Haptic Feedback Integration for Surgical Training Simulation

Siddharth Jha, Zichen Gui, Benjamin Delbos, Richard Moreau, Arnaud Lelevé, Irene Cheng

Haptic Feedback Integration for Surgical Training Simulation

Siddharth Jha
Department of Computing Science
University of Alberta
Edmonton, Canada
sjha4@ualberta.ca

Zichen Gui
Department of Computing Science
University of Alberta
Edmonton, Canada
zgui@ualberta.ca

Benjamin Delbos
Department of Electrical Engineering
Insa, Lyon
Lyon, France
benjamin.delbos@insa-lyon.fr

Richard Moreau
Department of Electrical Engineering
Insa, Lyon
Lyon, France
richard.moreau@insa-lyon.fr

Arnaud Lelevé
Département of Electrical Engineering
Insa, Lyon
Lyon, France
arnaud.leleve@insa-lyon.fr

Irene Cheng
Department of Computing Science
University of Alberta
Edmonton, Canada
Iocheng@ualberta.ca

***Abstract*—This paper introduces an innovative approach to integrate haptic feedback in surgical training simulations, utilizing a combination of Python and C++ with supporting technologies to enhance interactivity and realism. By overcoming limitations associated with traditional haptic feedback systems, such as local processing constraints, our methodology enables the seamless transmission of haptic cues over the internet, significantly improving accessibility and distribution. This research aims to demonstrate the potential of our system to enhance user experience in medical education and beyond, setting a new standard for immersive virtual environments.**

***Index Terms*—haptic feedback, surgical simulation, python, redis, websocket, Chai3d, front-end and back-end integration, software engineering, remote control, signal transmission**

I. Introduction

Haptic feedback technology enhances virtual and augmented reality environments by providing immersive physical sensations that mimic real-world interactions [1]. However, the integration of haptic feedback faces significant challenges, including complex development processes, high latency, limited compatibility with various platforms, and reliance on local processing. These issues restrict the scalability and accessibility of haptic feedback systems, particularly in remote or distributed settings [2].

Addressing these challenges [3], our research proposes a novel integration framework leveraging Python's versatility and C++'s performance efficiency. Redis is utilized for its data synchronization capabilities, ensuring accurate and timely haptic feedback. WebSocket technology enables real-time, bidirectional communication between the frontend and backend, reducing latency and improving user experience. Furthermore, we employ Chai3D for its advanced haptic rendering capabilities, enabling more realistic and immersive simulations. Our novel approach not only overcomes the limitations of existing systems but also advances the deployment of haptic feedback in virtual training environments, paving the way for broader applications (eg., robotic surgery) and enhanced user experiences.

II. Challenges and Background in Haptic Feedback Integration

The integration of haptic feedback into virtual environments poses a multifaceted challenge, shaped by its historical evolution and contemporary technical hurdles. The pioneering work by Dorin Mircea Popovici et al. [4] not only outlines the transition from hardware-centric designs to software-driven solutions but also emphasizes the role of software in enhancing scalability and accessibility. This paper highlights how software advancements have enabled more sophisticated simulations, allowing for an enriched user presence through tactile, visual, and auditory feedback, thus broadening the application spectrum to include critical fields such as medicine, engineering, and education.

Felix G. Hamza-Lup et al. [5] delve into the pedagogical implications of haptic feedback, illustrating its potential to revolutionize distance learning. By integrating tactile interactions within online learning platforms, they demonstrate how haptic feedback can significantly enhance cognitive and social presence. Their method focuses on creating interactive haptic environments that engage students more deeply with the learning material, thereby facilitating a more immersive educational experience.

The research into large-scale kinesthetic haptic integration presented by Kunzler et al. [6] emphasizes the technical complexity of creating realistic tactile sensations while ensuring low-latency communication

in distributed systems. They propose a Haptics-Server architecture designed to address these challenges. This architecture enables scalable, immersive haptic experiences by allowing for efficient communication and interaction between various haptic devices and virtual environments, thus overcoming interoperability issues and reducing feedback latency.

Our research builds upon these foundational works, introducing a novel integration framework that leverages the strengths of Python and C++ for performance efficiency and flexibility. By utilizing Redis for data synchronization and WebSocket technology for real-time, bidirectional communication, our approach addresses the critical challenge of reducing latency. Furthermore, the employment of Chai3D for its advanced haptic rendering capabilities enables the creation of more realistic and immersive simulations. This comprehensive approach not only addresses the limitations highlighted by the previous studies but also expands the potential applications of haptic feedback, enhancing user experiences in virtual training environments and beyond.

By drawing on the insights provided by these studies and integrating cutting-edge technologies, our framework aims to overcome the challenges of haptic feedback integration, paving the way for more seamless, immersive, and accessible virtual experiences.

III. Technical Framework and Tools for Haptic Feedback Integration

The development and integration of haptic feedback systems within virtual environments necessitate a multifaceted approach, leveraging a combination of advanced programming languages, data management tools, communication protocols, and haptic rendering technologies. This section delves into the fundamental concepts and technologies underpinning our research, including the utilization of Python and C++ for crafting the frontend and backend of our system, Redis for efficient data synchronization [7], WebSocket for real-time communication [8], and Chai3D for immersive haptic rendering. By outlining the architecture, integration process, implementation strategies, and evaluation methodologies, we aim to provide a comprehensive overview of the innovative solutions deployed to enhance the fidelity and responsiveness of haptic feedback in surgical training simulations.

A. Fundamental Concepts of Python Frontend and C++ Backend

Python is a high-level programming language that is widely used for web development, data analysis, and scripting tasks. It is known for its simplicity, readability, and versatility. Python provides a user-friendly and intuitive interface for developers, making it an ideal choice for frontend development. It also enables the use of Machine learning libraries and frameworks, allowing for advanced functionalities in haptic feedback systems. On the other hand, C++ is a widely used programming language known for its efficiency and performance. It is commonly used for developing backend systems that require low-level control and application optimization.

B. The Role of Redis in Haptic Feedback Integration

Redis, a popular in-memory data structure store, plays a crucial role in the haptic feedback integration process. It serves as a central communication hub between the Python frontend and the C++ backend, allowing for seamless data exchange and synchronization. It facilitates real-time communication and data sharing between the frontend and backend systems, enabling efficient haptic feedback generation and control [9].

C. Exploring WebSocket for Data Transmission

WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection. It allows for real-time, bidirectional communication between the frontend and backend systems, making it an ideal choice for haptic feedback integration. By utilizing WebSocket, the Python frontend can send haptic commands and receive real-time feedback from the C++ backend. This enables a seamless and responsive haptic feedback experience for the user [10].

D. Understanding Chai3D and Its Applications

Chai3D is a powerful haptic framework written in C++, designed for the development of interactive and immersive haptic applications. It provides a wide range of functionalities for haptic rendering and interaction, including the ability to render haptic forces and attach virtual objects to haptic device for realistic haptic feedback [11].

IV. Our Novel Approach to Connect Haptic Feedback to Python Frontend and C++ Backend

Our novel approach involves integrating Redis, WebSocket, Chai3D, Python and Django [12] to create a seamless integration between the haptic feedback system and the frontend-backend architecture. This approach allows for efficient data transmission and synchronization between the Python frontend and C++ backend. The Django frontend utilizes WebSocket to establish a real-time connection with the C++ backend, enabling bidirectional communication. Django channels with Redis as a backend allows for message queuing and data synchronization, ensuring that haptic feedback commands and real-time data are exchanged accurately and efficiently. The C++ backend uses Chai3d to keep track of positions and movements of the haptic device, and to generate appropriate force feedback based on

user interactions and predefined algorithms. It then sends the serialized haptic feedback data through the WebSocket connection to the Python frontend. The Python frontend receives the haptic feedback data and uses it to update the visualization. By leveraging the capabilities of Redis, WebSocket, Chai3D, Python, and Django, our approach provides a robust and efficient solution for connecting haptic feedback to a Python frontend and C++ backend. This integrated system allows for seamless and real-time bidirectional communication between the haptic device, C++ backend, and Python frontend.

a) Discussion on WebSocket and Chai3D Synergy: The integration of WebSocket and Chai3D in our approach provides a powerful synergy for real-time haptic feedback. WebSocket is a communication protocol that enables bidirectional communication between a client and a server over a single, long-lived connection. It allows for efficient and low-latency data transmission, making it ideal for real-time applications such as haptic feedback. Chai3D, on the other hand, is a haptic rendering and interaction framework that provides the ability to render haptic forces and attach virtual objects to physical haptic devices. By combining WebSocket with Chai3D, we can achieve seamless integration between the haptic feedback system and the frontend-backend architecture.

V. Implementation and Testing Results

Our haptic feedback system incorporates two major components: a frontend system for user interface and surgery visualization, and a backend system for haptic feedback calculation and device communication.

A. Frontend Implementation

Implementation of the frontend system, mostly consisting of MRI segmentation, reconstruction and visualization, is described in our separate paper [13]. Built upon this visualization framework, our final haptic feedback system further utilized Three.js [14] to create proper virtual representations of the real-world haptic feedback device environment. To be more exact, we have done the following steps:

a) Creation of Three.js Objects: In our haptic feedback simulation system, we utilize a sophisticated setup that includes a pivotal ball joint and a feedback-providing needle, as depicted in Fig. 1. The ball joint serves as the central pivot for the skull model, facilitating rotational movements, while the needle is designed to replicate the sensations of haptic feedback during the simulation. To accurately represent these critical components within our virtual environment, we crafted two virtual objects using Three.js. The ball joint is represented by the sphere and the needle movement is represented by a 3d line.

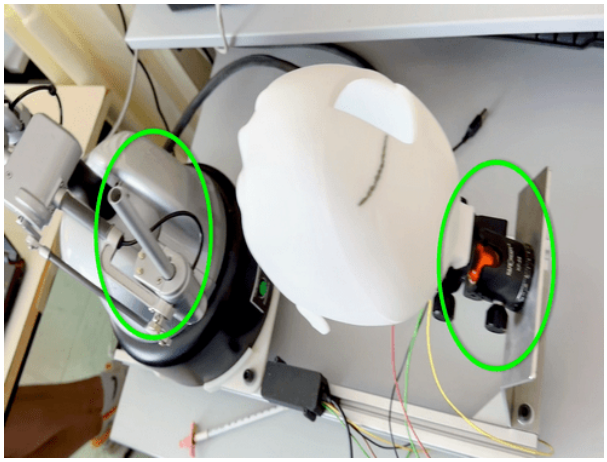

Fig. 1: Our haptic device system setup to simulate the virtual operation.

b) Virtual-Reality Alignment: In order to make sure that the virtual scene is in synchronization with reality, we need to align them in the same coordinate system. We decided to use the haptic device's coordinate system as our base, and then convert everything else (ball joint, MRI reference, etc.) to the base coordinate system. This conversion process mainly involves rotation and scaling, because these coordinate systems have different x, y, and z directions and different units of lengths. In addition, we have manually measured the position of the ball joint and the needle in the physical setup and translated them accordingly in the Three.js scene. After this step, the final Three.js scene is shown in Fig. 2.

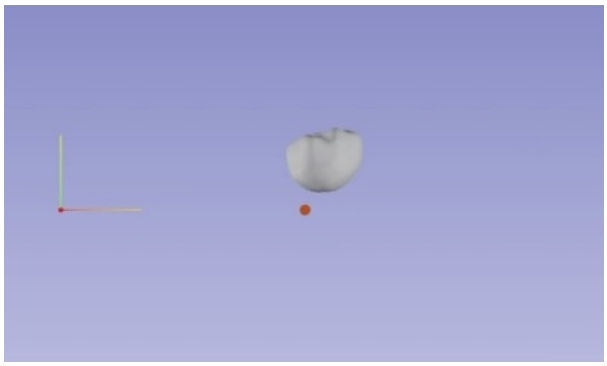

Fig. 2: Setup of the virtual Three.js scene after alignment.

c) Three.js interfaces implementation: A few interface functions have also been defined and implemented to facilitate frontend-backend communication. These interfaces are mostly related to the transforms of Three.js objects. In this way, whenever an object is moved in the real world and the device informs the backend system, it can also update the position and ro-

tation of this object in the frontend easily. Furthermore, lines will also be drawn in the virtual scene to reflect the real-world trajectory of the needle.

B. Backend Implementation

The core of our haptic feedback functionality is powered by CHAI3D, an open-source set of C++ libraries for computer haptics, visualization, and interactive real-time simulation. This haptic engine is responsible for generating force feedback and tracking the movements of the haptic device.

In addition, we established a Redis server to manage the queuing and synchronization of data between the frontend and backend systems. This integration plays a crucial role in maintaining the system's responsiveness and accuracy.

Algorithm 1 Haptic Communication Algorithm

$position \leftarrow hapticDevice.getPosition()$
$force \leftarrow calculateHapticForce(position)$
$rotation \leftarrow hapticDevice.getRotation()$
$jsonObject \leftarrow$ JSON.stringify($\{force : force, position : position, rotation : rotation\}$)
$sendWebSocketMessage(jsonObject)$
$handleWebSocketMessage()$

Algorithm 2 Send WebSocket Message Function

function SENDWEBSOCKETMESSAGE($message$)
    $WebSocket.send(message)$
end function

Algorithm 3 Handle WebSocket Message Function

function HANDLEWEBSOCKETMESSAGE
    $WebSocket.onmessage$(function(message))
        $data \leftarrow$ JSON.parse($message.data$)
        $sendToDjangoChannels(data)$
end function

Algorithm 4 Send to Django Channels Function

function SENDTODJANGOCHANNELS($message$)
    ▷ Send data to different consumers
end function

Algorithm 5 Update HTML Function (JavaScript)

function UPDATEHTML($data$)
    ▷ Update needle position using received data
    ▷ Set needle trajectory using received data
    ▷ Update skull rotation using received data
end function

Our comprehensive testing aimed to evaluate both the effectiveness and the performance of our system. These tests particularly focused on the integration of WebSocket and Chai3D with our Python-based frontend and C++ backend. The results (with more details in the next section) demonstrated that our system provides highly responsive and accurate haptic feedback, which is essential for realistic medical simulations.

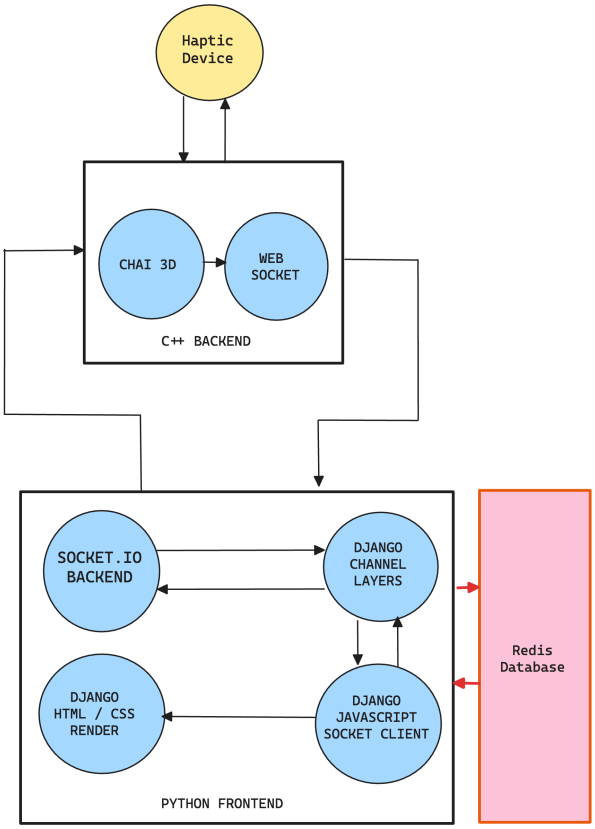

Fig. 3: Architectural Overview of our Haptic-Based Medical Simulation System

VI. Analysis of Research Outcomes

A. Haptic feedback for brain segmentation

Upon successful implementation and integration of the frontend and backend modules, our system showcases robust communication between the real-world haptic device and the virtual environment, with the haptic feedback system effectively functioning as intended. Actual needle or skull movement will update their transform correctly in the Three.js scene, as well as produce expected haptic feedback accordingly. We did some quantitative analysis to to measure the precision and responsiveness of our system, which will be discussed in the performance evaluation section.

Some screenshots of an example needle movement and the corresponding trajectory in Three.js are provided in Fig. 4, which illustrates that the virtual needle

accurately reflects real-world movement. To evaluate the haptic feedback force and ensure the virtual needle accurately reflects real-world movement, we conducted a series of tests focusing on the force feedback accuracy and responsiveness. These tests involved comparing the force applied by the haptic device in the real world against the simulated force feedback in the virtual environment. We measured the delay between actual needle movement and the corresponding haptic feedback, as well as the precision of force replication in various surgical scenarios simulated in the Three.js environment. Results from these tests, detailed in the performance metrics table, demonstrated our system's capability to process data and transmit haptic feedback with minimal latency, ensuring a realistic and immersive experience.

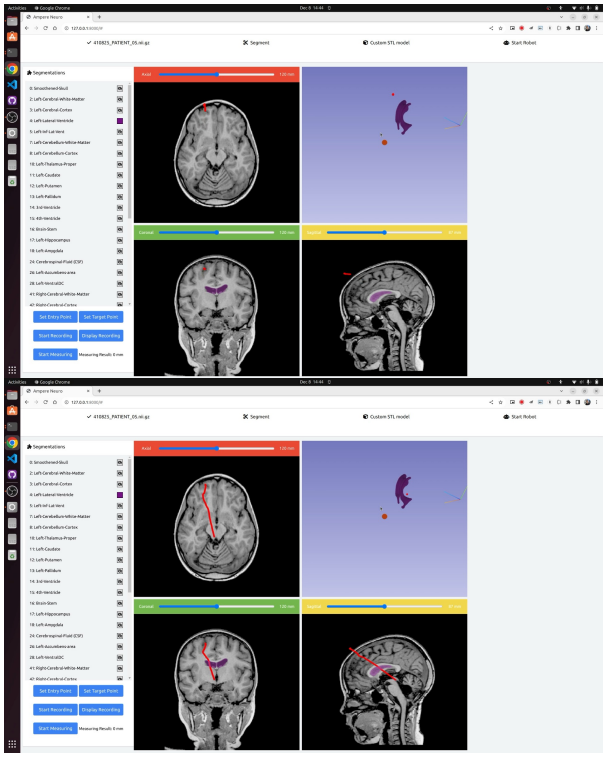

Fig. 4: An example of needle movement and trajectory in our system.

B. Performance Evaluation

We have also performed multiple rounds of performance evaluation in terms of data transmission. The results, displayed in Table I, not only validate the system's capability for real-time performance but also underscore the efficiency of our integrated haptic feedback system. These benchmarks highlight the system's potential in high-stakes environments where timely data transmission is crucial.

TABLE I: Performance Metrics of System Components.

Component	Metric	Measurement	Unit
C++ Backend	Processing Rate	15	ms/op
	Transmission Rate	5	ms/op
Redis Server	Queue Length	10	items
	Processing Latency	1	ms
Python Frontend	Retrieval Rate	10	ms/op
	Update Latency	5	ms

C. Doctors' Feedback

During a comprehensive evaluation session held at a renowned hospital in Lyon, our haptic feedback system was rigorously tested by medical professionals, including resident students and two distinguished neurosurgeons. The system, acclaimed for its integration and performance enhancements over the previous Qt C++-based implementation, was assessed for its practicality and effectiveness in a real-world clinical setting.

Doctor 1 remarked, *"The system's responsiveness and precision in simulating complex surgical procedures are outstanding. It significantly surpasses previous versions in both stability and user engagement."*

Doctor 2 provided constructive feedback, *"While the system's performance is impressive, enhancing the MRI visuals could further improve surgical planning and execution for complex procedures. The integration of machine learning for MRI segmentation has significantly revolutionized the visual clarity and detail available, making surgical planning more precise and reliable. This technological advancement in the visuals directly contributes to our ability to perform with greater accuracy."*

Both neurosurgeons and resident students found the system exceptionally useful, with students expressing a keen interest in utilizing the system for their surgical training. By leveraging Python's flexibility for the frontend, we could seamlessly integrate cutting-edge machine learning models for MRI segmentation, enhancing the visual fidelity and surgical planning accuracy. This architectural decision facilitated a more efficient development process, allowing for rapid iteration and integration of advanced algorithms, directly contributing to the improvements observed and praised by Doctor 2. This feedback highlights the system's potential to become an indispensable tool in surgical education and practice.

VII. Future Prospects for Haptic Feedback Technology Development

The successful implementation and testing of our approach demonstrate the feasibility and potential of integrating haptic feedback with a Python frontend and C++ backend. This novel approach opens up avenues for various applications, particularly in the field of virtual reality, medical simulations, and gaming. By leveraging the power of Django, WebSocket, Redis, and Chai3D,

we were able to create a responsive and accurate haptic feedback system that can enhance user experiences in these domains.

Future work could explore optimizing the haptic feedback algorithms for even greater responsiveness and accuracy, as well as expanding the system's compatibility with a wider range of haptic devices and applications. Additionally, investigating the integration of machine learning techniques to predict and adapt haptic responses in real-time could further enhance the system's capabilities and user experience.

VIII. Acknowledgement

The funding support from MITACS is gratefully appreciated.

References

[1] C. Rossa et al. "Improving Realism Through Effective and Transparent Actuation in Haptic Simulation". In: *Frontiers in Robotics and AI* 7 (2014), p. 25. DOI: 10.3389/frobt.2014.00025. URL: https://www.frontiersin.org/articles/10.3389/frobt.2020.00025/full.

[2] Aaron Raymond See, Jose Antonio G. Choco, and Kohila Chandramohan. "Touch, Texture and Haptic Feedback: A Review on How We Feel the World around Us". In: *Applied Sciences* (2022), p. 4686. DOI: 10.3390/app12094686. URL: https://www.mdpi.com/2076-3417/12/9/4686.

[3] Dangxiao Wang et al. "Haptic display for virtual reality: progress and challenges". In: (2022). URL: https://www.sciencedirect.com/science/article/pii/S2096579619300130#s0155.

[4] Dorin-Mircea Popovici et al. *Comparative Study of APIs and Frameworks for Haptic Application Development*. Sept. 2012. URL: https://doi.org/10.1109/cw.2012.13.

[5] Felix G. Hamza-Lup and Ioana Andreea Stănescu. *The haptic paradigm in education: Challenges and case studies*. Jan. 2010. URL: https://doi.org/10.1016/j.iheduc.2009.12.004.

[6] U. Kunzler and C. Runde. "Kinesthetic haptics integration into large-scale virtual environments". In: *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics Conference*. 2005, pp. 551–556. DOI: 10.1109/WHC.2005.84.

[7] Alperen Sayar et al. "High-Performance Real-Time Data Processing: Managing Data Using Debezium, Postgres, Kafka, and Redis". In: *2023 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2023, pp. 1–4. DOI: 10.1109/ASYU58738.2023.10296737.

[8] Shiqi Guan, Wenshan Hu, and Hong Zhou. "Real-Time Data Transmission Method Based on Websocket Protocol for Networked Control System Laboratory". In: *2019 Chinese Control Conference (CCC)*. 2019, pp. 5339–5344. DOI: 10.23919/ChiCC.2019.8865690.

[9] Gulara Muradova, Mehran Hematyar, and Jala Jamalova. "Advantages of Redis in-memory database to efficiently search for healthcare medical supplies using geospatial data". In: *2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT)*. 2022, pp. 1–5. DOI: 10.1109/AICT55583.2022.10013544.

[10] *Redis benchmark on bare metal*. URL: https://gist.github.com/bobrik/c67189e88efcc2a1491c54c15f5fe006.

[11] Lei Wei et al. "Extending support to customised multi-point haptic devices in CHAI3D". In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2014, pp. 1864–1867. DOI: 10.1109/SMC.2014.6974192.

[12] Pooja Thakur and Prashant Jadon. "Django: Developing web using Python". In: *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. 2023, pp. 303–306. DOI: 10.1109/ICACITE57410.2023.10183246.

[13] Zichen Gui et al. *Interactive Manipulation and Visualization of 3D Brain MRI for Surgical Training*. arXiv preprint arXiv:submit/5492377. Submitted to arXiv. 2024. URL: https://arxiv.org/submit/5492377/view.

[14] Boying Wu et al. "Application and Development Prospect of Monitoring Screen based on Three.js Unit Equipment Control System". In: *2022 IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*. 2022, pp. 347–351. DOI: 10.1109/QRS-C57518.2022.00058.