iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Service-Oriented Applications Based On an Enterprise Service Bus

Prof. Swati Anantwar[1], Miss Yamini Mishra[2]
Information Technology,S.G.B.A.U.
Amravati, Maharashtra, India

Abstract--The concept of enterprise service bus has been originally invented to reduce complexity of enterprise application integration within an enterprise. Enterprise service bus (ESB) is a software architecture model used for designing and implementing communication between mutually interacting software applications in a service-oriented architecture (SOA). As software architectural model for distributed computing it is a specialty variant of the more general client server model and promotes agility and flexibility with regards to communication between applications. Its primary use is in enterprise application integration (EAI) of heterogeneous and complex landscape. In this paper, we develop a performance model for analyzing and predicting the runtime performance of service applications composed on a COTS ESB platform. Our approach utilizes benchmarking techniques to measure primitive performance overheads of service routing activities in the ESB. Queuing network, which facilitates the performance prediction of service oriented applications.

I. INTRODUCTION

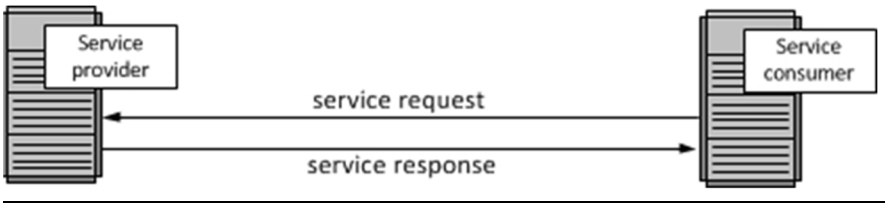
A service–oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed. A service–oriented architecture are not a new thing. The first service–oriented architecture for many people in the past was with the use DCOM or Object Request Broker (ORBs) based on the CORBA specification, For more on DCOM and CORBA, see prior service–oriented architectures

A. Services

If a service–oriented architecture is to be effective, we need a clear understanding of the term service .A service is a function that is well-defined, self-contained, and does not depend on the context or state of other service.

B. Connection

The technology of Web Services is the most likely connection technology of service–oriented architecture. The following figure shows the basic service–oriented architecture. It shows a consumer at right sending a service request message to a service provider at the left. The service provider return a response message to the service consumer . The requests and a subsequent response connections are defined in some way that is understandable to both the service consumer and provider .A service provider can be a service consumer



Service-Oriented Architecture (SOA) has emerged as the leading IT agenda for infrastructure reformation, to optimize service delivery and ensure efficient business process management. Part of the paradigm shift of SOA are fundamental changes in the way IT infrastructure is designed—moving away from an application infrastructure to a converged service infrastructure. Service-Oriented Architecture enables discrete functions contained in enterprise applications to be organized as layers of interoperable,

standards-based shared "services" that can be combined and reused in composite applications and processes. In addition, this architectural approach also allows the incorporation of services offered by external service providers into the enterprise IT architecture. As a result, enterprises are able to unlock key business information in disparate silos, in a cost-effective manner. By organizing enterprise IT around services instead of around applications, SOA helps companies achieve faster time-to-service and respond more flexibly to fast-paced changes in business requirements.

In recent years, many enterprises have evolved from exploring pilot projects using ad-hoc adoption of SOA and expanded to a defined repeatable approach for optimized enterprise-wide SOA deployments. All layers of an IT SOA architecture have become service-enabled and comprise of presentation services, business processes, business services, data services, and shared services. The performance of an SOA-based application is often critically important to a business. If the SOA doesn't perform well, then the applications that leverage the composition capability of SOA have limited benefit. In SOAs, service applications can be considered as a composition of individual services. Each individual service exposed is implemented by component hosted in some node in the distributed environment. It can be seen that the SOA performance problem falls into two broad categories: ensuring sufficient performance of individual services as well as of the composite services . Individual services provide service interfaces that encapsulate existing systems, ensuring their performance necessitates managing the performance of the components, applications, and systems that lie beneath the services abstraction. Well-established capacity planning methods, techniques and tools can be leveraged to manage the performance of individual services, such as logging- based instrumentation or simulating the load on service interfaces by load testing in a similar way in simulating traditional web application performance . Dealing with the performance of services integrated by ESBs falls into the second category of SOA performance issues, and it is far more complex than atomic services. Services registered to an ESB can be uniquely identified by their endpoints. An ESB composes services in a loosely coupled way by routing and transforming messages among services with different protocols. Some services expect high volumes of traffic from a specific consuming application, while others expect high reuse, which implies that a service is used by different consumers. This makes the workload characterization of composite services complicate Moreover, although ESBs mask the complexities of composing applications and services, they also introduce performance overheads incurred by message brokering and transformation between heterogeneous services and applications. Therefore, analyzing and predicting the performance of ESB-based applications requires both the understanding of atomic services and the performance characteristics of ESBs. One challenging issue is that these performance characteristics of ESBs are hard to measure by instrumentation or monitoring due to the nature of highly distributed architecture in SOA. This necessitates a systematic approach that facilitates estimating performance characteristics of ESBs and analyzing the performance relationship between the ESB and the services it composes

In this paper we present a combined performance modeling and benchmark approach to address the above problems. A queueing network performance model is devised to represent the performance critical ESB infrastructure components and the services running on the ESB. The performance characteristics of these ESB components are estimated by benchmarking results.

II. UNDERSTANDING THE ESB ARCHITECTURE

In order to model the performance of an ESB, we first need to understand the architecture of the ESB and identify key components of the ESB that have critical performance impact. The capabilities of an ESB are implemented by middleware technologies such as web services, message broking, security management and so on. Despite the diversities in ESB implementation, there are common components in the architecture of an ESB. An abstract architecture of an ESB is illustrated in Figure 1. The core functionality of an ESB is supporting disparate applications and services to communicate using different protocols as shown in the protocol stack of Figure 1. The ESB message brokering provides content- based routing for dispatching a request message from one service to another service or application registered on the ESB. A service or application is connected to the service bus through an ESB 'gateway'.

A gateway (also called a proxy in some ESB implementations) can be configured with a routing table to route a request to the appropriate business service. The routing table entry defines a set of actions, including routing to a service, call out a service, composing messages with input parameters for invoking services, transforming output parameters to response messages, and validating the message content. At runtime the routing table guides the message flow. Each action is executed by checking the predefined expression with variables, operators and variable values associated with an action. The gateway also marshals a request from the message bus to the service.

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

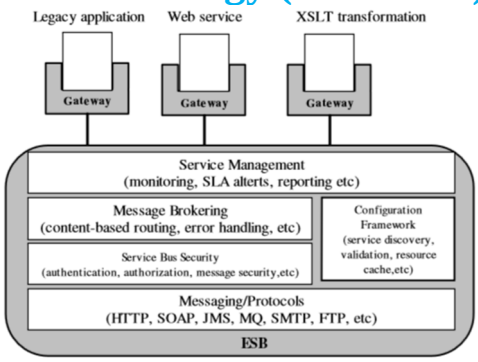


Figure 1 High level ESB architecture

Figure 2 shows an example using content-based routing to connect an application to two services depending on some condition configured in the routing table encapsulated in RouteNode1. This routing configuration is attached to the ESB gateway LoanGateway1. An ESB implementation normally provides comprehensive services and utilities such as service monitoring, reporting and message security. Details of these services can be found from individual ESB implementations

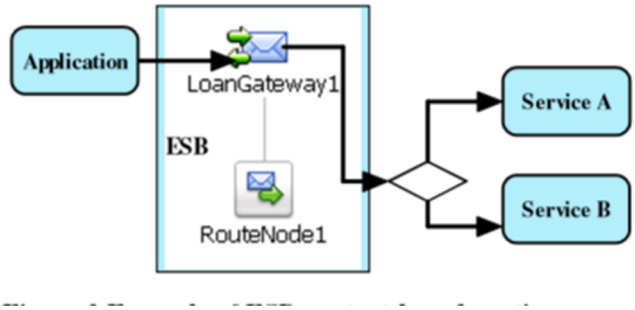


Figure 2 Example of ESB content-based routing

Based on the above understanding, we can see that the end-to-end responsiveness of a service request depends on the performance characteristics of the ESB routing and the destination services. As we discussed in section 1, the performance characteristics of individual services can beobtained as they are atomic services. If these services are also composed then the activity of obtaining their performance characteristics becomes recursive. The performance characteristics of an ESB are mainly determined by its capacity for content-based routing and message transformation. Extra ESB infrastructure functions such as message flow monitoring and message security management can also incur additional overhead. In this paper, we focus on the performance overhead of ESB routing and transformation. This can simplify the performance analysis approach and establish a base-line model, which can be further extended to incorporate more complex ESB features such as monitoring and security management. In the rest of this paper, we propose a modeling-based approach that maps the ESB architecture to an analytical model that is solved to analyze and predict the performance of the ESB-based loan processing application.

A real world example of ESB-based loan application In this paper, we use an ESB integrated application modeled from a real world loan processing application as an example to illustrate our modeling approach. We consider this loan application because its scenarios are representative. They illustrate the basic requirements for SOA design and integration technologies such as Web Services and ESB. The architecture design is discussed in detail in (see chapter 9). Different versions of the loan application were implemented on several ESB platforms including Mule and BEA Aqua Logic Service Bus (ASB) . Therefore, analyzing the performance of this loan application can help us to identify research issues involved in performance modeling of ESB-based applications, and show insight in to their solutions. In this paper we use the COTS BEA ASB implementation of the loan application. The basic business logic is that a primary mortgage company uses BEA ASB to route loan applications to appropriate business services. It includes three scenarios shown in Figure 3:

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

The messages for every loan application through the BEA ASB are validated (loan Validation). If the application is: • Incomplete, it is written to an error directory and an error message is returned to the client. • Complete, it is routed to an appropriate business service for review. • Approved, the service returns a message indicating whether the loan is accepted or rejected. 2. An application containing a request for a rate less than 5% requires management approval and is routed to an appropriate business service (manager Loan-) Review Service) for processing. All other loan applications are routed to another business service (Normal Loan) for processing. 3. Loan applications with a principal request of greater than US$ 25 million are candidates for sale to a secondary loan company. The applicant's credit rating information is retrieved by making a callout to a Web Service (Credit Rating Service). The credit rating information is added to the loan application, then the application is forwarded to the secondary mortgage company Web Service to be processed (Loan Sale Processor). Loan applications with a principal request equal to or less than US$ 25 million are routed to the Normal Loan business service for processing.

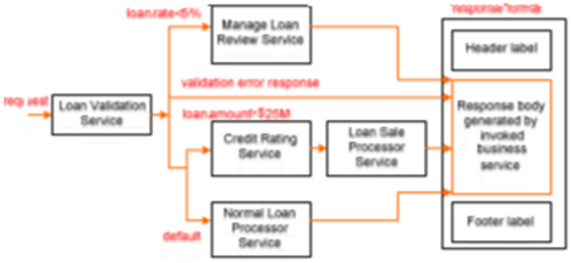


Figure 3 Scenarios of load application

In the ASB's implementation of this application, each scenario has a gateway in the ESB (called a proxy service in BEA ASB) for routing the loan request according to different conditions. The interactions between the loan application, business services and proxy services are shown in the sequence diagram of Figure 4. For performance testing, the loan application is deployed on three workstations: one for the client loan application; one for the ASB server and one for hosting all the business services. These three workstations are identical with Dual Intel Xeon 3.00GHz CPUs and with 3G RAM, running Windows XP.

III. PERFORMANCE MODELING APPROACH

The overall performance modeling approach follows the general capacity planning process . This includes:
Mapping the application components or services at the software architecture level to analytical model elements
Characterizing the workload pattern for the individual components as input for the performance model
Calibrating the performance model by populating the parameter values • Validating the performance model and predicting the performance
We describe our modeling approach by illustrating each step in the context of the loan application using the ESB.

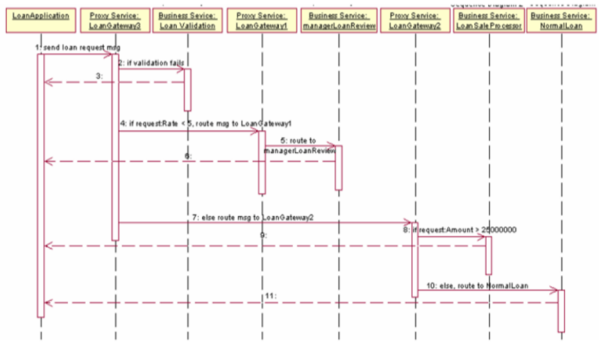


Figure 4 Interactions of the load application

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

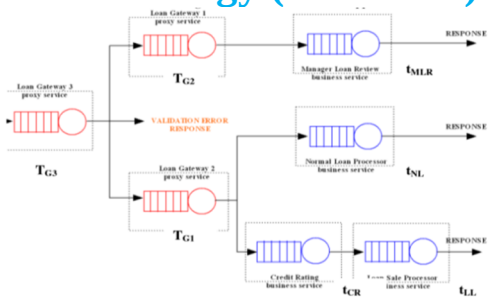


Figure 5 Queuing network model of the loan application

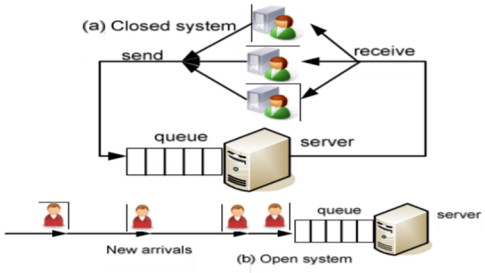
This approach is general and hence can be applied to other ESB-based applications.

A. The Performance Model

In the interaction diagram of the loan application (see Figure 4), there are three proxy services for each scenario, and five business services, namely loan Validation, manage Loan Review, normal Loan, credit Rating and loan Sale Process. Note that the validation of messages is simple and executed by a proxy service without any business service involved in this scenario. These proxy and business services form a queuing network as illustrated in Figure 5. In this model each service is mapped to a load- independent resource (represented by a queue notation in Figure 5) that serves arriving requests in the analytical queuing network model. The assumption that each service is load-independent simplifies the modeling complexity as well as reduces the effort of collecting parameter values to represent the load-dependent behavior of a service. This assumption is later validated by the accuracy of the modeling results in this section. The connections between proxy services and business services can be of different types of networks, such as dedicated network channels, Ethernet, or WAN. There can be firewalls and also load balancing proxies installed along the requests invocation paths of business services. The service delay introduced by network protocols and software can be represented by a system level model (see Chapter 8 in [10]) and combined into the performance model in Figure 5 by applying component level modeling methods (see Chapter 9 in [10]). Details of modeling network protocols and software are out of the scope of this paper.

B. Workload Characterization

The overall throughput of an ESB depends on the workload imposed to it. The ASB platform is shared by the three scenarios in this load application, and their usage patterns of the ESB are different from each other as discussed in section 3. Overall four classes of workload are identified based on the message flow branches, namely loan validation (VAL), manager loan review (MLR), large loan (LL) and normal loan (NL). In the queuing network model, the mode that workload arrives at the system determines if the system is modeled as a closed or an open model. For a closed model, new job/request arrivals are only triggered by job completions (followed by think time), as in Figure 6 (a). By contrast in an open model, new jobs/requests arrive independently of job completions, as in Figure 6 (b). Schroeder et al demonstrated that closed and open system models yield significantly different results, even when both models are run with the same workload and service demands [1]. This means determining the type of the model as closed or open is critical to the accuracy of the performance modelling results.



By the nature of the loan application, the requests arrive at the ASB independently of job completions, and we can consider modeling the system as an open model. In order to model and validate the system in an open model, we need to generate workload in the open model accordingly. However, most available and popular web- based workload generators assume a closed system model when generating workload, including TPC-W, RUB is, Microsoft Web Application Stress Tool, WES tone, SPECJ2EE and many others [1]. For these tools, a fixed number of concurrent clients are specified and requests are generated from each client. A new request starts after the previous invocation completes with a configurable time interval between two consecutive requests. This issue hence raises the engineering problem of how to model an open system with only workload generators available for closed systems. Of course, one can implement a workload generator that generates open workload with a certain distribution of the job arrival rate. However, this requires extra software development effort and it is non-trivial to have a fully fledged workload generator with functions for collecting metrics. Our solution to this problem applies closed and open models in two stages. At the stage of performance modeling and validating, we use the available closed system workload generator and model the system as a closed model under a workload represented as the number of requests. Using this closed model, we can validate the assumption made on the type of resources and the accuracy of input parameter values estimated. Then at the stage of predicting performance, we replace the model as an open one with the workload presented as arrival rates. In this open model, the type of resources and the parameter values remain the same as they are in the closed model.

IV. DISCUSSIONS ON POTENTIAL EXTENSIONS

So far we have described our performance modeling approach for ESBs, which focuses on performance characteristics of routing messages to different services. A baseline performance model is derived from this work. ESBs have more advanced features to facilitate the integration of services, such as message security management. We envision our approach can be extended to model the performance of ESB-based systems with security management enabled. Figure 9 shows an abstract security model in an ESB. Similar to Web Service security management, security of ESB messages can be controlled at different levels: transport level message encryption, and message level authentication and authorization. The key issue in performance modeling is to identify the critical components involved in enabling this feature. We can see from the abstract security model in Figure 9 that proxy services are responsible for invoking authentication or authorization providers and CAs to obtain the necessary security tokens. One solution is to calibrate the service demand of the proxy services in the baseline model. The extra delay occurred by security management are augmented to the service demand of the proxy services. This value can be obtained by benchmark measurement. Alternatively, the security management can be abstracted and modeled as another resource and combined with the baseline model. Accordingly, the delay of the security management serves as the service demand of this newly introduced resource. It remains our future work to evaluate and compare the accuracy of these two modeling solutions.

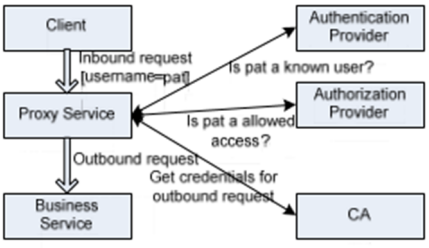


Figure 9 Abstract Security Model in ESB

V. RELATED WORK

Performance of SOAs is mainly studied as Web Service performance, as a web service is one of the key enabling technologies of SOA. Menasce discussed the QoS issues in Web Service and presented how a systematic capacity planning process can be applied to Web Service . applied layered queuing networks to a web service-based health software architecture. Other work focuses on the performance evaluation and analysis of the SOAP protocol in Web Services . To the best of our knowledge, there hasn't been work

done in modeling the performance of ESB-based applications.

VI. CONCLUSION

The performance model developed forms a baseline model and it remains our future work to extend this model to analyze other features of an ESB such as the security management. This case study focuses on the performance modeling method instead of doing performance evaluation of the ESB with stress testing. Therefore, the workload imposed is synthetic. For a real world application, the workload characterization approach described in section 4.2 can be applied. Obtaining the parameter values to solve the model requires the measurement of the delay incurred by individual ESB infrastructure gateway/proxy service. This can be achieved using performance testing tools and linear regression. One on of this paper is that it combines both closed system and contribution open system analysis method at the model validation and prediction stage. Simplified assumptions have been made to reduce the engineering effort for performance measurements. These assumptions are validated by empirical results, otherwise inappropriate assumptions need to be revised and revalidated. The case study using a loan application validates our approach and shows that it is practical for COTS ESBs. The experience gained from this work provides us with insightful understanding of performance analysis issues involved in ESB enable applications. In the future, this approach will be further validated to different ESB implementations and applications integrated through different communication protocols by ESBs **.**

REFERENCES

[1] Almeida, V. A. and Menascé, D. A. 2002. Capacity Planning: An Essential Tool for Managing Web Services. IT Professional 4, 4 (Jul. 2002), 33-38. DOI= http://dx.doi.org/10.1109/MITP.2002.1046642

[2] Bianca Schroeder, Adam Wierman and Mor Harchol- Balter. Open vs closed: a cautionary tale, Networked System Design and Implementation NSDI, 2006.'06

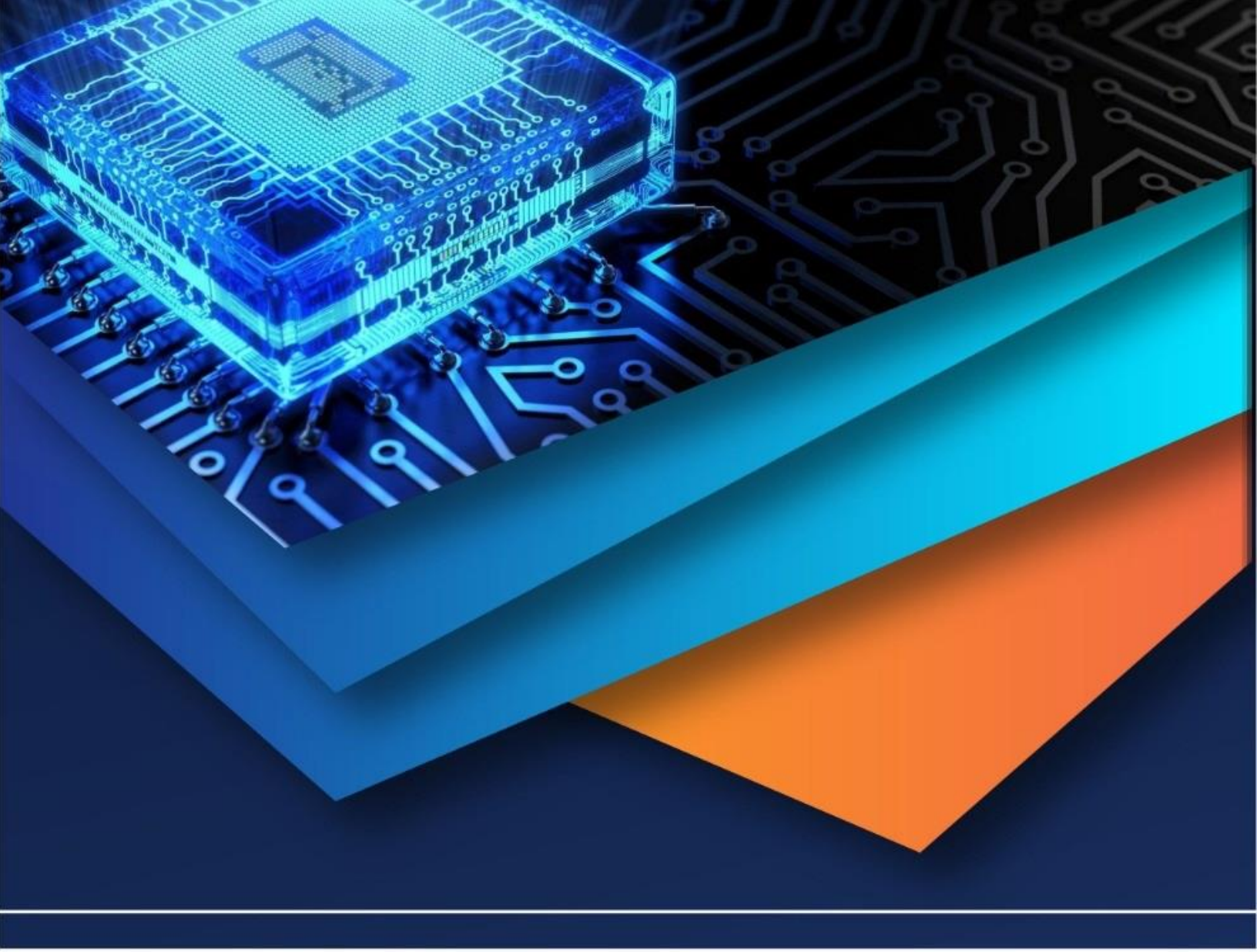
[3] Carolyn McGregor, Josef Schiefer, "A Framework for Analyzing and Measuring Business Performance with
Web Services," cec, p. 405, 2003 IEEE International Conference on E-Commerce Technology (CEC'03), 2003.

[4] Chen, S., Yan, B., Zic, J., Liu, R., and Ng, A. 2006. Evaluation and Modeling of Web Services Performance. In Proceedings of the IEEE international Conference on Web Services (Icws'06) - Volume 00 (September 18 - 22, 2006). ICWS. IEEE Computer Society, Washington, DC, 437-444. DOI= http://dx.doi.org/10.1109/ICWS.2006.59

[5] Christina Catley, Dorina C. Petriu, Monique Frize, Software Performance Engineering of a Web service- based Clinical Decision Support infrastructure. In Proceedings of the 4th international Workshop on Software and Performance (Redwood Shores, California, January 14 - 16, 2004). WOSP '04. ACM Press, New York, NY, 130-138. DOI= http://doi.acm.org/10.1145/974044.974066

[6] Davis, D. and Parashar, M. Latency Performance of SOAP Implementati
ons. In Proceedings of the IEEE Cluster Computing and the GRID 2002 (CCGRID'02), Berlin, Germany, IEEE, 2002

[7] Gregor Hohpe, Bobby Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Professional (October 10, 2003), ISBN-13: 978-0321200686



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY