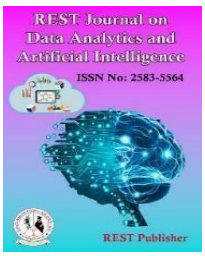
**REST Journal on Data Analytics and Artificial Intelligence**
**Vol: 4(2), June 2025**
**REST Publisher; ISSN: 2583-5564**
**Website: http://restpublisher.com/journals/jdaai/**
**DOI: https://doi.org/10.46632/jdaai/4/2/11**




# Graphical interactive is debugging for distributed systems using WASPAS method

***Vimala Saravanan, M. Ramachandran, Nathiya Murali, Poonkodi Sathiyamoorthy**
*REST Labs, Kaveripattinam, Krishnagiri, Tamil Nadu, India.*
*Corresponding Author Email: vimalarsri@gmail.com

***Abstract:*** *Graphical Interactive Debugging (GID) for distributed systems is an advanced debugging methodology that addresses the intricacies of diagnosing and rectifying errors in distributed software applications. This approach combines visual representations, real-time monitoring, and interactive controls to provide developers and system administrators with a comprehensive view of the distributed system's behavior and internals. By offering an intuitive and informative interface, GID facilitates the identification of anomalies, performance bottlenecks, and communication failures that commonly arise in distributed environments. Distributed systems are complex software architectures that consist of multiple interconnected components or nodes that collaborate to achieve a common goal Debugging such systems is challenging due to the inherent complexities introduced by network communication, concurrency, and the potential for failures at various levels. Graphical Interactive Debugging (GID) is an approach that leverages visual representations and interactive tools to aid in the identification and resolution of issues within distributed systems. Distributed systems are complex software architectures that consist of multiple interconnected components or nodes that collaborate to achieve a common goal. Debugging such systems is challenging due to the inherent complexities introduced by network communication, concurrency, and the potential for failures at various levels. Graphical Interactive Debugging (GID) is an approach that leverages visual representations and interactive tools to aid in the identification and resolution of issues within distributed systems. Applying the "Weighted Aggregated Sum Product Assessment (WASPAS)" method, those making decisions are able to evaluate possibilities based on a wide range of parameters. It involves providing criteria weights, determining scores for every potential and summing the results to determine the most suitable option. Result: From the result "Federated Array of Bricks (FAB)" is in the 1st rank, and bullet is in the lowest rank.*

**Keywords:** *Distributed systems, debugger, event debugging, behavioral abstraction, communication and synchronization errors*

## 1. INTRODUCTION

The complexity of distributed systems surpasses that of single-node applications. For example, a single logical operation can entail the transmission of numerous messages and engagement with multiple nodes. Because events' timing and locations can vary between actions, the diversity of distributed behavior is heightened. Consequently, identifying bugs in distributed systems can be demanding, given their potential impact on multiple nodes or specific behavior patterns. To address this, an automated system like Pip compares the behavior of a distributed system with the assumptions made by programmers. Pip assesses system behaviors, classifying them as valid or invalid, grouping them into sets for logical analysis, and presenting the overall behavior in various formats. Given that imperfections are present in any complex system, the process of identifying and rectifying errors is often considered more of an artistic endeavor than a strictly scientific one. Despite this, no precise or elegant approach has yet been devised for this undertaking. The situation becomes more challenging when dealing with distributed environments, as this introduces intricacies into the debugging process and leads to the emergence of novel types of glitches. In the realm of distributed scenarios, scholars have recently formulated effective strategies for debugging, one such example being event-based debugging. This methodology is particularly beneficial for pinpointing issues related to synchronization or communication. Moreover, the design of a debugger for networked real-time multimedia systems should encompass certain elements. This type of debugging proves valuable in identifying anomalies in synchronization or communication. Furthermore, the debugger should incorporate hooks that allow users to employ traditional sequential debugging methods, which focus on the internal logic or behavior of processes. For the sequential debugging aspect, a standard debugger like GDB can suffice. The debugger itself

**Copyright@ REST Publisher**

consists of three primary components: events, filters, and recognizers. Events represent the lowest level of system activity observable by the event debugger. Filters serve the purpose of eliminating events that hold no interest to the programmer from the stream of events generated by the program being debugged. Recognizers, on the other hand, play a role in discerning whether the system's behavior is correct or flawed. By combining events at different levels of abstraction, the programmer's task is made more manageable. PDSs have emerged as viable solutions for meeting the ever-increasing demand for enhanced processing power and improved data handling capabilities. To cater to computationally demanding tasks such as weather forecasting and earthquake research, these systems offer a significant boost in computational performance. Creating a PDS, however, is a formidable and intricate task that involves a multitude of challenges. These challenges encompass a range of issues including resource allocation, network efficiency, security, system diversity, fault tolerance, adaptability, scalability, concurrency, and transparency. In essence, the development of a PDS requires tackling a set of intricate hurdles that are distinct to distributed architecture and parallel execution models, challenges that are absent in the context of sequential execution models and centralized architectures [3]. A collection of communication objectives constitutes a distributed computing framework. Throughout this investigation, we will exclusively denote the software as "cooperative processors engaged in distributed computing and operating within a unified goal." This term, "distributed computing framework," pertains to either the system software or the hardware components. In the realm of hardware, key terms include debugging of distributed deployment, debugging from foundational stages, traceability, and a two-step debugging approach. the communication system in its entirety, along with the communication setup. The software constitutes a collection of applications that operate on processors. To establish the shared system, these applications operate as a unified entity. A network of communication objectives forms the basis of a distributed computing system. Within this study, we will simply denote the collection as "distributed computing and collaborating processors functioning as a working system" [4]. All the utilities within the Net Logger Toolkit adhere to a uniform log structure and depend on properly synchronized and precise system clocks. This toolkit comprises three core components: a collection of utilities designed for the compilation and systematic arrangement of log documents, a utility intended for the graphical representation and examination of these log files, and an API as well as a function library aimed at simplifying the process of generating event logs at the application level. [5] Software debugging entails the procedure of pinpointing the underlying reasons behind glitches in a software system and suggesting potential solutions for correction. Because of the scarcity of suitable techniques for debugging intricate software and the complications stemming from the distinct attributes of distributed systems, the process of debugging software systems that are distributed in nature becomes intricate and demanding [Enslow 1978]. Such challenges encompass the inability to reach system components not currently running on the local setup, possession of only incomplete or inaccurate status details, and the time gap between sending requests and receiving responses. [6]. The PVM system serves as a programming environment designed to develop and execute highly parallel or simultaneous programs consisting of numerous interconnected yet distinct modules. It's intended for utilization with a collection of diverse computing units linked through one or multiple networks. Participating processors can encompass scalar machines, multiprocessors, or dedicated-purpose computers. This setup permits the components of applications to operate on the most appropriate algorithm-suited architecture. The fundamental framework makes it possible to run programs within a virtual computing environment that accommodates various parallel computation approaches. Meanwhile, PVM provides an uncomplicated and universal interface that permits the depiction of numerous algorithm types and how they interact with each other. PVM can be adjusted to different hardware setups, incorporates features for running application components concurrently, sequentially, or conditionally, and incorporates specialized techniques for identifying and rectifying errors [7]. An intricate programming environment known as GRADE is in development with the aim of providing advanced graphical assistance for the creation of programs based on PVM (Parallel Virtual Machine). Currently, GRADE supplies tools that enable the creation, execution, debugging, monitoring, and visualization of parallel programs that utilize message-passing techniques. It introduces a fresh graphical language called GRAPNEL and establishes a framework that abstracts high-level graphical programming, facilitating the construction of parallel applications. Throughout the stages of program development and debugging, GRADE maintains a consistent graphical user interface for programmers. When dealing with distributed memory computer systems, a distributed debugging engine (DDBG) is available within GRADE to assist users in debugging GRAPNEL programs. Furthermore, the performance tracking and visualization aspects of parallel programs developed within the GRADE environment are supported by Tape/PVM and PROVE. [8] Crafting accurate distributed systems code can prove to be a daunting task, particularly for novice programmers. Mistakes are nearly inevitable owing to the inherent a synchronicity and the need for robustness against faults. Though industrial-strength testing and model verification have shown success in pinpointing flaws, they come with a cost that surpasses what students can invest in terms of time and energy. With the aim of aiding students in swiftly identifying and rectifying issues in near real-time, we have developed an efficient model verification framework along with a visual debugger designed for distributed systems, specifically addressing this challenge. To enhance the process of detecting errors in student code implementations, we have identified two distinct strategies to narrow down the exploration of the search state space. Drawing from our experiences in utilizing these tools, we have guided over 200 students in constructing a dependable, fault-tolerant key-value store that dynamically shards data.[9] Debugging distributed systems can be a challenging task. Despite the potential for gradual debugging to reveal some issues during the development phase, it's

often difficult for developers to comprehensively test their systems prior to actual deployment in real-world scenarios. The debugging process involves several steps: (i) pinpointing a bug, (ii) collecting the necessary system state for analysis, and (iii) sifting through the collected information to identify the underlying cause. Real-world deployment exposes a system to authentic conditions. In this article, we present Mace ODB, an innovative debugging tool designed for distributed systems, which provides valuable assistance to programmers in this complex endeavor. In their system, developers define a collection of operational characteristics, and Macedo assesses these characteristics while the program is running to detect any breaches. Whenever Macedo detects a breach, it provides the necessary information for developers to comprehend its origins. Through the utilization of Macedo, we successfully pinpointed several significant vulnerabilities in well-established distributed systems. This article delves into two of these issues. Performance assessments indicate that this approach minimally affects system performance and is well-suited for real-time debugging of systems already in operation [10]. VERDI is a visual platform designed for the development, experimentation, and analysis of distributed systems. Within VERDI, designers have the ability to construct a visual model depicting a collection of asynchronous processes and procedures. This is facilitated by VERDI's graphical tools and integrated language editors. The resulting representation can range from a basic implementation to a highly abstract depiction, encompassing a spectrum of possibilities in between. Central to the representation are the design's structures governing control and communication. To evaluate the design's efficacy and functionality, designers can employ VERDI to execute the project. Additionally, the designer can engage in interactive debugging by utilizing the optional feature of animation for execution. This paper comprehensively explores the graphical language, editing, and animation functionalities offered by VERDI. [11] Given the considerable challenges presented by cloud computing, software developers are seeking robust and innovative resources. This encompasses exploring novel programming languages, application frameworks, software packages, visualization platforms, and other utilities. Regardless of the specific route taken, the fundamental core of all these developmental endeavors is comprehending the software's structure and functionality. This paper revolves around an unconventional approach to software comprehension by delving into the possibility of a programmer unfamiliar with a particular distributed computing setup gaining a reasonable comprehension of its operations without direct code examination. We believe that to generate truly innovative methods for comprehending programs, individuals should welcome this manner of thought. Once this perspective is adopted, it becomes necessary to view the implementation of a distributed computation as a subject of empirical research that is open to active exploration [12]. Distributed systems have the capacity to enhance reliability by leveraging built-in redundancy, enhance performance by harnessing parallel processing, and optimize resource utilization through resource sharing, among other recognized benefits. Nevertheless, the absence of appropriate monitoring and measurement utilities has posed challenges in effectively observing the operational patterns and computational efficiency attributes of distributed systems. The intricacy of computer systems and their typical lack of design for monitoring constitute the primary hurdles in the broader evaluation of computer systems. The inherent absence of centralized control, precise global time, and accurate global state in distributed systems contributes significantly to the elevated complexity of the situation. [Reference 13] Software debugging entails users of debugging tools constructing representations of specific elements of program execution. These representations faithfully depict the functioning of the system under analysis. In order to identify significant deviations, these depictions of real system behavior are juxtaposed against projected behavior models that are upheld by system users and creators. Through the process of generating and contrasting these models, the deficiencies in the system become evident. Consequently, recommendations for corrections or further investigation are proposed (Source: [14]).

## 2. MATERIALS & METHODS

**2.1 Federated Array of Bricks:** "Federated Array of Bricks" refers to a computing architecture where multiple individual hardware components or units, often referred to as "bricks," are combined into a cohesive system through a federated approach. Each "brick" typically consists of its own processing power, memory, and storage resources. The term "federated" indicates that these individual units work together as a unified whole, often through networking and coordination mechanisms. This architecture is often used in distributed computing or storage systems, where the individual bricks can be geographically dispersed but still collaborate to perform tasks or store data. The federated approach allows for scalability, fault tolerance, and efficient resource utilization since tasks can be distributed across the various bricks in the system.

**2.2 Split stream":** generally, refers to a process or situation where something is divided or separated into multiple streams or paths. It can be used in various contexts, such as technology, business, and even natural processes. Here are a few examples to help clarify the concept: Video Streaming: In the context of video streaming, "split stream" could refer to the distribution of a single video feed into multiple streams. This might be done to cater to different devices or platforms, providing varying levels of quality based on the capabilities of each device. Data Processing: In data processing, "split stream" could involve dividing a data flow into different paths for parallel processing. This can help in optimizing computational tasks and speeding up data analysis. Business Strategy: In business, "split stream" might refer to dividing resources, efforts, or marketing strategies into different directions or target audiences. For example, a company might split its marketing efforts into two streams, each targeting a different demographic. Fluid Dynamics: In fluid dynamics, "split stream" could refer to the division of a fluid flow into multiple

channels or directions. This might occur in natural processes like river deltas, where water is split into various streams as it flows into the sea. In essence, "split stream" signifies the separation or division of a single flow or entity into multiple streams or paths, often for the purpose of optimization, diversification, or accommodating different requirements. The exact meaning will depend on the context in which it's used.

**2.3 Bullet:** Illustrates a third-generation framework for dispersing materials, where Bullet establishes a network mesh by enabling individual nodes involved in downloading to select multiple peers. This stands in contrast to overlay multicast protocols. This framework allows for the transmission and reception of data. Peers share inventories of the blocks they have received sequentially. In this process, a node can choose to either proactively send a list of accessible blocks to its peers or await a request from another node for such a list (pull). These inventories are transmitted as incremental updates, encompassing only the alterations that have occurred between the two nodes since the previous transmission.

**2.4 Ran Sub:** Acts as a foundational element for more intricate protocols. Constructs a hierarchical arrangement akin to a tree, furnishing each node with information regarding an entirely random portion of the remaining nodes within that tree. Sub's functioning entails two communication stages—distribution and collection. In the distribution phase, commencing from the root, every node imparts information about a randomly chosen subset to its corresponding offspring nodes. Beginning with the lowest-level nodes, each node conveys a concise update on its current state to its immediate parent node in the data collection stage. It's important to highlight that internal nodes only transmit their summarized message after receiving input from all of their offspring nodes. A total of 100 hosts took part in the 5-minute trial, which was conducted as a component of our assessments for the RanSub project.

**2.5 Incident Resolution Time:**" refers to the duration it takes to successfully resolve or address an incident or problem. This metric is used to measure the efficiency and effectiveness of a process or system in resolving issues that may arise in various contexts, such as customer support, technical troubleshooting, or project management. The shorter the incident resolution time, the more quickly and efficiently the problem has been resolved.

**2.6"Lines of code recognizers":** refers to tools or techniques that are designed to identify, count, or analyze the number of lines of code within a software program. These tools can assist in assessing the size and complexity of a codebase, which can be useful for various purposes such as project management, code optimization, and estimating development effort.

**2.7"Recognizers"**: typically refers to devices, software, or systems that are capable of identifying or detecting specific patterns, objects, signals, or elements within a given context. These systems are designed to recognize and interpret data or inputs to provide relevant information or trigger appropriate responses. Recognizers can be used in various fields such as image recognition, speech recognition, handwriting recognition, and more, depending on the specific domain and application.

**2.8"Lines of annotations":** typically refers to explanatory or additional notes that are added to a text, document, or piece of writing. These annotations are usually placed alongside specific lines or sections of the original content and provide explanations, clarifications, insights, or references that help the reader better understand the material. Annotations can also include comments, criticisms, or analyses that contribute to a deeper comprehension of the text's meaning, context, or significance.

**2.9"Number of hosts:"** refers to the quantity or count of individuals, devices, or entities that are part of a network or system. It represents the total number of nodes or endpoints that are connected and active within a particular network environment. This term is commonly used in the context of computer networks and information technology to describe the extent of connectivity within a network.

## 3. WASPAS

The extended WASPAS approach is a brand-new integrated methodology that the researchers present in their work and is based on the WASPAS technique. It may be used to address MCDM problems, including interval type-2 sets of fuzzy values [15]. The time and attendance software issue for the system choice issue for the private hospital is addressed in this research using a combination of the CRITIC and WASPAS approaches. Simple mathematical application steps are required for both techniques. Various parameters for the WASPAS technique result in the identical ranking of the alternatives in this investigation. With different selection issues, the influence of different values may be observed [16]. The present research revisits the notion of rating correctness in the WASPAS tackle and re-derives the essential equations. It has been discovered that derivatives cannot be reliably calculated when the anticipated variation of the WPM is being used. Consequently, a modified equation is suggested, and the outcomes are supported by two examples. For the purpose of to assist practitioners in calculating estimated variances and selecting the perfect parameter, programmes for computers are also supplied [17]. We have created an additional extension of the classic WASPAS system within the MCDM framework. The single-valued neutrosophic collection's conceptual framework is used in the new extension that is being suggested, called WASPAS-SVNS. It has been performed in order to contrast the outcomes of other MCDM techniques. According to the calculations, Vilnius' Gariunai District is the best location for the plant to be built in order to burn non-hazardous garbage. Based on the findings, it can be said that this area is appropriate for the execution of the project to build a waste incinerator plant [18]. Numerous MCDM approaches have been enhanced for IVIFSs as as a consequence of rent areas. This research develops a novel WASPAS method-based approach under IVIFSs. The

developed technique is based on the IVIFS operators, some modifications to the conventional WASPAS method, and an innovative method for calculating the weights of the decision experts and the criteria. For the purpose to arrive at accurate weights, novel techniques have been suggested to calculate the weights of the choices made by experts and criterion using interval-valued intuitionistic fuzzy data values (entropy, divergence, and similarity measures)[19].To assess the alternatives, a suggested solution technique and a weighting calculation of the criteria established within the parameters of the suggested model were produced[20].Owing to the WASPAS findings, the best means of promoting and pursuing Iran's industrial planning for the future is to take advantage of nanotechnology in both health and healthcare[21].The performance of TSPs is compared in the current study using the intuitively fuzzy weighted aggregate sum evaluation (IF-WASPAS) method. The IF-WASPAS method integrates the "weighted sum (WSM)" and product model (WPM) for processes of decision-making, as well as linguistic uncertainty in MCDM. Furthermore, the objective weights obtained using the similarity measurement method will be added to the personalized weights stated by the professionals in order to determine criterion weights [22]. The station choosing problem for gas stations is examined applying the newly proposed spherical fuzzy AHP-integrated spherical WASPAS method [23].

## 4. RESULT AND DISCUSSION

**TABLE 1.** Graphical interactive is debugging for distributed systems

| system | Lines of code recogniers | Recognizers | Lines of annotations | Number of hosts |
|---|---|---|---|---|
| FAB | 124020 | 15 | 28 | 4 |
| SplitStream | 2435 | 20 | 9 | 100 |
| Bullet | 2450 | 1 | 23 | 100 |
| RanSub | 1700 | 8 | 35 | 100 |
| Incident Resolution Time | 1500 | 5 | 38 | 100 |

Table 1 presented table offers insights into several distinct distributed systems and their attributes pertaining to graphical interactive debugging. Focusing on key parameters, each system's effectiveness and complexity in debugging become apparent. For instance, the "Lines of Code Recognizers" metric indicates the extent of code segments dedicated to recognizing specific elements within the systems. The "Recognizers" count highlights the number of distinct algorithms or components employed for identification purposes. Additionally, the "Lines of Annotations" metric underscores the emphasis on explanatory comments or notes within the codebase, enhancing comprehensibility. Notably, the "Number of Hosts" statistic signifies the scale of each distributed system, revealing the extent of its distribution across different nodes. Altogether, this data-rich tableau paints a picture of how these systems are structured for debugging in distributed environments. If there are specific inquiries about the implications of this information, further exploration is encouraged.

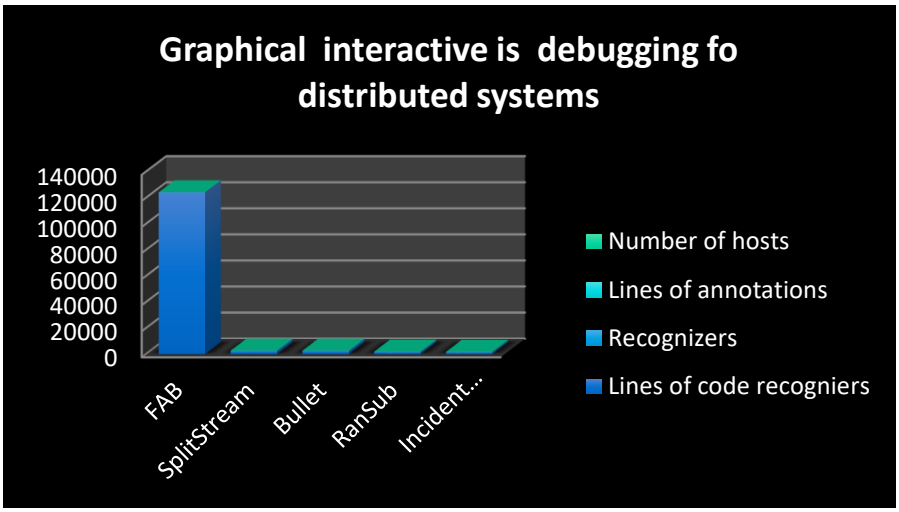


**FIGURE 1.** Graphical interactive is debugging for distributed systems

Figure 1 illustrates how the Graphical interactive is debugging for distributed systems are done.

**TABLE 2. Performance value**

| Performance value | | | |
|---|---|---|---|
| 1.00000 | 0.75000 | 0.32143 | 1.00000 |
| 0.01963 | 1.00000 | 1.00000 | 0.04000 |
| 0.01975 | 0.05000 | 0.39130 | 0.04000 |
| 0.01371 | 0.40000 | 0.25714 | 0.04000 |
| 0.01209 | 0.25000 | 0.23684 | 0.04000 |

Table 2 shows the performance value for weighted sum method

**TABLE 3.** Weight Matrix

| Weight | | | |
|---|---|---|---|
| 0.25 | 0.25 | 0.25 | 0.25 |
| 0.25 | 0.25 | 0.25 | 0.25 |
| 0.25 | 0.25 | 0.25 | 0.25 |
| 0.25 | 0.25 | 0.25 | 0.25 |
| 0.25 | 0.25 | 0.25 | 0.25 |

Table 3 shows the weight matrix which is taken as same for all the systems

**TABLE 4.** Weighted normalized decision matrix by WSM

| Weighted normalized decision matrix | | | |
|---|---|---|---|
| 0.25000 | 0.18750 | 0.08036 | 0.25000 |
| 0.00491 | 0.25000 | 0.25000 | 0.01000 |
| 0.00494 | 0.01250 | 0.09783 | 0.01000 |
| 0.00343 | 0.10000 | 0.06429 | 0.01000 |
| 0.00302 | 0.06250 | 0.05921 | 0.01000 |

Table 4 shows the weight normalized decision matrix for the four types of the alternate parameters and five of the evaluation parameters. It is done using the weight sum method

**TABLE 5.** Weight Normalized Decision Matrix by WPM

| Weighted normalized decision matrix | | | |
|---|---|---|---|
| 0.98268 | 0.93014 | 0.95846 | 0.94507 |

| | | | |
|---|---|---|---|
| 0.96681 | 0.93582 | 0.92440 | 0.89593 |
| 0.92195 | 0.90051 | 0.95821 | 0.93414 |
| 0.91311 | 0.91080 | 1.00000 | 1.00000 |
| 1.00000 | 1.00000 | 0.96850 | 0.98233 |

Table 5 shows the weight normalized decision matrix for the five types of the alternate parameters and four of the evaluation parameters. It is done using the weight product method

**TABLE 6.** Preference Score

| Preference Score Wsm | Preference Score Wpm |
|---|---|
| 0.76786 | 0.82794 |
| 0.51491 | 0.74932 |
| 0.12526 | 0.74314 |
| 0.17771 | 0.83166 |
| 0.13473 | 0.95139 |

Table 6 shows the Preference Score for five types of systems. The score is evaluated using two methods i) Weight Sum method ii) Weight Product method.

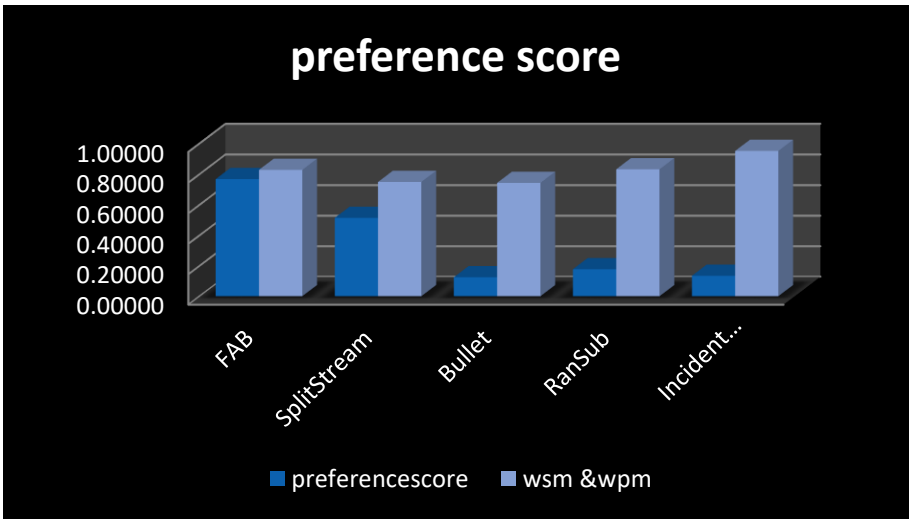



**FIGURE 2.** preference score

Figure 3 illustrates the preference score for the alternatives using weight sum and weight product method

**TABLE 7.** WASPAS Coefficient

| System | WASPAS Coefficient |
|---|---|
| FAB | 0.79790 |
| SplitStream | 0.63211 |
| Bullet | 0.43420 |
| RanSub | 0.50469 |
| Incident Resolution Time | 0.54306 |

Table 7 shows the WASPAS Coefficient for the Alternatives used here.

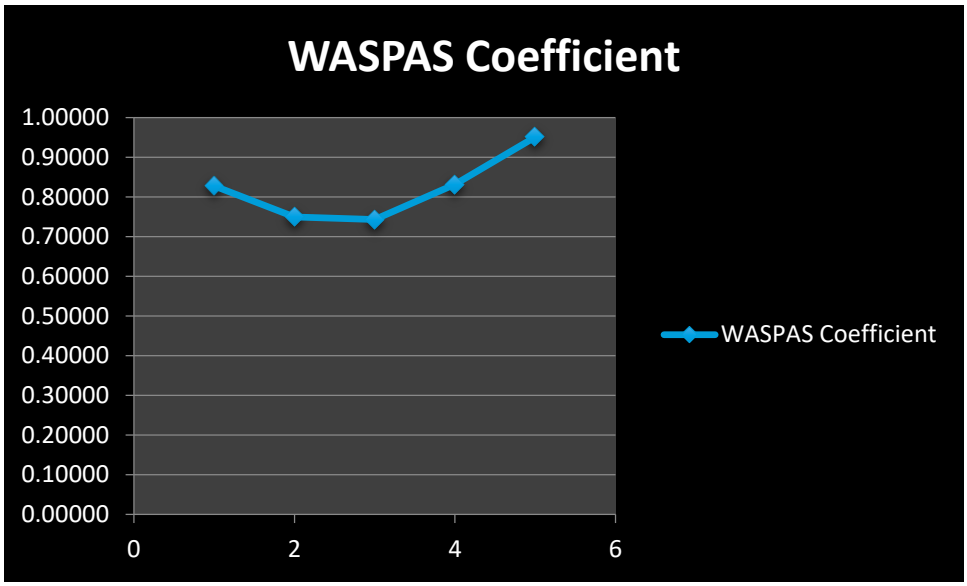



**FIGURE 3.** WASPAS Coefficient

**TABLE 8.** Rank

| System | Rank |
|---|---|
| FAB | 1 |
| SplitStream | 2 |
| Bullet | 5 |
| RanSub | 4 |
| Incident Resolution Time | 3 |

Table 8 shows the Rank for the five types of alternatives used in this research

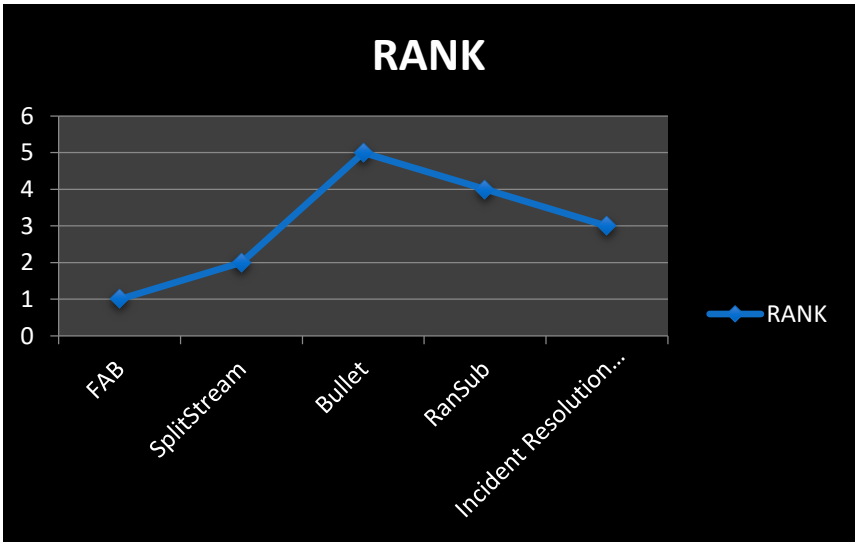



**FIGURE 4.** Rank for the Graphical interactive is debugging for distributed systems

Figure 5 illustrates the rank for the Graphical interactive is debugging for distributed systems. Here the "FAB" is in the 1st

rank, Split Stream is in the 2nd rank, Incident Resolution Time is in the 3rd rank, Ran Sub is in the 4th rank, Bullet is in the 5th rank.

## 5. CONCLUSION

Concluding the concept of "graphical interactive debugging for distributed systems," it can be inferred that this approach offers a visual and interactive method to diagnose and rectify issues within distributed systems. By utilizing graphical representations of system components, communication pathways, and data flows, developers gain a better understanding of the system's behavior. This assists in identifying bottlenecks, failures, and performance problems that might not be immediately evident through traditional debugging methods. The interactive aspect allows real-time manipulation and observation of system elements, aiding in pinpointing the root causes of problems and improving the overall efficiency and reliability of distributed systems. Graphical interactive debugging for distributed systems is a valuable approach to simplify the otherwise complex process of debugging such systems. These tools offer visual insights into the interactions and behaviors of components, helping developers diagnose problems efficiently. However, it's important to note that while these tools can be highly effective, they are not a one-size-fits-all solution. Distributed systems debugging remains challenging due to factors like network latency, asynchrony, and the sheer complexity of interactions. Therefore, while graphical interactive debugging can greatly assist in the process, it's still essential for developers to have a strong understanding of distributed systems concepts and debugging techniques to effectively resolve issues in these complex environments.

## REFERENCES


[1]. Reynolds, Patrick, Charles Edwin Killian, Janet L. Wiener, Jeffrey C. Mogul, Mehul A. Shah, and Amin Vahdat. "Pip: Detecting the Unexpected in Distributed Systems." In NSDI, vol. 6, pp. 9-9. 2006.

[2]. Joyce, Jeffrey, Greg Lomow, Konrad Slind, and Brian Unger. "Monitoring distributed systems." ACM Transactions on Computer Systems (TOCS) 5, no. 2 (1987): 121-150.

[3]. Agrawal, Vikash K., Srinivasa Rao Bogireddy, Lalit N. Patil, Mahesh Sonekar, Yashraj M. Patil, and Vikas Singh Panwar. "Optimizing Handwritten Character Recognition Systems: Neural Networks vs. Statistical Methods." In *2024 International Conference on Intelligent Systems and Advanced Applications (ICISAA)*, pp. 1-9. IEEE, 2024.

[4]. Sulistio, Anthony, Chee Shin Yeo, and Rajkumar Buyya. "A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools." Software: Practice and Experience 34, no. 7 (2004): 653-673.

[5]. Garcia-Molina, Hector, Frank Germano, and Walter H. Kohler. "Debugging a distributed computing system." IEEE Transactions on Software Engineering 2 (1984): 210-219.

[6]. Manikandan, G., D. Karunkuzhali, D. Geetha, and V. Kavitha. "Design of an IoT approach for security surveillance system for industrial process monitoring using Raspberry-Pi." In *AIP Conference Proceedings*, vol. 2519, no. 1, p. 030024. AIP Publishing LLC, 2022.

[7]. Gunter, Dan, Brian Tierney, Brian Crowley, Mason Holding, and Jason Lee. "Netlogger: A toolkit for distributed system performance analysis." In Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (Cat. No. PR00728), pp. 267-273. IEEE, 2000.

[8]. Bates, Peter C. "Debugging heterogeneous distributed systems using event-based models of behavior." ACM Transactions on Computer Systems (TOCS) 13, no. 1 (1995): 1-31.

[9]. Jyothi, P., and G. Pradeepini. "Heart disease detection system based on ECG and PCG signals with the aid of GKVDLNN classifier." *Multimedia Tools and Applications* 83, no. 10 (2024): 30587-30612.

[10]. Sunderam, Vaidy S. "PVM: A framework for parallel distributed computing." Concurrency: practice and experience 2, no. 4 (1990): 315-339.

[11]. Kacsuk, Péter, JoséC Cunha, Gábor Dózsa, João Lourenço, Tibor Fadgyas, and Tiago Antao. "A graphical development and debugging environment for parallel programs." Parallel Computing 22, no. 13 (1997): 1747-1770.

[12]. Michael, Ellis, Doug Woos, Thomas Anderson, Michael D. Ernst, and Zachary Tatlock. "Teaching rigorous distributed systems with efficient model checking." In Proceedings of the Fourteenth EuroSys Conference 2019, pp. 1-15. 2019.

[13]. Dao, Darren, Jeannie Albrecht, Charles Killian, and Amin Vahdat. "Live debugging of distributed systems." In Compiler Construction: 18th International Conference, CC 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings 18, pp. 94-108. Springer Berlin Heidelberg, 2009.

[14]. Shen, Vincent Y., Charles Richter, Michael L. Graf, and Jeffrey A. Brumfield. "VERDI: A visual environment for designing distributed systems." Journal of Parallel and Distributed Computing 9, no. 2 (1990): 128-137.

[15]. Hart, Delbert, Eileen Kraemer, and G-C. Roman. "Interactive visual exploration of distributed computations." In Proceedings 11th International Parallel Processing Symposium, pp. 121-127. IEEE, 1997.

[16]. Robinson Joel, M., G. Manikandan, G. Bhuvaneswari, and P. Shanthakumar. "SVM-RFE enabled feature selection with DMN based centroid update model for incremental data clustering using COVID-19." *Computer Methods in Biomechanics and Biomedical Engineering* 27, no. 10 (2024): 1224-1238.

[17]. Bogireddy, Srinivasa Rao, and Haritha Murari. "Comparing the Performance of Classification Algorithms in the Conservation of Plant Genetic Resources for Agriculture and Food Security." In *2024 IEEE International Conference on Agrosystem Engineering, Technology & Applications (AGRETA)*, pp. 116-122. IEEE, 2024.


**Copyright@ REST Publisher** 

[18]. Bharani, Neha, and Abhay Kothari. "Tools for analysis of various static software complexities for mat lab code." *Turkish Online Journal of Qualitative Inquiry* 12, no. 6 (2021).
[19]. Lange, Frank, Reinhold Kroeger, and Martin Gergeleit. "JEWEL: Design and implementation of a distributed measurement system." IEEE Transactions on Parallel and Distributed Systems 3, no. 6 (1992): 657-671.
[20]. Mekala, Sreenivas, Y. Balarama Krishna, M. Nagaraju, P. Abhiram, K. Pavan Mahith, and Y. Balarama Krishna. *CARDIOPATHY-HEART DISEASE PREDICTION USING MACHINE LEARNING*. 2023.
[21]. Bates, Peter. "Distributed debugging tools for heterogeneous distributed systems." In The 8th International Conference on Distributed, pp. 308-309. IEEE Computer Society, 1988.
[22]. Jyothi, P., and G. Pradeepini. "Classification of normal/abnormal heart sound recording through convolution neural network through the integration of baseline and adaboost classifier." In *Proceedings of the 2nd International Conference on Computational and Bio Engineering: CBE 2020*, pp. 441-447. Singapore: Springer Singapore, 2021.
[23]. Mohan, VakaMurali, MalliKarjuna Reddy, and KRN Kiron Kumar. "A New Approach to Optical Networks Security: Attack-Aware Routing and Wavelength Assignment." In *IJCA Special Issues on "2nd National Conference-Computing, Communication and Sensor Network" CCSN*. 2011.
[24]. Bharani, Neha. *Complexity study of software engineering phases and software quality*. Book Rivers, 2022.
[25]. Annapurna, D., N. Tejas, K. B. Raja, K. R. Venugopal, and L. M. Patnaik. "An energy efficient multicast algorithm for an Adhoc network using network coding and MAC scheduling." In *2013 international conference on signal processing and communication (ICSC)*, pp. 62-67. IEEE, 2013.
[26]. Ghorabaee, Mehdi Keshavarz, EdmundasKazimierasZavadskas, Maghsoud Amiri, and Ahmad Esmaeili. "Multi-criteria evaluation of green suppliers using an extended WASPAS method with interval type-2 fuzzy sets." Journal of Cleaner Production 137 (2016): 213-229.
[27]. Tuş, Ayşegül, and EsraAytaçAdalı. "The new combination with CRITIC and WASPAS methods for the time and attendance software selection problem." Opsearch 56 (2019): 528-538.
[28]. Subashini, G. S., and S. M. Kumar. "An investigation on adoption of lean production principles in kitchenware manufacturing industries." *Interdisciplinary Journal of Contemporary Research in Business* 4, no. 9 (2013): 271-279.
[29]. Challa, Ramya, Reethika Bijjala, and Mekala Sreenivas. "Breast Cancer Prediction Using Machine Learning." *International Journal for Research in Applied Science and Engineering Technology* (2022).
[30]. Baykasoğlu, Adil, and İlkerGölcük. "Revisiting ranking accuracy within WASPAS method." Kybernetes 49, no. 3 (2020): 885-895.
[31]. Jones, Christonson Berin, and Chakravarthi Murugamani. "Malaria parasite detection on microscopic blood smear images with integrated deep learning algorithms." *Int. Arab J. Inf. Technol.* 20, no. 2 (2023): 170-179.
[32]. Priya, R. "Stress coping mechanisms adopted by banking sector employees for downsize the stress: A study." *IOSR J Busi Mgt* 19 (2017): 13-7.
[33]. KazimierasZavadskas, Edmundas, RomualdasBaušys, and Marius Lazauskas. "Sustainable assessment of alternative sites for the construction of a waste incineration plant by applying WASPAS method with single-valued neutrosophic set." Sustainability 7, no. 12 (2015): 15923-15936.
[34]. Mishra, Arunodaya Raj, and Pratibha Rani. "Interval-valued intuitionistic fuzzy WASPAS method: application in reservoir flood control management policy." Group Decision and Negotiation 27 (2018): 1047-1078.
[35]. Yücenur, G. Nilay, and Ahmet Ipekçi. "SWARA/WASPAS methods for a marine current energy plant location selection problem." Renewable Energy 163 (2021): 1287-1298.
[36]. Praiselin, EJ Honesty, Dr G. Manikandan, Vilma Veronica, and Ms S. Hemalatha. "Sign language detection and recognition using media pipe and deep learning algorithm." *International Journal of Scientific Research in Science and Technology* 11, no. 2 (2024): 123-130.
[37]. Kumar, A. Ranjith, and A. Sivagami. "Balanced Load Clustering with Trusted Multipath Relay Routing Protocol for Wireless Sensor Network." In *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*, vol. 1, pp. 1-6. IEEE, 2019.
[38]. Varma, P. Bharat Siva, Prathap Adimoolam, Yamini Lahari Marna, Anantharamaiah Vengala, VS Divya Sundar, and MVT Ram Pavan Kumar. "Enhancing robust object detection in weather-impacted environments using deep learning techniques." In *2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, pp. 599-604. IEEE, 2024.
[39]. Inaganti, Mr Rambabu. "Preserving Patient Privacy: A Focus on Cyber security in Healthcare." (2023).
[40]. Ghorshi Nezhad, Mohammad Reza, Sarfaraz HashemkhaniZolfani, FathollahMoztarzadeh, EdmundasKazimierasZavadskas, and Mohsen Bahrami. "Planning the priority of high-tech industries based on SWARA-WASPAS methodology: The case of the nanotechnology industry in Iran." Economic research-Ekonomskaistraživanja 28, no. 1 (2015): 1111-1137.
[41]. Rani, Dr V. Vasudha, D. Vasavi, and K. Kumar. "Significance of multilayer perceptron model for early detection of diabetes over ml methods." *J. Univ. Shanghai Sci. Technol* 23, no. 08 (2021): 148-160.
[42]. Vidhya Prasanth, Sathiyaraj Chinnasamy, M. Ramachandran, Chinnasami Sivaji, "A Review on Cooperative Behavior and Colony Optimization in Ants", Building Materials and Engineering Structures, 2(2), June 2024, 7-11.
[43]. Mishra, Arunodaya Raj, Rahul Kumar Singh, and Deepak Motwani. "Multi-criteria assessment of cellular mobile telephone service providers using intuitionistic fuzzy WASPAS method with similarity measures." Granular Computing 4 (2019): 511-529.
[44]. Bharani, Neha, and Abhay Kothari. "Tools for analysis of various static software complexities for mat lab code." *Turkish Online Journal of Qualitative Inquiry* 12, no. 6 (2021).

[45]. Murugamani, C., Santosh Kumar Sahoo, Pravin R. Kshirsagar, Boppuru Rudra Prathap, Saiful Islam, Quadri Noorulhasan Naveed, Mohammad Rashid Hussain, Bui Thanh Hung, and Dawit Mamiru Teressa. "Wireless communication for robotic process automation using machine learning technique." *Wireless Communications and Mobile Computing* 2022, no. 1 (2022): 4723138.

[46]. Singh, Jagbir, Priyankkumar Patel, Balaji Shesharao Ingole, Rambabu Inaganti, Vishnu Ramineni, Manjunatha Sughaturu Krishnappa, and Bhushan Jayeshkumar Patel. "Advanced Computational Methods for Pelvic Bone Cancer Detection: Efficacy comparison of Convolutional Neural Networks." In *2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 287-293. IEEE, 2024.

[47]. Ayyildiz, Ertugrul, and Alev Taskin Gumus. "A novel spherical fuzzy AHP-integrated spherical WASPAS methodology for petrol station location selection problem: a real case study for İstanbul." Environmental Science and Pollution Research 27, no. 29 (2020): 36109-36120.

[48]. Swamy, M. B., and D. R. Priya. "The Measurement Levels of Financial Literacy among Postgraduate Management Students: An Empirical Study in Andhra Pradesh State." *IOSR Journal of Business and Management* 19, no. 06 (2017): 55-65.

[49]. Arunadevi, R., and G. S. Subashini. "Virtual Teams Effectiveness In The Selected IT Companies In South India." *Journal of Namibian Studies* 38 (2023).

[50]. Kumar, KRN Kiran, and K. Bhavani. "Folded spined cube: new topology in interconnection networks." In *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 314-319. IEEE, 2022.

[51]. Ragavan, V. K., N. S. Nithya, Anantharamaiah Vengala, Balambigai Subramanian, and C. Ambhika. "Refractive Index Biosensor–Based Detection of Mycobacterium Tuberculosis Using Sea Lion Political Optimizer and Deep Learning." *Plasmonics* (2025): 1-19.

[52]. Manjula selvam, Vidhya Prasanth, M. Ramachandran, Ramya Sharama, "A Study on Economic Models of Animal Communication Methods" Journal on Innovations in Teaching and Learning, 3(2), June 2024, 13-19.

[53]. Kumar, A. Ranjith, R. Vinoth, and Padmanathan Kasinathan. "Opposition Based Artificial Flora Algorithm for Load Balancing in LTE Network." *Wireless Personal Communications* 118, no. 1 (2021): 141-159.

[54]. Kumar, Naveen, Jyoti R. Desai, and D. Annapurna. "ACHs-LEACH: Efficient and Enhanced LEACH protocol for wireless sensor networks." In *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1-6. IEEE, 2020.

**Copyright@ REST Publisher** 