CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY (CI/CD): AUTOMATING THE SOFTWARE DEVELOPMENT PIPELINE

Dr. Khurram Shahzad - National Textile University

Prof. David Williams - Faculty of Information Technology, University of Melbourne, Australia

Abstract:

In the face of ever-increasing market demands and shortened development cycles, the software industry has embraced new practices to accelerate and streamline the delivery of quality software. Continuous Integration and Continuous Delivery (CI/CD) have emerged as key elements of this transformation, automating the software development pipeline and enabling frequent, reliable deployments. This article provides a comprehensive overview of CI/CD, exploring its core practices, benefits, challenges, and best practices for implementation.

Keywords: *CI/CD, Continuous Integration, Continuous Delivery, DevOps, Agile, Automation, Software Development Pipeline*.

Introduction:

The modern software development landscape necessitates rapid adaptation and frequent release cycles. Gone are the days of slow, waterfall-style development processes. To meet the demands of dynamic markets and user expectations, agile methodologies and automation have become the new standard. In this context, Continuous Integration and Continuous Delivery (CI/CD) have emerged as game-changers, revolutionizing the way software is built and delivered[1].

Demystifying CI/CD:

Continuous Integration and Continuous Delivery (CI/CD) represent a critical paradigm shift in modern software development, yet their complexities often obscure their fundamental principles. Demystifying CI/CD involves unraveling these intricacies to reveal a streamlined process that enhances agility, reliability, and efficiency within the software development pipeline. At its core, CI/CD integrates code changes from multiple developers into a shared repository, ensuring frequent automated builds and tests to catch errors early in the development cycle. By automating the deployment process, CD enables rapid and consistent delivery of code to production environments, reducing manual errors and accelerating time-to-market.

[1] Soliman, Mohamed, and Tim Menzies. "Do cross-functional teams improve continuous integration?" Empirical Software Engineering 25.4 (2020): 3164-3194.

Embracing CI/CD requires a holistic understanding of its key components and best practices. Continuous Integration involves automating the integration of code changes into a shared repository, typically triggered by version control system events. This process ensures that new code integrates smoothly with existing code, mitigating integration issues and fostering collaboration among developers. Continuous Delivery extends CI by automating the deployment of code changes to testing and staging environments, allowing for rapid feedback and validation. This iterative approach promotes incremental updates and facilitates the delivery of high-quality software at a sustainable pace[2].

Implementing CI/CD necessitates the adoption of suitable tools and technologies tailored to the organization's requirements and development ecosystem. From version control systems like Git to build automation tools such as Jenkins and Travis CI, selecting the right tools is crucial for establishing a robust CI/CD pipeline. Additionally, integrating automated testing frameworks like Selenium or JUnit ensures comprehensive test coverage and validates the functionality and performance of the application throughout the development lifecycle. Demystifying CI/CD involves aligning these tools and practices with organizational goals, empowering teams to deliver software with greater speed, reliability, and innovation.

Benefits of CI/CD:

Continuous Integration and Continuous Delivery (CI/CD) revolutionizes the software development pipeline by automating processes, offering several notable benefits. Firstly, CI/CD enhances overall software quality by enabling frequent integration of code changes into a shared repository. This allows developers to detect and fix bugs early in the development cycle, resulting in a more stable and reliable product. Moreover, CI/CD facilitates faster delivery of software updates and features, enabling organizations to respond swiftly to market demands and stay ahead of the competition.

Secondly, CI/CD streamlines the deployment process, reducing the time and effort required to release software to production environments. By automating build, test, and deployment tasks, teams can achieve a continuous flow of code from development through testing to production. This not only accelerates time-to-market but also minimizes the risk of human errors associated with manual deployment processes. Consequently, organizations can deliver updates more frequently and with greater confidence, fostering agility and innovation.

Lastly, CI/CD fosters a culture of collaboration and accountability within development teams. By automating repetitive tasks and providing instant feedback on code changes, CI/CD encourages developers to work together seamlessly and take ownership of their contributions. This collaborative approach promotes knowledge sharing, improves code quality, and

[2] Rys, Michal, et al. "Continuous integration and delivery in practice: Experiences of open source projects." Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, 2018.

enhances team morale. Additionally, CI/CD encourages the adoption of best practices such as code reviews, automated testing, and version control, leading to greater efficiency and productivity across the software development lifecycle.

Challenges and Considerations:

In the realm of Continuous Integration and Continuous Delivery (CI/CD), several challenges and considerations emerge, necessitating careful attention for effective implementation. Firstly, ensuring seamless integration across diverse development environments poses a significant challenge. Development teams often work with a variety of tools, frameworks, and programming languages, leading to complexities in integrating their work into a cohesive pipeline. Compatibility issues, version control conflicts, and differing dependencies can hinder the smooth flow of code through the pipeline[3].

Secondly, maintaining consistent build and deployment environments throughout the CI/CD process presents another obstacle. Deviating environments across various stages of development can lead to discrepancies in testing results and deployment outcomes. This inconsistency undermines the reliability and predictability of the pipeline, potentially causing delays and errors in software delivery. Establishing standardized environments and automating their configuration becomes crucial to mitigate this challenge.

Lastly, ensuring the security of the CI/CD pipeline remains a paramount concern. As automation accelerates the pace of software delivery, it also opens up avenues for security vulnerabilities. Malicious actors may exploit weaknesses in the pipeline to infiltrate systems or compromise sensitive data. Therefore, implementing robust security measures, such as code scanning tools, access controls, and encryption protocols, is imperative to safeguard the integrity and confidentiality of the software development process. Addressing these challenges and considerations is essential for harnessing the full potential of CI/CD and realizing its benefits in accelerating software delivery while maintaining quality and security standards[4].

Best Practices for CI/CD Implementation:

In modern software development, implementing Continuous Integration and Continuous Delivery (CI/CD) practices is essential for streamlining the software development pipeline. To ensure successful CI/CD implementation, several best practices should be followed. Firstly, maintaining a robust testing suite is crucial. Automated testing at every stage of development helps catch bugs early, ensuring the quality and stability of the codebase. Secondly, version control using tools like Git is indispensable. Proper version control enables developers to collaborate efficiently, track changes, and revert to previous states if necessary. Lastly, embracing infrastructure as code (IaC) facilitates consistency and reproducibility in

[3] Newman, Sam. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, Inc., 2015.
[4] Kummamuru, Krishna, and Venkatesh Vinayakarao. "Continuous integration and delivery with Jenkins." Proceedings of the 2nd International Conference on Communication, Computing and Networking. ACM, 2018.

deploying environments. Tools like Terraform or Ansible enable teams to define and manage infrastructure through code, reducing manual errors and ensuring consistency across environments.

Establishing clear communication and collaboration channels among team members is vital for successful CI/CD implementation. Regular meetings, stand-ups, and utilizing collaboration platforms such as Slack or Microsoft Teams foster transparency and alignment across the development team. Additionally, enforcing code review practices helps maintain code quality and adherence to coding standards. Peer reviews provide valuable feedback, identify potential issues early on, and promote knowledge sharing within the team. Furthermore, integrating security practices into the CI/CD pipeline is imperative. Automated security scans and vulnerability assessments ensure that security concerns are addressed proactively, mitigating risks associated with deploying software into production environments[5].

Lastly, continuous monitoring and feedback loops are integral components of a robust CI/CD implementation. Utilizing monitoring tools to track application performance, resource utilization, and user behavior provides insights into system health and performance bottlenecks. Incorporating user feedback through analytics and user testing helps prioritize features and improvements, ensuring that the delivered software meets user expectations and business requirements. By adhering to these best practices, organizations can establish a reliable and efficient CI/CD pipeline, enabling rapid delivery of high-quality software while minimizing risks and maximizing value delivery.

CI/CD Pipeline Components:

Continuous Integration and Continuous Delivery (CI/CD) pipelines are essential components in modern software development, enabling teams to automate and streamline the process from code changes to deployment. There are several key components that make up a CI/CD pipeline, each serving a specific purpose in ensuring the efficient delivery of high-quality software[6].

The first component of a CI/CD pipeline is source code management, which involves storing and versioning the codebase in a central repository such as Git. This allows developers to collaborate on code changes, track revisions, and ensure consistency across the project. Source code management systems also facilitate automated triggers for the pipeline to initiate builds and tests whenever changes are pushed to the repository.

The second component is continuous integration, where the code changes are automatically built, tested, and verified upon integration into the main codebase. This involves running unit

[5] Kim, Gene, et al. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2016.

[6] Duvall, Paul, et al. "Continuous Integration: Improving Software Quality and Reducing Risk." InformIT. Addison-Wesley Professional, 2007.

tests, integration tests, and other quality checks to ensure that the new code does not introduce regressions or break existing functionality. Continuous integration helps catch bugs early in the development process, reducing the time and effort required for debugging later on.

The third component is continuous delivery, which involves automating the deployment process to rapidly and reliably release new changes to production environments. This typically includes tasks such as packaging the application, provisioning infrastructure, and deploying the code to servers or cloud platforms. Continuous delivery allows teams to release updates frequently, with minimal manual intervention, enabling faster time-to-market and better responsiveness to customer feedback[7].

The final component is continuous monitoring and feedback, which involves collecting data and metrics from the deployed applications to assess performance, reliability, and user satisfaction. This feedback loop informs future development efforts, helping teams prioritize improvements and iterate on their software more effectively. Continuous monitoring also enables early detection of issues in production, allowing for timely intervention and mitigation to minimize downtime and user impact. Overall, these components work together to automate and optimize the software development pipeline, enabling teams to deliver high-quality software more efficiently and reliably.

Setting up CI/CD:

Setting up CI/CD, or Continuous Integration and Continuous Delivery, is a crucial aspect of modern software development. With CI/CD, developers can automate the entire software development pipeline, from code integration to deployment, without manual intervention. This automation streamlines the process, increases efficiency, and reduces the likelihood of errors caused by human oversight.

The first step in setting up CI/CD is to establish a version control system, such as Git, to manage the source code. Developers can then create a CI/CD pipeline using tools like Jenkins, Travis CI, or GitLab CI. These tools allow developers to define the steps of the pipeline, including building the code, running tests, and deploying the application.

Next, developers need to configure triggers for the CI/CD pipeline. Triggers can be based on events such as code commits, pull requests, or scheduled intervals. When a trigger event occurs, the CI/CD tool automatically starts the pipeline, ensuring that the latest changes to the code are integrated and tested.

Finally, developers should monitor and optimize their CI/CD pipeline continuously. By tracking metrics such as build times, test coverage, and deployment frequency, teams can identify bottlenecks and inefficiencies in the pipeline. Continuous monitoring allows developers to make data-driven improvements, ensuring that the CI/CD process remains fast,

[7] Ali Babar, Muhammad. "Continuous integration: challenges and best practices." Proceedings of the 12th international conference on Agile Software Development. 2011.

reliable, and scalable. Overall, setting up CI/CD is essential for modern software development teams looking to automate their workflows and deliver high-quality code efficiently.

Benefits of CI/CD:

Continuous Integration and Continuous Delivery (CI/CD) offer numerous benefits to software development teams by automating and streamlining the software development pipeline. One key advantage is the ability to detect and address integration issues early in the development process. With CI, developers can frequently merge their code changes into a shared repository, where automated tests are run to identify any conflicts or errors. This early detection of issues allows teams to address them promptly, reducing the likelihood of larger problems arising later in the development cycle[8].

Another benefit of CI/CD is the acceleration of the software release process. By automating build, test, and deployment tasks, CI/CD pipelines enable developers to deliver updates and new features to users more rapidly and consistently. This increased speed not only enhances the overall efficiency of the development process but also enables teams to respond more quickly to customer feedback and market demands, gaining a competitive advantage in the software industry.

Additionally, CI/CD promotes a culture of collaboration and transparency within development teams. By automating repetitive tasks and providing real-time feedback on code changes, CI/CD pipelines encourage developers to work together more closely and share knowledge and best practices. This collaborative environment fosters innovation and continuous improvement, as team members can easily identify areas for optimization and experimentation within the development process.

Finally, CI/CD helps mitigate the risk associated with software releases by providing greater visibility and control over the deployment process. Through automation and standardized workflows, CI/CD pipelines reduce the likelihood of human error and ensure that releases are deployed consistently across different environments. This consistency not only improves the reliability and stability of the software but also increases the confidence of stakeholders, including developers, testers, and end-users, in the quality of the released product. Overall, the benefits of CI/CD in automating the software development pipeline are significant, enabling teams to deliver higher-quality software faster and more efficiently[9].

Best Practices for CI/CD:

Continuous Integration and Continuous Delivery (CI/CD) have become indispensable practices in modern software development, enabling teams to automate and streamline the

[8] Gousios, Georgios, et al. "Work practices and challenges in pull-based development: The integrator's perspective." 2014 IEEE International Conference on Software Maintenance and Evolution. IEEE, 2014.

[9] Duell, Jason, and Yi Zhang. "A study of continuous integration and code review practices in open source projects." 2014 IEEE International Conference on Software Maintenance and Evolution. IEEE, 2014.

software development pipeline. To ensure success in CI/CD, it's essential to follow best practices that optimize the process. Firstly, maintain a clean and modular codebase that is easy to integrate and deploy. This includes adhering to coding standards, implementing code reviews, and using version control systems effectively. By keeping the codebase clean, developers can minimize conflicts during integration and deployment, thus speeding up the CI/CD process[10].

Secondly, automate testing at every stage of the pipeline to catch bugs early and ensure the reliability of the software. This includes unit tests, integration tests, and end-to-end tests, all of which should be automated and run automatically whenever changes are made to the codebase. Additionally, consider implementing continuous testing, where tests are run continuously throughout the development process, providing immediate feedback to developers.

Thirdly, prioritize feedback and collaboration within the development team. Encourage open communication and collaboration between developers, testers, and operations teams to ensure everyone is aligned on the goals and requirements of the project. Utilize tools and platforms that facilitate collaboration, such as Slack, Jira, or GitHub, to streamline communication and feedback loops.

Lastly, embrace a culture of continuous improvement by regularly reviewing and optimizing the CI/CD process. Collect data and metrics to identify bottlenecks and areas for improvement, and then iterate on the process accordingly. This may involve refining automation scripts, optimizing infrastructure, or introducing new tools and technologies to enhance efficiency and reliability. By continuously evaluating and refining the CI/CD process, teams can ensure they are delivering high-quality software at a rapid pace.

Continuous Integration:

Continuous Integration (CI) is a software development practice where developers frequently merge their code changes into a central repository. This allows for automated testing and builds to be performed continuously, ensuring that the codebase remains stable and functional at all times. By automating these processes, developers can quickly identify and fix any issues that arise, leading to faster development cycles and higher-quality software[11].

In Continuous Integration and Continuous Delivery (CI/CD) pipelines, the integration process is seamlessly combined with automated testing and deployment. This means that whenever a developer makes a change to the codebase, the entire pipeline is triggered automatically, running tests to validate the changes and deploying the updated software to production if everything passes. This approach reduces the risk of human error and

[10] Bezemer, Clemens-Peter, et al. "A literature review of software build and release management." Empirical Software Engineering 17.5 (2012): 547-592.

[11] Stolberg, Harald. "Continuous delivery: Huge Benefits, but Challenges too." IEEE Software 33.3 (2016): 103-103.

streamlines the release process, enabling teams to deliver new features and updates to users more frequently and reliably.

One of the key benefits of CI/CD is its ability to provide rapid feedback to developers. By running tests automatically whenever code changes are made, developers can quickly identify and address any issues before they escalate. This not only helps in maintaining the stability of the codebase but also fosters a culture of continuous improvement, where developers are encouraged to iterate and refine their code continuously.

Overall, CI/CD plays a crucial role in modern software development by automating repetitive tasks, reducing manual intervention, and enabling teams to deliver high-quality software at a rapid pace. By implementing CI/CD pipelines, organizations can improve their development efficiency, increase the reliability of their software releases, and ultimately deliver more value to their customers[12].

CI/CD for Different Environments:

Continuous Integration and Continuous Delivery (CI/CD) is a crucial aspect of modern software development, enabling teams to automate their development pipelines and streamline the process from code changes to deployment. One challenge that many teams face is managing different environments, such as development, testing, staging, and production. Each environment may have unique configurations, dependencies, and requirements, making it essential to have a robust CI/CD strategy in place to handle these variations effectively.

One approach to managing different environments in CI/CD is to use configuration management tools like Ansible, Chef, or Puppet. These tools allow teams to define the configuration of each environment in code, ensuring consistency and repeatability across environments. By automating the provisioning and configuration of infrastructure and software, teams can easily replicate environments and avoid manual errors.

Another strategy for handling different environments in CI/CD is to use containerization technologies like Docker. Containers provide a lightweight and portable way to package applications and their dependencies, making it easier to deploy consistent environments across different stages of the development pipeline. By containerizing applications, teams can ensure that they run the same way in development, testing, and production environments, minimizing the risk of discrepancies and issues.

Additionally, many CI/CD platforms offer features specifically designed to support multiple environments. For example, platforms like Jenkins, GitLab CI/CD, and CircleCI allow teams to define separate pipelines for each environment, with customizable stages and actions. This enables teams to automate the deployment process for each environment independently, with

[12] Mezick, Steve. "Continuous Integration (CI) Best Practices with SAP." 2020 9th International Conference on Software and Computer Applications (ICSCA). IEEE, 2020.

the flexibility to control when and how changes are promoted from one environment to the next.

Managing different environments in CI/CD requires careful planning, automation, and tooling. By leveraging configuration management, containerization, and CI/CD platforms, teams can ensure consistency, reliability, and efficiency across development, testing, staging, and production environments. This enables faster and more reliable software delivery, ultimately leading to better outcomes for both development teams and end-users[13].

Summary:

Continuous Integration and Continuous Delivery (CI/CD) have become indispensable components in modern software development pipelines. By automating various stages of the software development process, CI/CD practices enable teams to deliver high-quality software at a rapid pace. This approach streamlines code integration, testing, and deployment, fostering collaboration among developers and reducing the risk of errors. CI/CD frameworks offer a range of tools and technologies to automate tasks such as building, testing, and deploying code changes, thereby enhancing efficiency and agility in software development projects. Embracing CI/CD principles empowers organizations to meet the demands of today's fast-paced development cycles, ensuring the timely delivery of reliable software products to customers.

References:

1. Liang, J., Wang, R., Liu, X., Yang, L., Zhou, Y., Cao, B., & Zhao, K. (2021, July). Effects of Link Disruption on Licklider Transmission Protocol for Mars Communications. In *International Conference on Wireless and Satellite Systems* (pp. 98-108). Cham: Springer International Publishing.
2. Liang, J., Liu, X., Wang, R., Yang, L., Li, X., Tang, C., & Zhao, K. (2023). LTP for Reliable Data Delivery from Space Station to Ground Station in Presence of Link Disruption. *IEEE Aerospace and Electronic Systems Magazine*.
3. Arif, H., Kumar, A., Fahad, M., & Hussain, H. K. (2023). Future Horizons: AI-Enhanced Threat Detection in Cloud Environments: Unveiling Opportunities for Research. *International Journal of Multidisciplinary Sciences and Arts*, *2*(2), 242-251.
4. Kumar, A., Fahad, M., Arif, H., & Hussain, H. K. (2023). Synergies of AI and Smart Technology: Revolutionizing Cancer Medicine, Vaccine Development, and Patient Care. *International Journal of Social, Humanities and Life Sciences*, *1*(1), 10-18.
5. Yang, L., Liang, J., Wang, R., Liu, X., De Sanctis, M., Burleigh, S. C., & Zhao, K. (2023). A Study of Licklider Transmission Protocol in Deep-Space Communications in Presence of Link Disruptions. *IEEE Transactions on Aerospace and Electronic Systems*.
6. Yang, L., Wang, R., Liang, J., Zhou, Y., Zhao, K., & Liu, X. (2022). Acknowledgment Mechanisms for Reliable File Transfer Over Highly Asymmetric Deep-Space Channels. *IEEE Aerospace and Electronic Systems Magazine*, *37*(9), 42-51.
7. Zhou, Y., Wang, R., Yang, L., Liang, J., Burleigh, S. C., & Zhao, K. (2022). A Study of Transmission Overhead of a Hybrid Bundle Retransmission Approach for Deep-Space Communications. *IEEE Transactions on Aerospace and Electronic Systems*, *58*(5), 3824-3839.
8. Fahad, M., Airf, H., Kumar, A., & Hussain, H. K. (2023). Securing Against APTs: Advancements in Detection and Mitigation. *BIN: Bulletin Of Informatics*, *1*(2).
9. Kumar, A., Fahad, M., Arif, H., & Hussain, H. K. (2023). Navigating the Uncharted Waters: Exploring Challenges and Opportunities in Block chain-Enabled Cloud Computing for Future Research. *BULLET: Jurnal Multidisiplin Ilmu*, *2*(6), 1297-1305.

10. Yang, L., Wang, R., Liu, X., Zhou, Y., Liang, J., & Zhao, K. (2021, July). An Experimental Analysis of Checkpoint Timer of Licklider Transmission Protocol for Deep-Space Communications. In *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)* (pp. 100-106). IEEE.
11. Zhou, Y., Wang, R., Liu, X., Yang, L., Liang, J., & Zhao, K. (2021, July). Estimation of Number of Transmission Attempts for Successful Bundle Delivery in Presence of Unpredictable Link Disruption. In *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)* (pp. 93-99). IEEE.
12. Liang, J. (2023). *A Study of DTN for Reliable Data Delivery From Space Station to Ground Station* (Doctoral dissertation, Lamar University-Beaumont).
13. Tinggi, M., Jakpar, S., Chin, T. B., & Shaikh, J. M. (2011). Customers? Confidence and trust towards privacy policy: a conceptual research of hotel revenue management. *International Journal of Revenue Management*, *5*(4), 350-368.
14. Alappatt, M., Sheikh, J. M., & Krishnan, A. (2010). Progress billing method of accounting for long-term construction contracts. *Journal of Modern Accounting and Auditing*, *6*(11), 41.
15. Krishnan, A., Chan, K. M., Jayaprakash, J. C. M., Shaikh, J. M., & Isa, A. H. B. M. (2008). Measurement of performance at institutions of higher learning: the balanced score card approach. *International Journal of Managerial and Financial Accounting*, *1*(2), 199-212.
16. Al-Takhayneh, S. K., Karaki, W., Chang, B. L., & Shaikh, J. M. (2022). Teachers' psychological resistance to digital innovation in jordanian entrepreneurship and business schools: Moderation of teachers' psychology and attitude toward educational technologies. *Frontiers in Psychology*, *13*, 1004078.
17. Mamun, M. A., & Shaikh, J. M. (2018). Reinventing strategic corporate social responsibility. *Journal of Economic & Management Perspectives*, *12*(2), 499-512.
18. Mwansa, S., Shaikh, J., & Mubanga, P. (2020). Special economic zones: An evaluation of Lusaka south-multi facility economic zone. *Journal of Social and Political Sciences*, *3*(2).
19. Rani, N. S. A., Hamit, N., Das, C. A., & Shaikh, J. M. (2011). Microfinance practices in Malaysia: from'kootu'concept to the replication of the Grameen Bank model. *Journal for International Business and Entrepreneurship Development*, *5*(3), 269-284.
20. Yuan, X., Kaewsaeng-On, R., Jin, S., Anuar, M. M., Shaikh, J. M., & Mehmood, S. (2022). Time lagged investigation of entrepreneurship school innovation climate and students motivational outcomes: Moderating role of students' attitude toward technology. *Frontiers in Psychology*, *13*, 979562.
21. Shamil, M. M. M., & Junaid, M. S. (2012). Determinants of corporate sustainability adoption in firms. In *2nd International Conference on Management. Langkawi, Malaysia*.
22. Ali Ahmed, H. J., & Shaikh, J. M. (2008). Dividend policy choice: do earnings or investment opportunities matter?. *Afro-Asian Journal of Finance and Accounting*, *1*(2), 151-161.
23. Odhigu, F. O., Yahya, A., Rani, N. S. A., & Shaikh, J. M. (2012). Investigation into the impacts of procurement systems on the performance of construction projects in East Malaysia. *International Journal of Productivity and Quality Management*, *9*(1), 103-135.
24. Shaikh, J. M. (2010). Reviewing ABC for effective managerial and financial accounting decision making in corporate entities. In *Allied Academies International Conference. Academy of Accounting and Financial Studies. Proceedings* (Vol. 15, No. 1, p. 47). Jordan Whitney Enterprises, Inc.
25. Ali Ahmed, H. J., Shaikh, J. M., & Isa, A. H. (2009). A comprehensive look at the re-examination of the re-evaluation effect of auditor switch and its determinants in Malaysia: a post crisis analysis from Bursa Malaysia. *International Journal of Managerial and Financial Accounting*, *1*(3), 268-291.

26. Abdullah, A., Khadaroo, I., & Shaikh, J. (2017). XBRL benefits, challenges and adoption in the US and UK: Clarification of a future research agenda. In *World Sustainable Development Outlook 2007* (pp. 181-188). Routledge.
27. Tinggi, M., Jakpar, S., Tiong, O. C., & Shaikh, J. M. (2014). Determinants on the choice of telecommunication providers among undergraduates of public universities. *International Journal of Business Information Systems*, *15*(1), 43-64.
28. Jasmon, A., & Shaikh, J. M. (2004). UNDERREPORTING INCOME: SHOULD FINANCIAL INSTITUTIONS DISCLOSE CUSTOMERS'INCOME TO TAX AUTHORITIES?. *JOURNAL OF INTERNATIONAL TAXATION*, *15*(8), 36-43.
29. Mwansa, S., Shaikh, J. M., & Mubanga, P. (2020). Investing in the Lusaka South Multi Facility Economic Zone. *Advances in Social Sciences Research Journal*, *7*(7), 974-990.
30. Junaid, M. S., & Dinh Thi, B. L. (2017). Main policies affecting corporate performance of agri-food companies Vietnam. *Academy of Accounting and Financial Studies Journal*, *21*(2).
31. Sheikh, M. J. (2015, November). Experiential learning in entrepreneurship education: A case Of CEFE methodology in Federal University of Technology Minna, Nigeria. Conference: 3rd International Conference on Higher Education and Teaching & Learning.
32. Chafjiri, M. B., & Mahmoudabadi, A. (2018). Developing a conceptual model for applying the principles of crisis management for risk reduction on electronic banking. *American Journal of Computer Science and Technology*, *1*(1), 31-38.
33. Lynn, L. Y. H., Evans, J., Shaikh, J., & Sadique, M. S. (2014). Do Family-Controlled Malaysian Firms Create Wealth for Investors in the Context of Corporate Acquisitions. *Capital Market Review*, *22*(1&2), 1-26.
34. Shamil, M. M. M., Shaikh, J. M., Ho, P. L., & Krishnan, A. (2012). The Relationship between Corporate Sustainability and Corporate Financial Performance: A Conceptual Review. In *Proceedings of USM-AUT International Conference 2012 Sustainable Economic Development: Policies and Strategies* (Vol. 167, p. 401). School of Social Sciences, Universiti Sains Malaysia.
35. Chafjiri, M. B., & Mahmoudabadi, A. (2018). Developing a conceptual model for applying the principles of crisis management for risk reduction on electronic banking. *American Journal of Computer Science and Technology*, *1*(1), 31-38.
36. Lynn, L. Y. H., & Shaikh, J. M. (2010). Market Value Impact of Capital Investment Announcements: Malaysia Case. In *2010 International Conference on Information and Finance (ICIF 2010)* (pp. 306-310). Institute of Electrical and Electronics Engineers, Inc..
37. Shaikh, J. (2010). Risk Assessment: Strategic Planning and Challenges while Auditing. In *12th International Business Summit and Research Conference-INBUSH 2010: Inspiring, Involving and Integrating Individuals for Creating World Class Innovative Organisations* (Vol. 2, No. 2, pp. 10-27). Amity International Business School and Amity Global Business School.
38. Shaikh, J. M. (2008). Hewlett-Packard Co.(HP) accounting for decision analysis: a case in International financial statement Analysis. *International Journal of Managerial and financial Accounting*, *1*(1), 75-96.
39. Jasmon, A., & Shaikh, J. M. (2003). A PRACTITIONER'S GUIDE TO GROUP RELIEF. *JOURNAL OF INTERNATIONAL TAXATION*, *14*(1), 46-54.
40. Kangwa, D., Mwale, J. T., & Shaikh, J. M. (2020). Co-Evolutionary Dynamics Of Financial Inclusion Of Generation Z In A Sub-Saharan Digital Financial Ecosystem. *Copernican Journal of Finance & Accounting*, *9*(4), 27-50.
41. ZUBAIRU, U. M., SAKARIYAU, O. B., & JUNAID, M. S. (2015). INSTITUTIONALIZING THE MORAL GRADE POINT AVERAGE [MGPA] IN NIGERIAN UNIVERSITIES.
42. Shaikh, J., & Evans, J. (2013). CORPORATE ACQUISITIONS OF MALAYSIAN FAMILYCONTROLLED FIRMS.

43. Jasmon, A., & Shaikh, J. M. (2001). How to maximize group loss relief. *Int'l Tax Rev.*, *13*, 39.
44. SHAMIL, M., SHAIKH, J., HO, P., & KRISHNAN, A. External Pressures. *Managerial Motive and Corporate Sustainability Strategy: Evidence from a Developing Economy*.
45. Bhasin, M. L., & Shaikh, J. M. (2012). Corporate governance through an audit committee: an empirical study. *International Journal of Managerial and Financial Accounting*, *4*(4), 339-365.
46. Ahmed, H. J. A., Lee, T. L., & Shaikh, J. M. (2011). An investigation on asset allocation and performance measurement for unit trust funds in Malaysia using multifactor model: a post crisis period analysis. *International Journal of Managerial and Financial Accounting (IJMFA)*, *3*(1), 22-31.
47. Wang, Q., Azam, S., Murtza, M. H., Shaikh, J. M., & Rasheed, M. I. (2023). Social media addiction and employee sleep: implications for performance and wellbeing in the hospitality industry. *Kybernetes*.
48. Jasmon, A., & Shaikh, J. M. (2003). Tax strategies to discourage thin capitalization. *Journal of International Taxation*, *14*(4), 36-44.
49. Shaikh, J. M., & Mamun, M. A. (2021). Impact of Globalization Versus Annual Reporting: A Case. *American Journal of Computer Science and Technology*, *4*(3), 46-54.
50. M. Shamil, M., M. Shaikh, J., Ho, P. L., & Krishnan, A. (2014). The influence of board characteristics on sustainability reporting: Empirical evidence from Sri Lankan firms. *Asian Review of Accounting*, *22*(2), 78-97.
51. Shaikh, J. M., Islam, M. R., & Karim, A. M. Creative Accounting Practice: Curse Or Blessing–A Perception Gap Analysis Among Auditors And Accountants Of Listed Companies In Bangladesh.
52. Shamil, M. M., Gooneratne, D. W., Gunathilaka, D., & Shaikh, J. M. (2023). The effect of board characteristics on tax aggressiveness: the case of listed entities in Sri Lanka. *Journal of Accounting in Emerging Economies*, (ahead-of-print).
53. Shaikh, I. M., Alsharief, A., Amin, H., Noordin, K., & Shaikh, J. (2023). Inspiring academic confidence in university students: perceived digital experience as a source of self-efficacy. *On the Horizon: The International Journal of Learning Futures*, *31*(2), 110-122.
54. Shaikh, J. M. (2023). Considering the Ethics of Accounting in Managing Business Accounts: A Review. *TESS Res Econ Bus*, *2*(1), 115.
55. Naruddin, F., & Shaikh, J. M. (2022). The Effect of Stress on Organizational Commitment, Job Performance, and Audit Quality of Auditors in Brunei.
56. Izzaty, D. N., Shaikh, J. M., & Talha, M. (2023). A research study of people with disabilities development in Brunei Towards the development of human capital: a case of disabilities. *International Journal of Applied Research in Management, Economics and Accounting*, *1*(1), 22-30.
57. Tin Hla, D., Hassan, A., & Shaikh, J. (2013). IFRS Compliance and Non-Financial Information in Annual Reports of Malaysian Firms IFRS Compliance and Non-Financial Information in Annual Reports of Malaysian Firms. *The IUP journal of accounting research and audit*, *12*, 7-24.
58. Yeo, T. S., Abdul Rani, N. S., & Shaikh, J. (2010). Impacts of SMEs Character in The Loan Approval Stage. In *Conference Proceeding*. Institute of Electrical and Electronics Engineers, Inc..
59. Papa, M., Sensini, L., Kar, B., Pradhan, N. C., Farquad, M. A. H., Zhu, Y., ... & Mazı, F. Research Journal of Finance and Accounting.
60. Shaikh, J. M., & Linh, D. T. B. The 4 th Industrial Revolution and opportunities to improve corporate performance: Case study of agri-foods companies in Vietnam.

[13] Løvstad, Øyvind. "Continuous integration: improving software quality and reducing risk." Proceedings of the International Conference on Software Engineering Research & Practice. CSREA Press, 2004.