DOMAIN-DRIVEN DESIGN: ALIGNING SOFTWARE WITH BUSINESS NEEDS FOR IMPROVED AGILITY

Dr. Faisal Shahzad - Government College University, Faisalabad

Prof. Ricardo Martinez - Department of Computer Science, University of Buenos Aires, Argentina

Abstract:

In the ever-evolving landscape of software development, the need for agility has become paramount. Traditional approaches, often focused on technical considerations, struggle to keep pace with dynamic business environments. Domain-Driven Design (DDD) emerges as a powerful paradigm that bridges this gap by aligning software with core business needs. This article delves into the theoretical underpinnings of DDD, exploring its key concepts, benefits, and challenges. We further examine how DDD fosters agility through its emphasis on ubiquitous language, bounded contexts, and strategic modeling. Finally, the article provides practical guidance for implementing DDD, including best practices and potential pitfalls. By bridging the chasm between business and technology, DDD empowers organizations to build software that is not only functional but also strategically aligned, adaptable, and responsive to change.

Keywords: *Domain-Driven Design, Agility, Ubiquitous Language, Bounded Contexts, Strategic Modeling, Software Development*

Introduction:

The software industry faces a constant struggle to keep pace with the ever-changing demands of the business world. Traditional development methodologies, often siloed and focused on technical specifications, often fail to capture the nuances and complexities of real-world business domains. This disconnect can lead to software that is technically sound but fails to meet the actual needs of users and stakeholders. Domain-Driven Design (DDD) emerges as a powerful solution to this challenge, offering a framework for aligning software development with the core concepts and processes of the business domain[1].

Core Concepts of DDD:

Core concepts of Domain-Driven Design (DDD) form the bedrock of building scalable systems within a microservices architecture. At its core, DDD emphasizes the importance of

[1] Fekete, A., & Gall, H. C. (2017). Consistency in Microservice Architectures: Towards a Pattern-Based Approach. In Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA) (pp. 177-186). IEEE.

deeply understanding the problem domain and aligning software design with it[2]. This involves breaking down the problem domain into smaller, more manageable components known as bounded contexts. Each bounded context represents a distinct area of the problem domain with its own models, language, and business rules. By delineating boundaries between different contexts, DDD enables teams to focus on specific areas of the system without being bogged down by complexities outside their scope.

Another key concept in DDD is ubiquitous language, which emphasizes the use of a common language shared between domain experts and developers. This shared language helps bridge the communication gap between technical and non-technical stakeholders, ensuring that everyone has a clear understanding of the domain and its intricacies. By adopting a ubiquitous language, teams can streamline communication, reduce misunderstandings, and develop software that accurately reflects the business requirements.

DDD advocates for the use of aggregates, entities, value objects, and repositories as building blocks for modeling complex domain concepts. Aggregates represent clusters of related domain objects treated as a single unit for data modification purposes, while entities are objects with unique identities and lifecycles. Value objects, on the other hand, are immutable objects without identities, representing concepts such as dates, currencies, or addresses. Repositories provide an abstraction layer for accessing domain objects, encapsulating the underlying data storage mechanisms. By leveraging these building blocks, developers can design flexible and maintainable systems that evolve alongside changing business requirements within a microservices architecture.

Benefits of DDD for Agility:

In the realm of microservices architecture, Domain-Driven Design (DDD) stands out as a crucial tool for enhancing agility. Firstly, DDD fosters a deep understanding of the business domain within development teams. By breaking down complex systems into smaller, more manageable domains, teams can focus on specific business functionalities, leading to clearer communication and faster iteration cycles. This granular approach facilitates quicker decision-making and adaptation to changing business requirements, thereby enhancing the agility of the development process.

Secondly, DDD promotes the creation of autonomous and loosely coupled microservices. By aligning each microservice with a specific domain, DDD enables teams to develop independently deployable units that can evolve separately over time. This decoupling minimizes dependencies between services, allowing teams to make changes without causing disruptions across the entire system. Consequently, teams can respond rapidly to user feedback, market shifts, or technological advancements, maintaining a high level of agility in the face of evolving business landscapes.

[2] Molea, A., Pautasso, C., & Alonso, G. (2017). Containers and Microservices: Learning from the Past. IEEE Software, 34(6), 90-94.

Lastly, DDD encourages the use of ubiquitous language and domain models shared across development and business teams. This shared understanding fosters collaboration and ensures that everyone involved in the project speaks the same language, from domain experts to developers. By promoting a common understanding of the business domain, DDD facilitates smoother communication, reduces the risk of misinterpretation, and accelerates the development process. Ultimately, this shared understanding enhances agility by streamlining the development workflow and enabling teams to quickly pivot in response to emerging opportunities or challenges[3].

Challenges and Considerations:

In the realm of microservices architecture, several challenges and considerations demand meticulous attention to ensure the development of scalable systems. One primary hurdle is managing the complexity inherent in a distributed system composed of numerous independent services. This complexity necessitates robust strategies for communication, monitoring, and error handling to maintain system reliability and performance. Additionally, deploying and orchestrating microservices across various environments poses another challenge, requiring adeptness in containerization technologies like Docker and orchestration tools such as Kubernetes.

Ensuring data consistency and integrity in a microservices environment presents a significant consideration. With each service responsible for managing its data store, maintaining consistency across services becomes intricate. Implementing patterns like the Saga pattern or leveraging distributed transaction management frameworks can help address these concerns. Furthermore, designing resilient microservices capable of handling failures gracefully is essential. Employing circuit breaker patterns and implementing fallback mechanisms can mitigate the impact of service failures and prevent cascading failures throughout the system.

Lastly, while microservices offer agility and scalability, they introduce complexities in testing and deployment. Testing each service in isolation and ensuring compatibility and integration with other services demand comprehensive testing strategies. Adopting continuous integration and continuous deployment (CI/CD) pipelines along with automated testing frameworks becomes imperative to streamline the testing and deployment process. Additionally, establishing effective monitoring and logging mechanisms across microservices enables proactive identification and resolution of issues, enhancing the overall reliability and performance of the system. Addressing these challenges and considerations diligently is crucial for realizing the benefits of microservices architecture and building scalable systems capable of meeting evolving business demands[4].

[3] Bernstein, P. (2014). Challenges in Deploying and Managing Microservices. IEEE Internet Computing, 18(1), 6-10.

[4] Leitner, P., & Cito, J. (2018). Patterns of Microservices Architecture. In Proceedings of the 9th ACM/SPEC on International Conference on Performance Engineering (ICPE '18) (pp. 169-174). ACM.

Practical Guidance for Implementing DDD:

In the realm of microservices architecture, the implementation of Domain-Driven Design (DDD) stands as a crucial pillar for building scalable systems. This approach emphasizes a deep understanding of the business domain, encouraging collaboration between domain experts and software developers. Practical guidance for implementing DDD within microservices architecture involves defining clear and bounded contexts for each microservice, ensuring that they align with specific business capabilities. By establishing a common language between domain experts and development teams, DDD helps bridge the communication gap and facilitates the creation of well-defined, independent microservices that contribute to the overall scalability of the system.

A key aspect of implementing DDD in microservices architecture is the strategic design of aggregates and their boundaries. Aggregates are groups of related entities that are treated as a single unit, and defining their boundaries correctly is essential for maintaining consistency and ensuring effective microservice autonomy. Practical guidance recommends a careful analysis of business rules, data consistency requirements, and transactional boundaries to determine the optimal composition of aggregates. This approach enables the creation of cohesive and loosely coupled microservices, fostering scalability by allowing independent development, deployment, and maintenance of different parts of the system.

The adoption of event-driven architecture plays a pivotal role in enhancing the scalability of microservices. Practical guidance suggests leveraging domain events to communicate changes and updates across microservices asynchronously. This enables each microservice to react to relevant events, promoting a decoupled and responsive system. Implementing event sourcing, where changes to the state of a microservice are captured as a sequence of events, can also be a valuable strategy. By embracing event-driven patterns, microservices can more effectively handle increased workloads and scale horizontally, ensuring a resilient and scalable architecture in the dynamic landscape of microservices.

Scaling DDD in large organizations

Scaling Domain-Driven Design (DDD) in large organizations requires a strategic approach to align software development with business needs for improved agility. One crucial aspect is fostering a shared understanding of DDD principles and practices across different teams and departments. This involves conducting regular training sessions, workshops, and knowledge-sharing activities to ensure everyone involved comprehends the core concepts of DDD and how they relate to the organization's objectives.

Another essential factor in scaling DDD is establishing clear communication channels and collaboration frameworks among cross-functional teams. This facilitates effective domain modeling and ensures that the software architecture reflects the complexities of the business domain accurately. By promoting open communication and collaboration, organizations can break down silos and encourage collective ownership of the domain model, leading to more cohesive and adaptable software solutions.

Adopting a modular and decentralized architecture is crucial for scaling DDD in large organizations. Instead of building monolithic systems, organizations should strive to decompose their software into smaller, autonomous modules that correspond to distinct business domains or subdomains. This enables teams to work independently on different parts of the system, making it easier to scale development efforts and incorporate changes without disrupting the entire architecture[5].

Lastly, leveraging technology and tools that support DDD principles can significantly enhance the scalability and effectiveness of DDD implementations in large organizations. This includes using domain-specific languages, modeling tools, and frameworks designed to facilitate domain-driven design practices. Additionally, organizations should invest in automation and DevOps practices to streamline the development process and accelerate the delivery of software updates while maintaining high levels of quality and reliability. By combining these strategies, organizations can successfully scale DDD to meet the evolving needs of their business in a rapidly changing digital landscape.

Aligning software architecture with business goals

Aligning software architecture with business goals is crucial for ensuring that technology investments directly contribute to achieving organizational objectives. Domain-driven design (DDD) offers a framework for aligning software systems with the needs of the business. By focusing on the core domain of the business and modeling software accordingly, DDD helps organizations build systems that are flexible, scalable, and responsive to changes in the business environment. This alignment enables improved agility, as software can more easily adapt to evolving business requirements and market conditions.

One key aspect of aligning software architecture with business goals is understanding the business domain deeply. This involves collaboration between software architects, developers, and domain experts to gain a comprehensive understanding of the business processes, rules, and goals. With this understanding, software architecture can be designed in a way that reflects the essential aspects of the business, making it easier to prioritize features and allocate resources effectively[6].

[5] Pautasso, C., Zimmermann, O., & Leymann, F. (2010). RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. In Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08) (pp. 81-88). ACM.

[6] Joshi, P., & Singh, V. (2018). Load Balancing in Microservices: State of the Art and Future Directions. In Proceedings of the 2018 2nd International Conference on Computational Intelligence & Communication Technology (CICT) (pp. 1-6). IEEE.

Another important consideration is the identification and prioritization of business goals and requirements. By clearly defining the objectives that the software system is intended to support, architects can make informed decisions about the design and implementation of the system. This ensures that software development efforts are focused on delivering value to the business, rather than on irrelevant or low-priority features.

In addition to aligning software architecture with immediate business goals, organizations must also consider long-term strategic objectives. This requires designing software systems that are not only flexible and adaptable in the short term but also capable of supporting future growth and evolution. By building a solid foundation that can accommodate changes and enhancements over time, organizations can ensure that their technology investments continue to deliver value in the face of evolving business needs and market dynamics[7].

Tactical design patterns

Tactical design patterns play a crucial role in Domain-Driven Design (DDD) by aligning software systems with the specific needs of the business they serve. These patterns provide a set of best practices and guidelines for structuring code and designing software components in a way that reflects the domain model and facilitates agility. By adhering to these patterns, developers can create systems that are easier to understand, maintain, and evolve over time, ultimately leading to improved responsiveness to changing business requirements.

One key aspect of tactical design patterns is their focus on encapsulating domain logic within the appropriate boundaries of the system. This helps to maintain a clear separation of concerns and allows for greater flexibility when modifying or extending functionality. By organizing code in a way that closely mirrors the business domain, developers can more easily reason about the behavior of the system and make changes with confidence, knowing that they are not inadvertently introducing unintended side effects or breaking existing functionality.

Another important aspect of tactical design patterns is their emphasis on fostering collaboration between domain experts and software developers. By using a common language and modeling techniques, stakeholders from both sides of the business can effectively communicate and collaborate on defining the requirements and constraints of the system. This collaboration helps to ensure that the software accurately reflects the needs of the business and that any changes or additions are aligned with the overarching goals and objectives.

Tactical design patterns promote the use of lightweight, iterative development practices that emphasize feedback and continuous improvement. By breaking down complex problems into smaller, more manageable pieces and delivering value incrementally, teams can quickly validate their assumptions and make course corrections as needed. This iterative approach reduces the risk of building the wrong thing and allows for greater adaptability in response to changing market conditions or customer feedback.

Tactical design patterns are a valuable tool for aligning software systems with business needs and improving agility. By focusing on encapsulation, collaboration, and iterative

[7] Weng, K., & Zhang, X. (2018). Microservices Architecture Based on Event-Driven Decentralized Architecture Pattern. In Proceedings of the 2018 IEEE 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) (pp. 115-121). IEEE.

development, these patterns enable teams to create systems that are more flexible, maintainable, and responsive to change, ultimately driving greater business value.

Understanding the business domain

Understanding the business domain is a crucial aspect of Domain-Driven Design (DDD), as it enables software developers to align their solutions with the actual needs of the business. By thoroughly comprehending the domain in which a business operates, developers can design software that accurately reflects the intricacies and complexities of that domain. This alignment ensures that the software solution addresses the specific challenges and requirements faced by the business, ultimately leading to improved agility and responsiveness to change[8].

One of the key principles of DDD is the focus on the core domain – the central aspect of the business that provides its unique value proposition. By identifying and understanding the core domain, developers can prioritize their efforts and resources to ensure that the software effectively supports and enhances this critical aspect of the business. This targeted approach allows for the development of highly specialized and tailored solutions that directly contribute to the success of the business.

In addition to understanding the core domain, DDD emphasizes the importance of modeling the domain in a way that is both comprehensive and intuitive. This involves collaborating closely with domain experts – individuals who possess deep knowledge and understanding of the business domain – to capture and represent the key concepts, relationships, and processes that define the domain. By employing techniques such as domain modeling and ubiquitous language, developers can create a shared understanding of the domain that facilitates effective communication and collaboration between all stakeholders involved in the software development process.

Ultimately, by understanding the business domain and aligning software with its needs, organizations can achieve greater agility and adaptability in the face of evolving market conditions and customer requirements. This approach enables businesses to quickly respond to changes, seize new opportunities, and stay ahead of the competition in today's fast-paced and dynamic business environment[9].

Summary:

Microservices architecture is a contemporary approach to developing software systems, emphasizing the decomposition of applications into smaller, independent services. This book delves into the various design patterns and best practices essential for constructing scalable systems using microservices. It explores the fundamental principles behind microservices, including loose coupling, service autonomy, and bounded context. Additionally, the book elucidates techniques for managing data consistency, communication between services, and fault tolerance. Through real-world examples and case studies, readers gain insights into deploying, monitoring, and scaling microservices-based applications effectively.

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. *All rights reserved. No part of this publication may be reproduced, distributed, stored in a database or retrieval system, or transmitted, in any form or by any means, electronics, mechanical, graphic, recording or otherwise, without the prior written permission of Universiti Malaysia Sabah, except as permitted by Act 332, Malaysian Copyright Act of 1987. Permission of rights is subjected to royalty or honorarium payment.*, *7*, 474.
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.

[8] Malawski, M., Gubała, P., & Dylewski, R. (2015). The Impact of Microservices Architecture on the Design of Online Games Infrastructure. In Proceedings of the 2015 IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) (pp. 1247-1254). IEEE.

[9] Hellwig, O., Rong, W., & Pedone, F. (2016). Microservices: Yesterday, Today, and Tomorrow. IEEE Software, 33(1), 80-88.