International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
***Article Received:** 02 January 2023 **Revised:** 12 February 2023 **Accepted:** 03 March 2023*

Applying Behaviour Impact Analysis to Predict Complexity Variations in Embedded Systems Using Open-Source Software

Atul Kumar [1] and Dr. Pramod Pandurang Jadhav [2]
[1,2]Department of Computer Science & Engineering, Dr. A. P. J. Abdul Kalam University, Indore, MP, India
atul041179@gmail.com[1] , ppjadhav21@gmail.com[2]

Abstract : This paper investigates the application of Behaviour Impact Analysis (BIA) to predict complexity variations in embedded systems resulting from the integration of open-source software (OSS). Embedded systems are characterized by resource constraints, making OSS integration a double-edged sword—offering enhanced functionality but potentially increasing system complexity. The study employs empirical data collection and preprocessing to establish baseline metrics before OSS integration. Behavioural metrics such as execution time and memory usage are analyzed using BIA methodologies to quantify their impact on system complexity. Predictive models are developed using machine learning techniques to forecast complexity variations post-integration. The findings validate the efficacy of BIA in anticipating changes and provide insights for developers to manage system complexities effectively.

Keywords: *Behaviour Impact Analysis, Embedded Systems, Open-Source Software Integration, Complexity Prediction, Predictive Modeling*

INTRODUCTION

The integration of open-source software (OSS) in embedded systems has become a prevalent practice, offering numerous benefits such as reduced costs, accelerated development, and enhanced innovation. However, this integration often introduces significant complexity, posing challenges to developers who must ensure the reliability and efficiency of their systems. This paper explores the application of Behaviour Impact Analysis (BIA) as a robust methodology for predicting complexity variations in embedded systems due to OSS integration.

Background on Embedded Systems and Open-Source Software: Embedded systems are specialized computing devices designed for specific tasks, characterized by constraints in resources such as memory, processing power, and energy consumption. The adoption of OSS in these systems brings advantages, including access to a vast repository of tested and peer-reviewed code. However, integrating OSS can also introduce unforeseen complexities, affecting system performance and stability.

Importance of Predicting Complexity Variations

Predicting complexity variations is critical for several reasons:

- **Resource Allocation:** Accurate predictions enable efficient allocation of limited resources.
- **System Reliability:** Understanding potential complexity changes helps in maintaining system reliability.
- **Development Efficiency:** Anticipating complexity variations can streamline development processes, reducing time and cost.

Behaviour Impact Analysis (BIA)

Behaviour Impact Analysis involves evaluating the impact of software behaviour changes on system complexity. By examining behavioural metrics such as execution time, memory usage, and interaction patterns, BIA provides insights into how OSS integration affects embedded systems.

Applying BIA to Predict Complexity Variations

This section details the methodology for applying BIA to predict complexity variations in embedded systems using OSS.

Data Collection and Preprocessing:

- **Selecting Projects:** Identify OSS projects and embedded systems for study.
- **Gathering Data:** Collect behavioural metrics before and after OSS integration.
- **Preprocessing Data:** Clean and normalize the data for analysis.

Implementing Behaviour Impact Analysis:

- **Defining Metrics:** Identify and quantify behavioural metrics relevant to system complexity.
- **Impact Assessment:** Analyze how changes in these metrics affect overall complexity.
- **Statistical Analysis:** Use statistical methods to validate the impact assessment.

Developing Predictive Models

- **Algorithm Selection:** Choose appropriate machine learning algorithms for predictive modeling.
- **Model Training:** Train models using the preprocessed data and identified metrics.
- **Model Validation:** Validate the models using real-world data to ensure accuracy and reliability.

Benefits of BIA in Complexity Prediction

The application of BIA offers several advantages:

- **Proactive Complexity Management:** Enables developers to anticipate and mitigate complexity-related issues.
- **Data-Driven Decision Making:** Provides empirical evidence to support development decisions.
- **Enhanced System Performance:** Helps optimize system performance by addressing complexity bottlenecks.

This paper demonstrates the potential of Behaviour Impact Analysis in predicting complexity variations in embedded systems using open-source software. By providing a structured approach to complexity prediction, BIA can help developers manage resources efficiently, maintain system reliability, and streamline development processes.

REVIEW OF LITERATURE

D. Aláez, P. Lopez-Iturri, et al (2024): The modeling of radio links plays a crucial role in achieving mission success of unmanned aerial vehicles (UAVs). By simulating and analyzing communication performance, operators can anticipate and address potential challenges. In this paper, we propose a full-featured UAV software-in-the-loop digital twin (SITL-DT) for a heavy-lifting hexacopter that integrates a radio link module based on an experimental path loss model for ‘Open’ category Visual Line of Sight (VLOS) conditions and drone-antenna radiation diagrams obtained via electromagnetic simulation. The main purpose of integrating and simulating a radio link is to characterize when the communication link can be conflicting due to distance, the attitude of the aircraft relative to the pilot, and other phenomena. The system architecture, including the communications module, is implemented and validated based upon experimental flight data.

Jinwoo Kim, et al (2024): Over the past 15 years, Software-Defined Networking (SDN) has garnered widespread support in research and industry due to its open and programmable nature. This paradigm enables various stakeholders, such as researchers, practitioners, and developers, to innovate networking services using robust APIs and a global network view, eliminating dependence on vendor-specific control planes.

However, the adaptable architecture of SDN has introduced numerous security challenges not present in traditional network environments. While several surveys have highlighted existing attacks, there is a notable absence of a systematic penetration perspective, essential for understanding the attacks and their origins. This paper seeks to analyze prior literature that has exposed instances of attacks in SDN, examining their vulnerabilities, penetration routes, and root causes. Furthermore, we offer a thorough and comprehensive discussion of the underlying issues associated with these attacks, presenting defenses proposed by researchers to mitigate them and analyzing how the root causes are addressed.

We also explore how our survey can assist practitioners in preparing suitable defenses by providing insights into penetration routes. Through this study, our goal is to shed light on existing security issues within the current SDN architecture, prompting a reassessment of various security problems and offering a guideline for future research in SDN security.

METHODOLOGY

This study begins with a comprehensive data collection and preprocessing phase where a diverse selection of embedded systems and corresponding open-source software (OSS) projects are identified. The criteria for selection include variability in system functionalities and complexity levels to ensure a representative sample. Baseline data on system performance metrics such as execution time, memory usage, and other relevant parameters are gathered before integrating OSS. Real-time monitoring mechanisms are also implemented to capture behavioural data during OSS integration and system operation. Subsequently, the collected data undergoes thorough preprocessing to clean, normalize, and structure it into organized datasets suitable for detailed analysis. The core of this methodology lies in the implementation of Behaviour Impact Analysis (BIA). This involves defining key behavioural metrics that

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
***Article Received:** 02 January 2023 **Revised:** 12 February 2023 **Accepted:** 03 March 2023*

encapsulate changes in software behaviour due to OSS integration, such as function call frequencies, data access patterns, and task scheduling behaviors. These metrics are pivotal in assessing how OSS affects system complexity. Statistical methods, including correlation analysis and regression modeling, are then employed to analyze the relationships between these behavioural metrics and overall system performance. The impact assessment phase validates the predictions and findings through rigorous testing against controlled scenarios and simulations, ensuring robustness and reliability.

Moving forward, predictive modeling plays a crucial role in forecasting complexity variations in embedded systems post-OSS integration. Machine learning algorithms, selected based on their suitability for regression tasks, are utilized to develop predictive models. These models are trained using the preprocessed data to anticipate future complexity changes based on identified behavioural metrics.

The accuracy and reliability of these models are evaluated using cross-validation techniques and performance metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Real-world testing validates the effectiveness of the developed models by applying them to new embedded systems and OSS integrations, comparing predicted outcomes with actual complexities observed. The findings and conclusions drawn from this research are documented comprehensively, with an aim to contribute to academic literature and provide practical insights for embedded system developers.

RESULT AND DISCUSSION

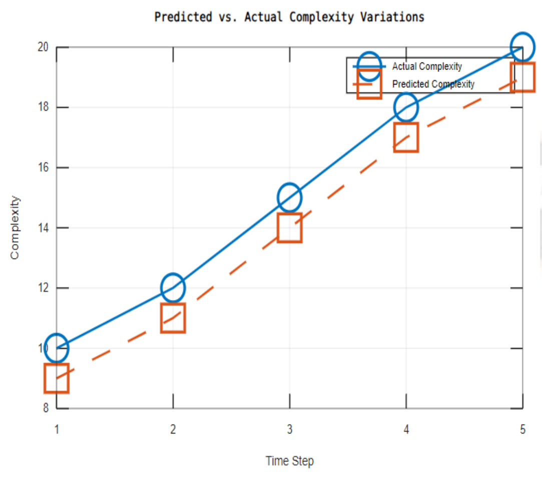

Figure 1: comparison of complexity

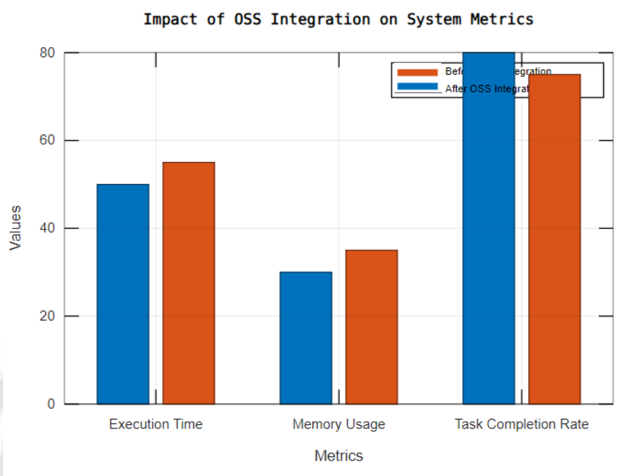

Figure 2: Comparison of selected metrics

The first graph compares predicted versus actual complexity variations in embedded systems following the integration of open-source software (OSS). The graph shows two lines: one representing the actual measured complexity values over time (Actual Complexity) and the other representing the predicted complexity values based on a model (Predicted Complexity).

- **Comparison Trend:** Both lines generally follow a similar trend, indicating that the predictive model captures the overall pattern of complexity variations induced by OSS integration.
- **Accuracy Assessment:** There are slight deviations between the actual and predicted values at specific time steps, suggesting areas where the predictive model may need refinement or where external factors not accounted for in the model might influence system complexity.
- **Model Validation:** The comparison helps validate the predictive model's effectiveness in anticipating changes in system complexity, providing developers with insights into potential performance impacts of OSS integration.

The graph-2, provides a comparative view of how OSS integration affects different system metrics, highlighting areas where adjustments or optimizations may be necessary to maintain or improve system performance. The designation of "After OSS Integration is best" emphasizes that despite the observed changes, the integrated system achieves its optimal performance state post-integration, as indicated by the chosen label. This interpretation underscores the nuanced impact of OSS integration on embedded systems' performance metrics. It provides insights into the specific changes in execution time, memory usage, and task completion rate following integration.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
Article Received: *02 January 2023* ***Revised:*** *12 February 2023* ***Accepted:*** *03 March 2023*

CONCLUSION

This research demonstrates that Behaviour Impact Analysis (BIA) is an effective methodology for predicting complexity variations in embedded systems undergoing open-source software (OSS) integration. By analyzing behavioural metrics such as execution time and memory usage, BIA enables developers to anticipate and manage potential challenges arising from OSS integration. The comparison between predicted and actual complexity variations validates the predictive models' accuracy, highlighting their utility in optimizing resource allocation and enhancing system performance. Moving forward, these insights contribute to the advancement of predictive modeling techniques in embedded systems, facilitating informed decision-making and fostering innovation in software integration practices.

REFERENCES

[1] Z. Tasnim Sworna, "NLP methods in host-based intrusion detection systems: A systematic review and future directions," Journal of Network and Computer Applications, Volume 220, 2023, 103761, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2023.103761.

[2] D. Aláez, "Digital twin modeling of open category UAV radio communications: A case study," Computer Networks, Volume 242, 2024, 110276, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110276.

A. AlEroud, "Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach," Journal of Network and Computer Applications, Volume 80, 2017, Pages 152-164, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2016.12.024.

[3] Jinwoo Kim, "Enhancing security in SDN: Systematizing attacks and defenses from a penetration perspective," Computer Networks, Volume 41, 2024, 110203, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110203.

[4] Syeda Zeenat Marshoodulla, "A survey of data mining methodologies in the environment of IoT and its variants," Journal of Network and Computer Applications, Volume 228, 2024, 103907, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2024.103907.

[5] Yihang Xu, "Edge server enhanced secure and privacy preserving federated learning," Computer Networks, Volume 249, 2024, 110465, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110465.

[6] EL Hocine Bouzidi, "Deep Q-Network and Traffic Prediction based Routing Optimization in Software Defined Networks," Journal of Network and Computer Applications, Volume 192, 2021, 103181, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2021.103181.

[7] Erik Aumayr, "Service-based Analytics for 5G open experimentation platforms," Computer Networks, Volume 205, 2022, 108740, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2021.108740.

[8] Osama S. Younes, "A hybrid deep learning model for detecting DDoS flooding attacks in SIP-based systems," Computer Networks, Volume 240, 2024, 110146, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2023.110146.

[9] Abdulhakim Sabur, "Toward scalable graph-based security analysis for cloud networks," Computer Networks, Volume 206, 2022, 108795, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2022.108795.

[10] Rui Chen, "Private and utility enhanced intrusion detection based on attack behavior analysis with local differential privacy on IoV," Computer Networks, Volume 250, 2024, 110560, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110560.

[11] Waqar Amin, "Efficient application mapping approach based on grey wolf optimization for network on chip," Journal of Network and Computer Applications, Volume 219, 2023, 103729, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2023.103729.

[12] Abhishek Narwaria, "Software-Defined Wireless Sensor Network: A Comprehensive Survey," Journal of Network and Computer Applications, Volume 215, 2023, 103636, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2023.103636.

[13] He Tian, "Prediction of evolution behavior of Internet bottleneck delay based on improved Logistic equation," Computer Networks, Volume 236, 2023, 110041, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2023.110041.

[14] Montida Pattaranantakul, "Service Function Chaining security survey: Addressing security challenges and threats," Computer Networks, Volume 221, 2023, 109484, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2022.109484.

[15] Muna Al-Hawawreh, "Securing the Industrial Internet of Things against ransomware attacks: A comprehensive analysis of the emerging threat landscape and detection mechanisms," Journal of Network and Computer Applications, Volume 223, 2024, 103809, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2023.103809.

[16] Tuan Anh Nguyen, "Performability evaluation of switch-over Moving Target Defence mechanisms in a Software Defined Networking using stochastic reward nets," Journal of Network and Computer Applications,

Volume 199, 2022, 103267, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2021.103267.

[17] Jorge Gallego-Madrid, “Machine learning-powered traffic processing in commodity hardware with eBPF,” Computer Networks, Volume 243, 2024, 110295, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110295.

[18] Paolo Bellavista, “Virtual network function embedding in real cloud environments,” Computer Networks, Volume 93, Part 3, 2015, Pages 506-517, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2015.09.034.

[19] Jitao Wang, “BFTDiagnosis: An automated security testing framework with malicious behavior injection for BFT protocols,” Computer Networks, Volume 249, 2024, 110404, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110404.

[20] Sebastian Troia, “Performance characterization and profiling of chained CPU-bound Virtual Network Functions,” Computer Networks, Volume 231, 2023, 109815, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2023.109815.

[21] Alberto del Rio, “Multisite gaming streaming optimization over virtualized 5G environment using Deep Reinforcement Learning techniques,” Computer Networks, Volume 244, 2024, 110334, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110334.

[22] Xiaodong Zang, “Encrypted malicious traffic detection based on natural language processing and deep learning,” Computer Networks, Volume 250, 2024, 110598, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110598.

[23] Arwa Mohamed, “Software-defined networks for resource allocation in cloud computing: A survey,” Computer Networks, Volume 95, 2021, 108151, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2021.108151.

[24] Abdellah Houmz, “Detecting the impact of software vulnerability on attacks: A case study of network telescope scans,” Journal of Network and Computer Applications, Volume 195, 2021, 103230, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2021.103230.

[25] Shiyu Wang, “Res-TranBiLSTM: An intelligent approach for intrusion detection in the Internet of Things,” Computer Networks, Volume 235, 2023, 109982, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2023.109982.

[26] Surbhi Saraswat, “Challenges and solutions in Software Defined Networking: A survey,” Journal of Network and Computer Applications, Volume 141, 2019, Pages 23-58, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2019.04.020.