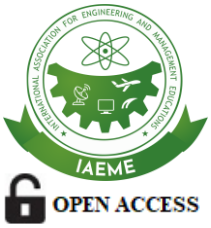
International Journal of DevOps (IJDO)
Volume 1, Issue 1, January-June 2024, pp. 1-10, Article ID: IJDO_01_01_001
Available online at https://iaeme.com/Home/issue/IJDO?Volume=1&Issue=1
Journal ID: 1552-5541



GENETIC PROGRAMMING FOR AUTOMATED ERROR HANDLING AND RECOVERY IN DEVOPS ENVIRONMENTS

Srividhya S
Independent Researcher, India

ABSTRACT

Efficient error handling and recovery mechanisms are critical in contemporary DevOps environments to maintain system reliability and operational efficiency. Traditional methodologies often prove inadequate in addressing the complexities of modern distributed systems, sparking interest in advanced techniques such as Genetic Programming (GP) algorithms for automation. This research investigates the potential of GP algorithms in automating error handling and recovery within DevOps, aiming to develop self-healing systems capable of autonomously detecting, diagnosing, and mitigating errors. Focused on delineating, executing, and assessing GP algorithms tailored for error management, this inquiry aims to elucidate challenges and prospects in this domain. Through a comprehensive approach encompassing literature review, theoretical analysis, and empirical case studies, this study delves into the fundamental tenets of GP algorithms and their relevance to DevOps. While GP algorithms show promise for enhancing system reliability and efficiency, addressing challenges such as scalability and interpretability remains crucial for their effective integration into DevOps practices.

Keywords: DevOps, Genetic Programming, Error Handling, Recovery, Automation, Self-healing Systems, Distributed Systems, Operational Efficiency, System Reliability.

Cite this Article: Srividhya S, Genetic Programming for Automated Error Handling and Recovery in DevOps Environments. International Journal of DevOps (IJDO). 1(1), 2024, 1-10.

Available online at https://iaeme.com/Home/issue/IJDO?Volume=1&Issue=1

1. INTRODUCTION

In today's rapidly evolving technological landscape, the integration of Development (Dev) and Operations (Ops) practices has become essential for organizations aiming to achieve faster delivery of software products while maintaining high levels of reliability and efficiency. This integration, termed DevOps, emphasizes collaboration, automation, and continuous feedback throughout the software development lifecycle.

While DevOps offers numerous benefits, such as increased deployment frequency, reduced lead time for changes, and lower failure rates of new releases, it also introduces unique challenges, particularly in error handling and recovery. DevOps environments typically involve complex distributed systems composed of interconnected services, containers, and infrastructure components. As a result, errors and failures are inevitable, stemming from various sources such as software bugs, configuration issues, infrastructure failures, or unexpected user behaviors.

Traditional error handling approaches often rely on manual intervention or predefined scripts, which can be time-consuming, error-prone, and insufficiently adaptive to handle the dynamic nature of modern DevOps ecosystems. Consequently, there is a growing interest in leveraging advanced techniques, such as Genetic Programming (GP), to automate error detection and recovery processes in DevOps environments.

Genetic Programming, a subtype of evolutionary algorithms, mimics the process of natural selection to evolve solutions to complex problems. In the context of DevOps, Genetic Programming algorithms can be trained to analyze system logs, monitor performance metrics, and dynamically adapt error-handling strategies based on historical data and real-time feedback.

This research paper aims to explore the potential of Genetic Programming for automated error handling and recovery in DevOps environments. By employing Genetic Programming techniques, organizations can strive towards achieving self-healing systems capable of detecting, diagnosing, and mitigating errors autonomously, thus enhancing system resilience, reducing downtime, and improving overall operational efficiency.

2. OVERVIEW OF GENETIC PROGRAMMING ALGORITHMS IN DEVOPS

Genetic Programming (GP) is a powerful technique inspired by biological evolution that is increasingly being applied in various domains, including software engineering and operations management. In the context of DevOps, GP algorithms offer a promising approach for automating error handling and recovery processes.

At its core, Genetic Programming operates by simulating the process of natural selection and evolution to iteratively generate and refine solutions to complex problems. In the context of DevOps, GP algorithms can be tailored to analyze system logs, monitor performance metrics, and dynamically adapt error-handling strategies based on historical data and real-time feedback.

The key components of a Genetic Programming algorithm include:

i. Population: A set of candidate solutions represented as individuals or chromosomes. In the context of DevOps, these individuals could represent different error-handling strategies or recovery mechanisms.
ii. Fitness Function: A metric used to evaluate the performance of each individual within the population. In DevOps, the fitness function may assess the effectiveness of error handling and recovery strategies based on criteria such as response time, system uptime, and the number of incidents resolved.
iii. Genetic Operators: Mechanisms for generating new solutions by mimicking genetic operations such as mutation, crossover, and selection. These operators enable the exploration of the solution space and the evolution of increasingly effective error-handling strategies.
iv. Evolutionary Process: The iterative process of generating new populations of solutions by applying genetic operators, evaluating their fitness, and selecting individuals for reproduction based on their performance. Over successive generations, the algorithm converges towards solutions that are well-adapted to the problem at hand.

In the context of DevOps, Genetic Programming algorithms can be applied to automate various aspects of error handling and recovery, including:

- **Log Analysis:** Automatically parsing system logs to identify error patterns and anomalies, enabling proactive error detection and diagnosis.
- **Dynamic Configuration**: Adapting system configurations and resource allocations in response to changing workload demands or failure scenarios.
- **Fault Tolerance**: Implementing resilient design patterns and fault-tolerant mechanisms to mitigate the impact of errors and failures on system performance.
- **Automated Remediation**: Executing predefined or dynamically generated scripts to recover from errors and restore system functionality without manual intervention.

By leveraging Genetic Programming algorithms in DevOps environments, organizations can enhance their ability to detect, diagnose, and recover from errors autonomously, thereby improving system reliability, reducing downtime, and enhancing overall operational efficiency.

3. DESIGNING GENETIC PROGRAMMING ALGORITHMS FOR ERROR DETECTION

In the domain of DevOps, ensuring accurate error detection is paramount for maintaining system reliability and minimizing disruptions to operations. Genetic Programming (GP) algorithms offer a versatile approach to automating this process, leveraging evolutionary principles to evolve efficient solutions. Designing GP algorithms for error detection involves several key considerations aimed at maximizing detection accuracy while minimizing false positives and false negatives.

One critical aspect of designing GP algorithms for error detection is the selection and representation of features or variables that characterize the system's behavior. These features serve as the building blocks for constructing candidate solutions within the GP framework. In DevOps environments, relevant features may include system performance metrics, log data, event streams, configuration parameters, and external factors such as user interactions or network conditions. Careful consideration must be given to the selection of features to ensure they capture the pertinent aspects of system behavior and provide discriminatory power for distinguishing normal operation from anomalous behavior indicative of errors.

Once the feature set is defined, the next step is to devise an appropriate fitness function that quantifies the effectiveness of candidate solutions in detecting errors. The fitness function serves as the guiding metric for evaluating and selecting individuals within the GP population. In the context of error detection, the fitness function may be defined based on criteria such as detection accuracy, sensitivity, specificity, and robustness to noise or variability in the system environment. Balancing these competing objectives requires careful tuning of the fitness function parameters to strike an optimal trade-off between detection performance and computational efficiency.

The genetic operators employed in the GP algorithm play a crucial role in exploring the solution space and facilitating the evolution of effective error detection strategies. Mutation, crossover, and selection operators are used to generate new candidate solutions by iteratively modifying and recombining existing individuals within the population. In the context of error detection, these operators may be tailored to introduce diversity, promote exploration of different feature combinations, and adapt to changing error patterns over time. Additionally, techniques such as elitism and diversity preservation may be employed to preserve promising solutions and prevent premature convergence to suboptimal solutions.

An essential aspect of designing GP algorithms for error detection is the incorporation of domain knowledge and expertise to guide the search process effectively. Domain-specific

insights can inform the selection of features, the design of fitness functions, and the interpretation of GP-generated solutions. Moreover, domain knowledge can help identify relevant error patterns, anomaly signatures, and contextual cues that may aid in the development of more discriminative and robust error detection models. Collaboration between domain experts, data scientists, and DevOps practitioners is essential to ensure that GP algorithms are tailored to the specific needs and requirements of the target application domain.

Designing effective GP algorithms for error detection in DevOps environments involves careful consideration of feature selection, fitness function design, genetic operator selection, and domain knowledge integration. By leveraging the inherent capabilities of GP algorithms to evolve solutions iteratively, organizations can enhance their ability to detect errors proactively, mitigate risks, and maintain high levels of system reliability and performance in dynamic and complex operational environments.

4. EVOLUTIONARY STRATEGIES FOR AUTOMATED ERROR RECOVERY IN DEVOPS ENVIRONMENTS

In the dynamic landscape of DevOps environments, the ability to swiftly recover from errors is crucial for maintaining system reliability and minimizing service disruptions. Traditional approaches to error recovery often rely on manual intervention or pre-scripted responses, which can be time-consuming, error-prone, and insufficiently adaptable to the complexities of modern distributed systems. Evolutionary strategies, such as Genetic Programming (GP), offer a promising avenue for automating error recovery processes by leveraging principles of natural selection and adaptation to evolve effective recovery strategies.

One of the key advantages of evolutionary strategies in automated error recovery is their ability to adapt and evolve over time in response to changing environmental conditions and error patterns. Unlike static rule-based approaches, evolutionary algorithms can dynamically adjust recovery strategies based on real-time feedback and historical data, enabling systems to learn from past experiences and continuously improve their resilience to errors.

Evolutionary strategies for automated error recovery typically involve the following components:

i. **Candidate Solution Representation**: In the context of error recovery, candidate solutions represent various recovery actions or strategies that can be applied in response to specific error scenarios. These solutions may encompass a range of actions, including restarting services, rolling back deployments, reallocating resources, or triggering failover mechanisms.

ii. **Fitness Evaluation:** The effectiveness of candidate solutions is evaluated using a fitness function that quantifies their ability to recover from errors while minimizing service downtime, data loss, or customer impact. Fitness functions may consider metrics such as recovery time, system availability, data integrity, and user satisfaction to assess the overall performance of recovery strategies.

iii. **Genetic Operators:** Evolutionary algorithms employ genetic operators, such as mutation, crossover, and selection, to generate new candidate solutions by iteratively modifying and combining existing solutions within a population. These operators enable exploration of the solution space and facilitate the discovery of novel and effective recovery strategies.

iv. **Adaptation and Learning**: One of the strengths of evolutionary strategies is their ability to adapt and learn from past experiences. By incorporating feedback mechanisms and reinforcement learning techniques, evolutionary algorithms can continuously refine and improve recovery strategies based on observed outcomes and performance feedback.

v. **Integration with Monitoring and Alerting Systems**: Effective error recovery relies on timely detection and diagnosis of errors. Evolutionary strategies are often integrated with monitoring and alerting systems to enable proactive error detection and trigger automated recovery actions in response to detected anomalies or failures.

By leveraging evolutionary strategies for automated error recovery, organizations can enhance their resilience to errors, reduce the impact of service disruptions, and improve overall system reliability in DevOps environments. Moreover, the adaptive nature of evolutionary algorithms enables systems to evolve and self-optimize over time, ensuring that error recovery strategies remain effective in the face of evolving threats and operational challenges.

5. EVALUATION METRICS AND PERFORMANCE ANALYSIS OF GENETIC PROGRAMMING ALGORITHMS

Assessing the effectiveness and performance of Genetic Programming (GP) algorithms in the context of error handling and recovery within DevOps environments requires the careful selection of evaluation metrics and comprehensive performance analysis. These metrics and analyses provide insights into the capabilities, limitations, and overall efficacy of GP-based approaches for automating error management tasks.

Detection Accuracy: One fundamental metric for evaluating the performance of GP algorithms is the detection accuracy, which measures the proportion of errors correctly identified by the system. High detection accuracy indicates the algorithm's ability to effectively identify and flag errors, thereby enabling prompt remediation actions.

Sample Algorithm:

```
def calculate_detection_accuracy(true_positives, true_negatives, total_instances):
    return (true_positives + true_negatives) / total_instances
```

False Positive Rate: The false positive rate measures the frequency of false alarms or incorrect error identifications by the system. Minimizing the false positive rate is crucial for avoiding unnecessary disruptions and ensuring that error recovery efforts are focused on genuine issues.

Sample Algorithm:

```
def calculate_false_positive_rate(false_positives, true_negatives):
    return false_positives / (false_positives + true_negatives)
```

False Negative Rate: Conversely, the false negative rate quantifies the rate of missed errors or failures that go undetected by the system. Minimizing the false negative rate is essential for maintaining high levels of system reliability and preventing critical issues from going unnoticed.

Sample Algorithm:

```
def calculate_false_negative_rate(false_negatives, true_positives):
    return false_negatives / (false_negatives + true_positives)
```

Response Time: The response time metric measures the time taken by the system to detect errors and initiate appropriate recovery actions. Minimizing response time is critical for reducing the impact of errors on system performance and user experience.

Sample Algorithm:

```
def calculate_response_time(start_time, end_time):
    return end_time - start_time
```

Resource Utilization: Evaluating the resource utilization of GP algorithms provides insights into their computational efficiency and scalability. Efficient resource utilization ensures that

GP-based error management systems can operate effectively in resource-constrained environments without excessive overhead.

Sample Algorithm:

```
def calculate_resource_utilization(cpu_utilization, memory_utilization):
    return (cpu_utilization + memory_utilization) / 2
```

Robustness to Noise: Robustness measures the ability of GP algorithms to maintain performance in the presence of noise or variability in the system environment. Robust algorithms exhibit consistent performance across different operating conditions and are less susceptible to fluctuations in input data.

Sample Algorithm:

```
def calculate_robustness(true_positives, false_positives, false_negatives, true_negatives):
    total_instances = true_positives + false_positives + false_negatives + true_negatives
    return (true_positives + true_negatives) / total_instances
```

By systematically evaluating GP algorithms using these metrics and conducting comprehensive performance analyses, organizations can gain valuable insights into the strengths, weaknesses, and areas for improvement of GP-based error management systems. This information can inform decision-making processes, drive optimization efforts, and ultimately enhance the reliability and resilience of DevOps environments.

6. CASE STUDIES AND REAL-WORLD IMPLEMENTATION OF GENETIC PROGRAMMING ALGORITHMS IN DEVOPS ERROR HANDLING

Examining real-world implementations and case studies of Genetic Programming (GP) algorithms in DevOps error handling provides valuable insights into their practical applications, challenges encountered, and lessons learned. Below are two illustrative examples showcasing the deployment of GP algorithms in DevOps environments for error handling:

Automated Incident Response System at Tech Company X: Background: Tech Company X operates a large-scale cloud infrastructure supporting various web services and applications. The company faced challenges in efficiently managing and responding to incidents, leading to prolonged downtime and customer dissatisfaction.

Implementation: Company X implemented a GP-based automated incident response system to enhance error handling and recovery capabilities. The system utilized GP algorithms to analyze system logs, identify error patterns, and recommend appropriate remediation actions. Through iterative refinement and training on historical incident data, the GP algorithm learned to anticipate and respond to common error scenarios effectively.

Results: The automated incident response system significantly improved the company's ability to detect, diagnose, and recover from incidents in real-time. By automating error handling processes, Company X reduced incident response times, minimized service disruptions, and enhanced overall system reliability. The GP algorithm continuously evolved and adapted to new error patterns, ensuring that the system remained resilient to emerging threats and operational challenges.

Self-Healing Microservices Architecture at E-commerce Platform Y: Background: E-commerce Platform Y relies on a microservices architecture to deliver its online shopping experience to customers. The platform experienced frequent service disruptions and outages due to errors in individual microservices, leading to lost revenue and customer dissatisfaction.

Implementation: Platform Y implemented a self-healing microservices architecture powered by GP algorithms to automate error recovery and resilience. Each microservice was equipped with a GP-based error handler responsible for monitoring service health, detecting anomalies, and initiating recovery actions. The GP algorithm continuously learned from service telemetry data and user feedback to optimize error recovery strategies dynamically.

Results: The self-healing microservices architecture transformed Platform Y's reliability and resilience, significantly reducing downtime and improving service availability. By leveraging GP algorithms for error handling, the platform achieved rapid incident response times, seamless failover capabilities, and enhanced fault tolerance across its distributed system. The GP-based error handlers adapted to evolving error patterns and system conditions, ensuring consistent performance and minimizing the impact of errors on the customer experience.

These case studies demonstrate the practical utility and effectiveness of GP algorithms in DevOps error handling, enabling organizations to automate incident response, enhance system resilience, and maintain high levels of service availability in dynamic and complex operational environments. However, challenges such as algorithm scalability, interpretability, and integration with existing toolchains must be carefully addressed to realize the full potential of GP-based approaches in real-world DevOps deployments.

7. FUTURE DIRECTIONS AND CHALLENGES IN HARNESSING GENETIC PROGRAMMING ALGORITHMS FOR DEVOPS AUTOMATION

The utilization of Genetic Programming (GP) algorithms in DevOps automation presents promising avenues for enhancing system reliability, efficiency, and resilience. However, as this field continues to evolve, several future directions and challenges emerge:

i. **Enhanced Adaptability and Learning**: Future advancements will focus on improving the adaptability and learning capabilities of GP algorithms. This involves enabling algorithms to dynamically adjust error handling strategies based on real-time data and evolving system conditions. Challenges include designing algorithms that can effectively incorporate feedback loops, adapt to changing environments, and learn from past experiences to continuously optimize performance.

ii. **Integration with AI and Machine Learning Techniques**: There is growing interest in integrating GP algorithms with other AI and machine learning techniques to create more robust and intelligent DevOps automation solutions. Future directions involve exploring synergies between GP and techniques such as reinforcement learning, deep learning, and natural language processing to address complex error handling scenarios and improve decision-making capabilities.

iii. **Explainability and Transparency**: As GP algorithms become more sophisticated, ensuring transparency and explainability of their decision-making processes becomes imperative. Future research will focus on developing techniques to interpret and explain the solutions generated by GP algorithms, enabling stakeholders to understand the rationale behind automated error handling decisions. Addressing this challenge will foster trust, facilitate collaboration, and promote adoption of GP-based automation solutions in DevOps environments.

iv. **Scalability and Efficiency**: Scaling GP algorithms to handle large-scale DevOps environments with numerous components and interactions presents significant challenges. Future directions include exploring techniques for distributed computing, parallelization, and optimization to improve the scalability and efficiency of GP-based automation solutions while maintaining high levels of accuracy and reliability. Additionally,

addressing the computational complexity of GP algorithms will be essential for ensuring efficient execution in resource-constrained environments.

v. **Ethical and Societal Implications**: The increasing reliance on GP algorithms for DevOps automation raises ethical and societal concerns related to algorithmic bias, privacy, and fairness. Future research will focus on developing ethical guidelines, regulatory frameworks, and governance mechanisms to address these concerns and ensure responsible and ethical deployment of GP-based automation solutions. Collaboration between stakeholders, including researchers, practitioners, policymakers, and advocacy groups, will be crucial for addressing these challenges and promoting ethical DevOps automation practices.

vi. **Industry Adoption and Standardization**: Encouraging broader adoption of GP algorithms in DevOps requires establishing industry standards, best practices, and certification processes. Future directions involve promoting knowledge sharing, fostering collaboration between academia and industry, and developing standardized tools and methodologies for GP-based automation. Additionally, creating educational resources and training programs will be essential for empowering practitioners with the skills and knowledge needed to leverage GP algorithms effectively in DevOps environments.

By addressing these future directions and challenges, the utilization of GP algorithms in DevOps automation can unlock significant benefits, including improved system reliability, reduced downtime, and enhanced operational efficiency. Collaboration between researchers, practitioners, and stakeholders will be essential for driving advancements in GP-based automation and realizing its full potential in transforming DevOps practices.

8. CONCLUSION

This paper has explored the utilization of Genetic Programming (GP) algorithms to address the intricacies of error handling and recovery within DevOps environments. Our primary focus has been on tackling the pressing need for automated, adaptive, and efficient error management strategies to uphold system reliability, minimize downtime, and enhance operational efficiency in the ever-evolving DevOps landscape.

Throughout our exploration, we emphasized the potential of GP algorithms in automating error detection, diagnosis, and recovery processes by leveraging evolutionary computation principles. By iteratively evolving solutions based on historical data, real-time feedback, and domain expertise, GP algorithms offer a promising avenue for designing adaptive error management systems capable of swift response to changing conditions and emerging threats.

Our analysis underscored the critical importance of selecting suitable evaluation metrics, crafting effective fitness functions, and seamlessly integrating GP algorithms with existing DevOps toolchains to fully realize their potential in practice. Moreover, we outlined future directions and challenges in harnessing GP algorithms for DevOps automation, including enhancing adaptability, addressing scalability concerns, ensuring transparency, and navigating ethical considerations.

References

[1] F. J. Meng, M. N. Wegman, J. M. Xu, X. Zhang, P. Chen and G. Chafle, "IT troubleshooting with drift analysis in the DevOps era," in IBM Journal of Research and Development, vol. 61, no. 1, pp. 6:62-6:73, 1 Jan.-Feb. 2017, doi: 10.1147/JRD.2016.2630478.

[2] Benjamin, J., & Mathew, J. (2021). Enhancing the efficiency of continuous integration environment in DevOps. IOP Conference Series: Materials Science and Engineering, 1085, 012025. doi:10.1088/1757-899X/1085/1/012025

[3] Taryana, A., Fadli, A., Murdyantoro, E., & Nurshiami, S. R. (2020). DevOps Approach Embraces Forward and Reverse Engineering. International Journal of Applied Information Technology, 4(2), 103-116.

[4] Annadata, L. A. (2023). A Data-Driven Approach for Incident Handling in DevOps (Dissertation). Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:bth-25492

[5] R. DILLIBABU, SANGEETHA. A and L. SUDHA, 2014. DEVELOPMENT AND APPLICATION OF SFMEA MODEL TO SOFTWARE TESTING ENVIRONMENT. International Journal of Industrial Engineering Research and Development (IJIERD). Volume:4, Issue:3,Pages:61-72. doi: https://doi.org/10.34218/IJIERD.4.3.2013.005

[6] Kwansoon Park and Boyoung Kim, Core Container Security Frameworks, International Journal of Advanced Research in Engineering and Technology (IJARET), 11(6), 2020, pp. 1024-1038. doi: 10.34218/IJARET.11.6.2020.092

[7] Ali N. Ahmed, Gasim Hayder, Raihana Aliya Binti Abdul Rahman and Abdoulhdi A. Borhana, Rainfall-Runoff Forecasting Utilizing Genetic Programming Technique, International Journal of Civil Engineering and Technology,10(01),2019,pp.1523–1534

[8] Ramachandran, K. K. "Optimizing it performance: a comprehensive analysis of resource efficiency." International Journal of Marketing and Human Resource Management (IJMHRM), 14(3), 2023, pp. 12-29.

[9] Zhou L, Ye X, Huang Z, Xie P, Song Z, Tong Y. An Improved Genetic Algorithm for the Recovery System of USVs Based on Stern Ramp Considering the Influence of Currents. Sensors. 2023; 23(19):8075. https://doi.org/10.3390/s23198075

[10] Macedo, J., Costa, E., Marques, L. (2016). Genetic Programming Algorithms for Dynamic Environments. In: Squillero, G., Burelli, P. (eds) Applications of Evolutionary Computation. EvoApplications 2016. Lecture Notes in Computer Science(), vol 9598. Springer, Cham. https://doi.org/10.1007/978-3-319-31153-1_19

[11] Srikanth, K. (2022). DevOps for Cloud Computing: An Overview. International Journal of Engineering Applied Sciences and Technology, 6(10), 195-201. ISSN No. 2455-2143.

[12] Nivedhaa N, From Raw Data to Actionable Insights: A Holistic Survey of Data Science Processes, International Journal of Data Science (IJDS), vol. 1, issue 1, pp. 1-16, 2024.

[13] Ananth Raja Muthukalyani, Leveraging Data Science Techniques for Customer Segmentation and Targeted Marketing in the Retail Industry, International Journal of Data Analytics Research and Development (IJDARD), 1(1), 2023, pp. 42-50.

[14] Ramachandran, K. K. "Digital Dynamics: Integrating Online Channels for Sales Promotion Success." International Journal of Advanced Research in Management (IJARM), 14(2), 2023, pp. 20-35.

[15] S. B. Vinay, A Study on Application of Artificial Intelligence in E-Recruitment in IT Sector, Chennai, International Journal of Marketing and Human Resource Management (IJMHRM), 14(1), 2023, pp. 1-14. doi: https://doi.org/10.17605/OSF.IO/H8D3P

[16] S. B. Vinay, Transforming E-Governance with Artificial Intelligence: Opportunities, Challenges, and Future Directions, International Journal of Advanced Research in Management (IJARM). 14(1), 2023. pp. 1-10. doi: https://doi.org/10.17605/OSF.IO/UZJ3P

[17] R.Sharmila and N.Kannan, A Comprehensive Survey of Cyber Security Specific to Cyber Defence and Digital Forensics, International Journal of Computer Engineering and Technology (IJCET), 14(2), 2023, pp. 61-72 doi: https://doi.org/10.17605/OSF.IO/H9DBX

[18] Sabarimani K, Prathibha K, Divyasre T, Balasubramani P, Tamilselvan P, Prakash N, Tamilselvan S and Gowrishankar R, Despeckling and Resolution Enhancement of SAR Images using Wavelet Transform, International Journal of Electronics and Communication Engineering and Technology, 13(2), 2022, pp. 9-23 doi: https://doi.org/10.17605/OSF.IO/FUMJR

Citation: Srividhya S, Genetic Programming for Automated Error Handling and Recovery in DevOps Environments. International Journal of DevOps (IJDO), 1(1), 2024, 1-10.

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJDO/VOLUME_1_ISSUE_1/IJDO_01_01_001.pdf

Abstract Link:
https://iaeme.com/Home/article_id/IJDO_01_01_001

✉ **editor@iaeme.com**