIs. *(iii)* New Relic: Similarly, New Relic's data processing and analysis capabilities assist in evaluating database performance, identifying slow queries, monitoring resource usage, and ensuring infrastructure stability and all critical factors affecting SLAs and SLOs.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *05 July 2023* ***Revised:*** *25 August 2023* ***Accepted:*** *15 September 2023*

D. Service Level Agreements (SLA) Monitoring

The proposed architecture enables continuous monitoring of SLAs using the insights derived from the metrics [2]. Thresholds can be defined for different metrics across Kibana, Sysdig, and New Relic dashboards. When a metric breaches its threshold, notifications or alerts can be triggered to notify relevant stakeholders. This proactive monitoring approach allows for immediate actions to be taken to rectify any SLA deviations.

E. Service Level Indicators (SLI) Definition and Tracking

SLIs quantify specific aspects of service performance and serve as key indicators of system health. The proposed architecture facilitates the definition and tracking of SLIs using the metrics collected by the respective tools. *(i)* Kibana: Business metrics collected via Kibana contribute to SLI definitions such as response time, error rates, and user satisfaction scores. These SLIs can be visualized and tracked on Kibana dashboards, providing real-time insights into service performance. *(ii)* Sysdig: Technical metrics from Sysdig contribute to SLIs related to system and network performance. SLIs such as CPU utilization, memory usage, and network latency can be defined and tracked through Sysdig dashboards. *(iii)* New Relic: Database and infrastructure metrics collected by New Relic can be utilized to define SLIs related to database response time, query throughput, and server availability. The proposed architecture utilizes Kibana, Sysdig, and New Relic dashboards to collect and analyze business, technical, and database/infrastructure metrics, enabling continuous monitoring of SLAs and tracking of SLIs. Enterprises can proactively address performance issues, identify bottlenecks, and ensure adherence to SLOs, streamline software development processes by leveraging these metrics-driven dashboards.

IV. RESULTS AND DISCUSSION

The metrics-driven development approach is implemented using the proposed architecture and tested on two industry applications. The performance of two applications are measured and compared over a three-month period using New Relic and Sysdig dashboards to monitor technical and infrastructure metrics. Figures 2–5 present details of response time metrics, query execution time, CPU utilization, and error rate while testing the proposed MDD based architecture on Industry Application 1.

The SLA for both applications included a response time of 700 ms, high availability of 98%, and a 3-hour resolution time for any issues that violated the contract. Application-1 consistently met the SLA throughout the monitoring period, with a maximum response time of 21 ms and an error rate of 4%. The database query execution time remained within the SLA, peaking at 21 queries per minute. The CPU utilization of the application was also remarkably low, with a maximum of 0.6%. Additionally, Application-1 maintained a high availability rate of 99.7%, surpassing the defined SLA of 98%.

Figures 6 – 9 show details of response time metrics, query execution time, CPU utilization and error rate while testing the proposed MDD based architecture on Industry Application 2. Application-2 successfully met the defined SLA, with a maximum response time of 250 ms and an error rate of 24%. Although the database query execution time is higher, reaching up to 1.8 K queries per minute, it has not violated the SLA.

The CPU utilization of the application is also relatively low, with a maximum of 1%. The high availability of Application-2 was maintained at 98.9%, which is almost equal to the SLA of 98%, but still within an acceptable range. Overall, both applications managed to meet the defined SLA, albeit with different levels of performance. Application-1 demonstrated better technical and infrastructure metrics, meeting the SLA more efficiently and with greater availability. Overall, both applications managed to meet the defined SLA, albeit with different levels of performance. Application-1 demonstrated better technical and infrastructure metrics, meeting the SLA more efficiently and with greater availability. On the other hand, Application-2 also succeeded in meeting the SLA despite higher resource utilization and error rates. Findings of this study highlight the effectiveness of metrics-driven deployment approach in monitoring and optimizing application performance, leading to improved SLA compliance and customer satisfaction. By leveraging the insights provided by New Relic and Sysdig dashboards, enterprises can proactively identify performance bottlenecks, resource utilization issues, and potential areas for improvement. These dashboards enable teams to take timely actions, optimize system performance, and ensure adherence to SLAs.

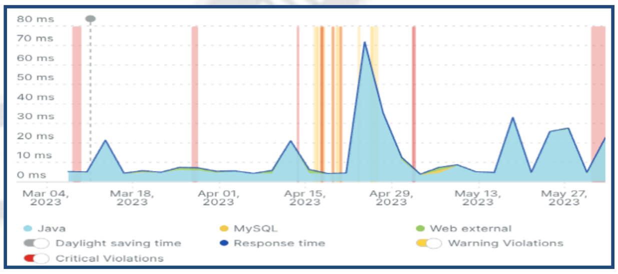

Figure 2. Response Time Metrics of Industry Application-1 using the Proposed MDD Architecture

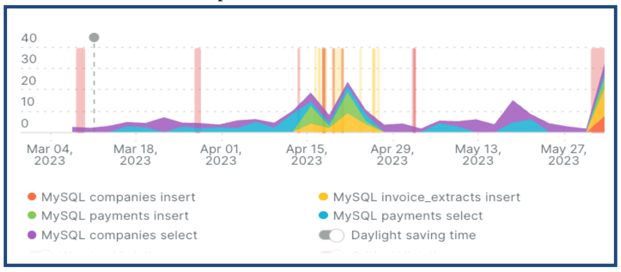

Figure 3. Query Execution Time in the Database for Industry Application-1 using the Proposed MDD Architecture

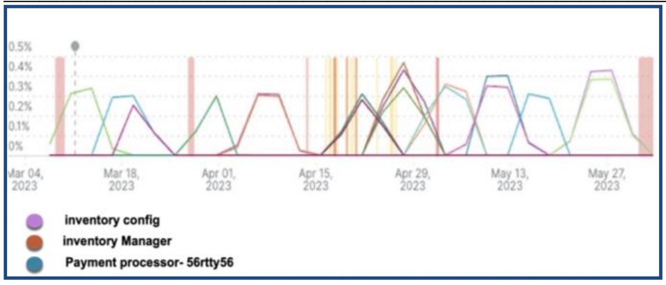

Figure 4. CPU Utilization of the Industry Application-1 using the Proposed MDD Architecture

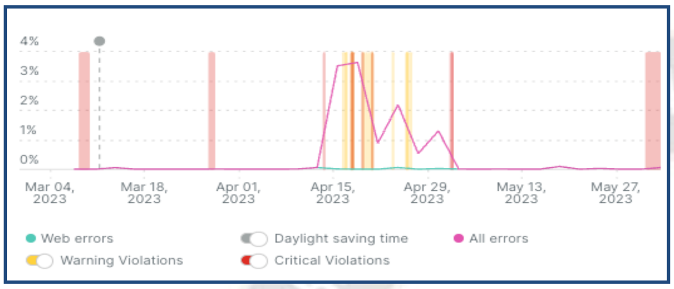

Figure 5. Error Rate of the Industry Application-1 using the Proposed MDD Architecture

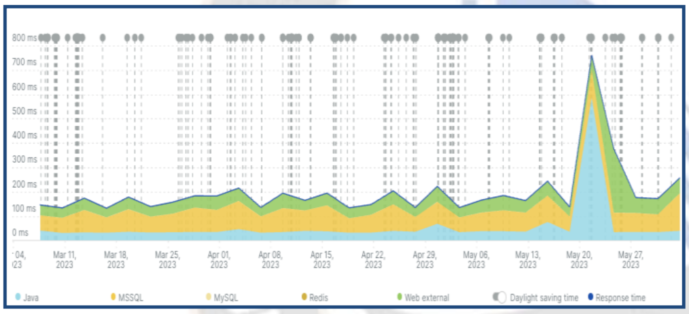

Figure 6. Response Time Metrics for Industry Application-2 using the Proposed MDD Architecture

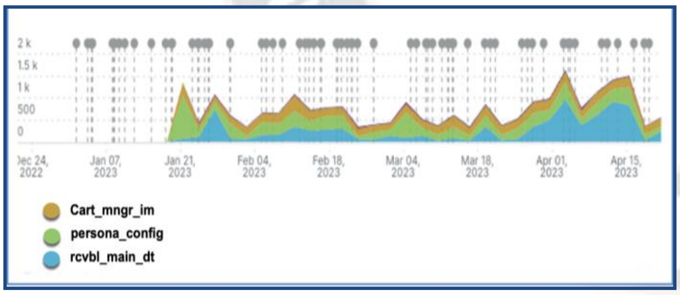

Figure 7. Query Execution Time in the Database for Industry Application-2 using the Proposed MDD Architecture

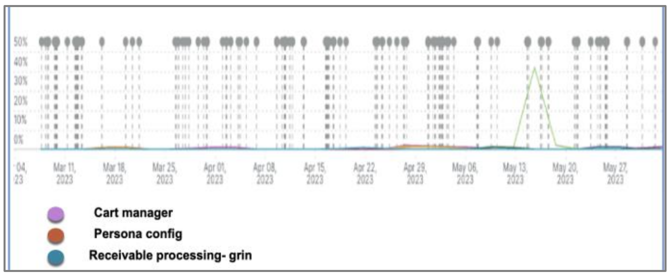

Figure 8. CPU utilization of the Industry Application-2 using the Proposed MDD Architecture

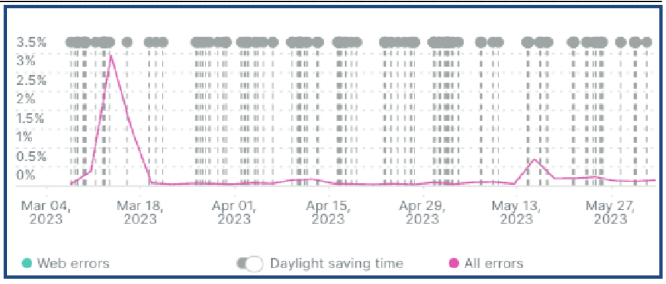

Figure 9. Error Rate of the Industry Application-2 using the Proposed MDD Architecture

It is important to note that while Application-1 demonstrated superior performance metrics, Application-2's ability to meet SLA highlights the resilience and adaptability of metrics-driven approach. Even with higher resource utilization and error rates, Application-2 has successfully maintained satisfactory performance levels. Further research could focus on fine-tuning the metrics collection process, exploring additional tools and techniques, and investigating the correlation between specific metrics and SLA compliance. Enterprises can streamline software development processes, improve service quality, and achieve higher levels of SLA, SLI, and SLO compliance by continuously refining and optimizing the metrics-driven development approach.

V. CONCLUSIONS AND FUTURE WORK

This paper indicates the importance of metrics driven development while implementing enterprise applications using the Cloud. A precise summary of the work done based on metrics driven development is also presented in this work. It is observed that the implementation of the proposed architecture leveraging tools such as Kibana, Sysdig, and New Relic or its equivalent dashboards for collecting and analyzing metrics in Cloud-based software development within enterprises has yielded significant results in managing and improving Service Level Agreements (SLAs), Service Level Indicators (SLIs) and Service Level Objectives (SLOs). Utilization of the above-mentioned Cloud-based tools has allowed for comprehensive tracking and analysis of metrics related to SLA, SLI, and SLO management. These dashboards have provided real-time insights into various aspects of the Cloud-based software development process i.e. full stack development, enabling teams to proactively address performance issues and ensure adherence to service-level commitments. It is found that enterprises have experienced improved SLA compliance by continuously monitoring metrics through integrated dashboards. The real-time alerts and notifications provided by Kibana have enabled teams to identify and address performance issues promptly, ensuring that service performance remains within the defined SLA thresholds. The ability to track SLIs across multiple dimensions, such as response time, error rates, and user satisfaction, has facilitated a proactive approach to maintain high-quality service delivery. The visualizations and analytics provided by Kibana, Sysdig, and New Relic dashboards or its equivalent have enhanced SLI tracking across various technical and business aspects. Kibana dashboards or their equivalent provide insights into user behavior, service performance, and overall system health, empowering stakeholders to make informed decisions. Sysdig dashboards or its equivalent offer detailed technical metrics, allowing teams to identify resource utilization issues, network bottlenecks, and other performance-related concerns. New Relic dashboards or its equivalent provide visibility into database performance, ensuring optimized query execution and minimizing potential SLI deviations related to the database and infrastructure.

The proposed architecture has significantly improved SLO management within enterprises. Teams have gained valuable insights into service performance trends and patterns through historical analysis of metrics collected by Kibana, Sysdig, and New Relic. This data-driven approach has facilitated the establishment of realistic and achievable SLOs. By setting appropriate thresholds for SLIs and continuously monitoring their compliance, teams can proactively address any deviations, enabling better alignment with customer expectations and service-level commitments. The adoption of the proposed architecture utilizing Kibana, Sysdig, and New Relic dashboards for metrics-driven development in enterprises has demonstrated significant benefits in terms of SLA compliance, SLI tracking, and SLO management. Continuous monitoring, proactive alerting, and data-driven decision-making facilitated by these dashboards have resulted in improved service performance, enhanced customer satisfaction, and streamlined software development processes.

Future research could focus on further optimization of metrics collection and analysis process and exploring additional tools and techniques to augment the metrics-driven development approach. The proposed architecture by leveraging Kibana, Sysdig, and New Relic dashboards has shown promising results in improving SLA, SLI, and SLO management in software development within enterprises. There are several avenues for future work and research to further enhance the metrics-driven development approach. The potential areas of focus can include the following:

Refinement of Metrics Collection and Analysis: Future research could explore ways to optimize the collection and analysis of metrics from various sources, including Kibana, Sysdig, and New Relic dashboards. This could involve developing more sophisticated data aggregation techniques, implementing machine learning algorithms for anomaly detection and predictive analytics, and leveraging advanced statistical models to uncover deeper insights from the collected metrics.

Integration of Additional Tools and Techniques: The proposed architecture can be expanded by integrating additional tools and techniques that complement the metrics-driven development approach. For example, incorporating log monitoring and analysis tools, performance testing frameworks, and Application Performance Management (APM) solutions can provide a more comprehensive view of system performance and enable more proactive issue detection and resolution.

Automation and Remediation: Future work could focus on automating the remediation process based on the insights derived from the metrics dashboards. By leveraging machine learning and automation techniques, it would be possible to develop intelligent systems that can automatically identify performance bottlenecks, suggest remedial actions, and even implement them in real-time, thereby reducing the manual effort required for issue resolution and improving system reliability.

REFERENCES

[1] H. Tu, W. Sun, and Y. Zhang, The research on software metrics and software complexity metrics, in Proceedings of International Forum on Computer Science-Technology and Applications, vol. 1, pp. 131–136, 2009.

[2] B. Gomathi, N. K. Karthikeyan, Saravana Balaji, Epsilon-fuzzy dominance sort-based composite discrete artificial bee colony optimization for multi-objective cloud task scheduling problem, International Journal of Business Intelligence and Data Mining, vol. 13, pp.247–266, 2018.

[3] Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer and D. Epema, Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing, in IEEE Transactions on Parallel and Distributed Systems, vol. 22, Issue no. 6, pp. 931-945, 2011.

[4] B. Gomathi, B. Saravana Balaji , V. Krishna Kumar, Mohamed Abouhawwash, Sultan Aljahdali, Mehedi Masud and Nina Kuchuk, Multi-Objective Optimization of Energy Aware Virtual Machine Placement in Cloud Data Center, Intelligent Automation and Soft Computing, vol.33, No.3, pp.1771-1785, 2022.

[5] Murali Dhar, M.S., Manimegalai, R., A Policy-Oriented secured service for the e-commerce Applications in Cloud, Personal and Ubiquitous Computing, Vol.22, pp.911–919, 2018.

[6] He, Q., Han, J., Yang, Y., Grundy, J. & Jin, H, QoS-driven service selection for multi-tenant SaaS, IEEE International Conference on Cloud Computing, United States, pp. 566-573, 2012.

[7] Bautista Villalpando, L.E., April, A, Abran, A, Performance analysis model for big data applications in cloud computing, Journal of Cloud Computing, vol. 3, no. 19, 2014.

[8] Ardagna, D., Casale, G., Ciavotta, M., Pérez, J.F. and Wang, W., Quality-of-service in cloud computing: modeling techniques and their applications, Journal of Internet Services and Applications, vol. 5, no.1, 2014.

[9] Ali, Md Mohsin, Shamsul Huda, Jemal H. Abawajy, Sultan Alyahya, Hmood Al-Dossari and John Yearwood, A parallel framework for software defect detection and metric selection on cloud computing, Cluster Computing, vol. 20, pp. 2267 - 2281, 2017.

[10] [Zhou, L., Zhang, L., and Ren, L, Modelling and simulation of logistics service selection in cloud manufacturing, 51st CIRP Conference on Manufacturing Systems, Procedia CIRP, vol. 72, pp. 916–921, 2018.

[11] Araujo, J., Maciel, P., Andrade, E., Decision making in cloud environments: an approach based on multiple-criteria decision analysis and stochastic models, Journal of Cloud Computing: Advances, Systems and Applications, vol. 7, No.1, pp. 1-19, 2018.

[12] Chawla, P., Chana, I. and Rana, A., Framework for cloud based software test data generation service, Software: Practice and Experience, vol. 49, no.8, pp.1307-1328, 2019.

[13] Chen, Xing, Junxin Lin, Bing Lin, Tao Xiang, Y. Zhang and Gang Huang. Self- learning and self -adaptive resource allocation for cloud based software services, Concurrency and Computation: Practice and Experience, vol. 31, 2019.

[14] Xing Chen, Haijiang Wang, Yun Ma, Xianghan Zheng, Longkun Guo, Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model, Future Generation Computer Systems, Vol. 105, pp. 287-296, 2020.

[15] Yaghoubi, M and Maroosi, A., Simulation and modeling of an improved multi-verse optimization algorithm for QoS-aware web service composition with service level agreements in the cloud environments, Simulation Modelling Practice and Theory, Vol. 103, 2020.

[16] K. S. Arvind, R. Manimegalai, and K. S. Suganya, Privacy Preserving Public Auditing for Cloud Storage Using Elliptic Curve Digital Signature, Journal of Computational and Theoretical Nanoscience, Vol.15, pp.1568–1572, 2018.

[17] B .Gomathi, S. T. Suganthi, Karthikeyan Krishnasamy, J. Bhuvana, Monarch Butterfly Optimization for Reliable Scheduling in Cloud, Computers, Materials and Continua, Vol.69, No.3, pp.3693-3710, 2021.