Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2024, 11(5):25-32

Research Article ISSN: 2394 - 658X

Theoretical Framework for Requirements Engineering: DevOps

Iqtiar Md Siddique

Department of Mechanical Engineering, the University of Texas at El Paso, US.
Email: iqtiar.siddique@gmail.com.

ABSTRACT

This paper presents a comprehensive theoretical framework for requirements engineering, elucidating the foundational principles, methodologies, and practices essential for the effective elicitation, analysis, specification, validation, and management of requirements in software development projects. Drawing from established literature and industry best practices, the framework provides a structured approach to requirements engineering, emphasizing the importance of stakeholder collaboration, communication, and alignment throughout the software development lifecycle. Key concepts such as requirement types, elicitation techniques, modeling methodologies, traceability, and change management are systematically discussed within the context of the framework. Additionally, the paper explores the role of requirements engineering in addressing evolving business needs, fostering innovation, and mitigating project risks. By adopting this theoretical framework, practitioners and researchers can gain a deeper understanding of the fundamental principles underpinning requirements engineering and leverage them to enhance the quality, reliability, and success of software-intensive systems.

Key words: Framework, DevOps, Requirements Engineering.

---

1. INTRODUCTION

Requirements engineering serves as a critical phase in the software development lifecycle, acting as the cornerstone for successful system design, implementation, and deployment. It encompasses a systematic approach to identifying, eliciting, analyzing, specifying, validating, and managing the requirements of a software system, ensuring that it meets the needs and expectations of stakeholders while aligning with organizational goals and objectives. As software systems become increasingly complex and interconnected, the importance of robust requirements engineering practices cannot be overstated, as they form the basis for effective communication, collaboration, and decision-making among project stakeholders. A qualitative research methodology was employed, focusing on four specific software companies located in Uganda, purposefully selected to participate in the study. Data collection was conducted using questionnaires. The framework design requirements were gathered and honed through a combination of primary data gathered directly from the companies involved and secondary data from existing sources. The critical needs for enhancing processes within small and medium-sized software enterprises were pinpointed as follows: active engagement of users, adoption of an evolutionary approach to requirements engineering process improvement (REPI), effective change management practices, provision of training and educational opportunities, and demonstration of support and commitment from management [4]. According to some, DevOps, which merges Development and Operations, represents a fresh perspective within the realm of software engineering, garnering significant interest of late. As it is a relatively nascent term and concept, a universally accepted definition of DevOps remains elusive. Consequently, existing definitions often capture only certain aspects relevant to the concept, rather than providing a comprehensive understanding [8].

In recent years, the field of requirements engineering has witnessed significant advancements driven by emerging technologies, evolving business environments, and changing user preferences. Consequently, there is

a growing need for a comprehensive theoretical framework that can provide a structured and cohesive understanding of the principles, methodologies, and techniques underpinning requirements engineering. Such a framework would not only facilitate the systematic development and management of requirements but also enable practitioners to navigate the complexities and challenges inherent in modern software development projects. Theoretical frameworks play a crucial role in guiding research, practice, and education in various disciplines, providing a conceptual structure for organizing knowledge, defining key concepts, and establishing relationships between different elements. In the context of requirements engineering, a theoretical framework serves as a lens through which practitioners and researchers can interpret, analyze, and evaluate the multifaceted aspects of requirements engineering processes and activities.

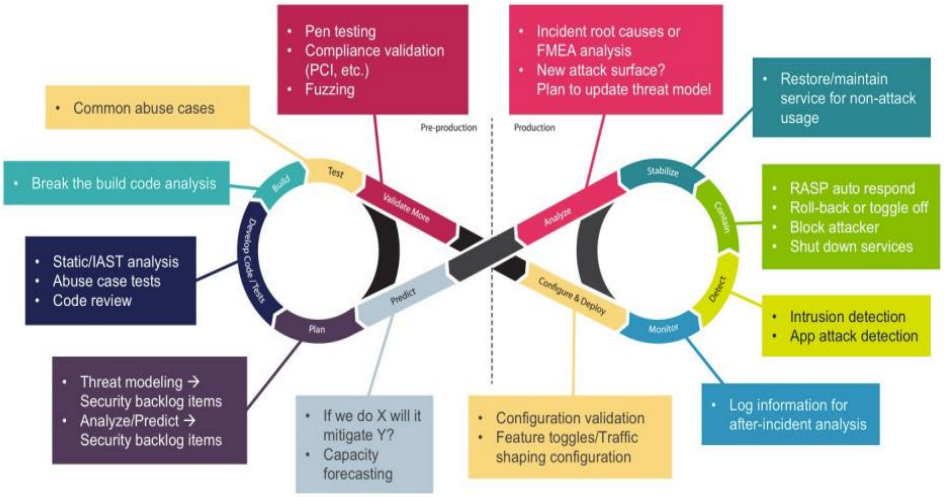

Figure 1: DevSecOps according to Larry Maccherone

This paper aims to address this need by presenting a comprehensive theoretical framework for requirements engineering. By synthesizing existing literature, industry best practices, and empirical evidence, the framework seeks to provide a unified and structured approach to understanding requirements engineering, encompassing the fundamental concepts, principles, methodologies, and techniques relevant to the discipline. Through the exploration of key topics such as requirement types, elicitation techniques, modeling methodologies, traceability, and change management, this framework aims to offer valuable insights and guidance to practitioners, researchers, and educators alike, fostering a deeper understanding of the role and significance of requirements engineering in software development endeavors.

2. LITERATURE REVIEW

Requirement prioritization involves assigning importance ranks or ratings to a collection of requirements, considering various factors like goals, risks, quality, and stakeholder viewpoints. With the evolving landscape of product development, characterized by the adoption of iterative release cycles and the intensifying pressure on developers to deliver solutions rapidly, prioritizing requirements has become crucial from the outset of the systems development life cycle (SDLC). This imperative arises due to several factors. Firstly, the iterative approach to product releases necessitates a clear understanding of which requirements are essential for each iteration, ensuring that the most critical features are developed and delivered first. Secondly, the competitive nature of modern markets demands swift delivery of products to meet evolving customer demands and market trends. Prioritizing requirements allows organizations to focus their resources on developing features that provide the greatest value to customers, enhancing competitiveness and market positioning. Moreover, involving stakeholders in the prioritization process ensures alignment with business objectives and enhances stakeholder satisfaction. By prioritizing requirements early in the SDLC, organizations can mitigate risks, optimize resource allocation, and streamline development efforts, ultimately leading to more successful and timely product launches. Thus, effective requirement prioritization serves as a cornerstone for achieving project success in today's dynamic and fast-paced development environment. Some researchers undertook a comprehensive investigation involving ten different organizations to understand their current practices in defining, interpreting, analyzing, and utilizing requirements for their software systems and products. This

investigation involved conducting a series of detailed and structured interviews with practitioners from diverse backgrounds. In this paper, we present a synthesis of the insights gleaned from this field study and discuss their implications for enhancing practice. We explore potential avenues for improvement, including organizational and methodological interventions, as well as the adoption of new technologies [2]. DevSecOps represents a paradigm shift in software development, embodying the fusion of security practices within the DevOps pipeline. It addresses the inherent tension between the imperative for rapid delivery and the imperative for secure code. This philosophy transcends the reactive approach of addressing security concerns after a threat or compromise, instead advocating for a proactive stance where critical security issues are addressed at every stage of the Software Development Life Cycle (SDLC). At its core, DevSecOps emphasizes not only the incorporation of additional tools to automate security tasks but also a fundamental shift in developer mentality. By instilling a culture of security awareness and responsibility among developers, organizations can foster a collaborative environment where security is prioritized from the outset of the development process. One of the key tenets of DevSecOps is the concept of "shifting left," whereby security considerations are brought forward to earlier stages of the development cycle. This approach enables the identification and remediation of bugs and vulnerabilities at their nascent stages, significantly reducing the cost and effort required for resolution. By integrating security testing and analysis into the development pipeline, organizations can detect and address issues before they escalate, thereby enhancing the overall security posture of their software products. Currently, significant attention is directed towards DevOps, aiming to streamline software development timelines through enhanced collaboration between development and operations teams. Despite this focus, there has been minimal exploration into the role of requirements management within the DevOps framework. This study aims to bridge this gap by offering insights into best practices concerning requirements engineering in DevOps and pinpointing areas warranting further investigation. To achieve this objective, a multivocal mapping study was conducted to examine the methodologies, techniques, and tools utilized to facilitate requirements management in DevOps environments [5]. Furthermore, the automation of security processes within the DevSecOps framework not only accelerates the delivery of code but also optimizes the utilization of resources. By automating routine security tasks, such as vulnerability scanning and code analysis, organizations can free up valuable time for their security teams to focus on more complex and strategic challenges. In essence, implementing DevSecOps represents the natural evolution of DevOps practices, wherein the emphasis on speed and agility is harmoniously balanced with robust security measures. By embracing DevSecOps principles, organizations can achieve faster, safer code delivery, minimize downtime, and reduce the incidence of vulnerabilities, thus ensuring the resilience and security of their software systems in an increasingly dynamic and threat-prone environment.

3. METHODOLOGY

Conceptual Framework Identification involves a meticulous examination of various theoretical frameworks, models, and approaches proposed in the field of requirements engineering. This analysis encompasses a broad spectrum of methodologies, ranging from traditional models like the Waterfall model, V-model, and Spiral model, to contemporary frameworks such as Agile, DevOps, and Lean requirements engineering. Traditional models such as the Waterfall model follow a sequential approach to software development, where each phase is completed before moving on to the next. While this model offers a structured and systematic approach, it is criticized for its lack of flexibility and inability to accommodate changing requirements. The concept of DevOps encompasses principles, practices, tools, and architectural frameworks, notably the microservices architectural style. This architectural style shares several characteristics with DevOps approaches, particularly modularity and flexibility, which facilitate continuous change and delivery. Two experiments were conducted, one at an academic level as part of a master's program course, and the other as industrial training. Drawing from these experiments, a comparative analysis is presented along with proposals aimed at enhancing and advancing DevOps education in the future [9].

The V-model, on the other hand, emphasizes the correlation between each phase of development and its corresponding testing phase. It advocates for early testing and validation of requirements to mitigate risks associated with late-stage defects. However, like the Waterfall model, it can be rigid and less adaptable to changing project requirements. The Spiral model incorporates iterative development cycles, allowing for repeated refinement of requirements and continuous risk assessment throughout the project lifecycle. This iterative approach fosters flexibility and adaptability, making it suitable for projects with evolving or uncertain

requirements. In contrast to traditional models, contemporary frameworks like Agile, DevOps, and Lean requirements engineering prioritize customer collaboration, iterative development, and rapid delivery of value. Agile methodologies, such as Scrum and Kanban, advocate for incremental development and frequent customer feedback, enabling teams to respond quickly to changing requirements and deliver high-quality software.

DevOps emphasizes collaboration and integration between development and operations teams, aiming to automate processes and streamline the software delivery pipeline. By breaking down silos and promoting continuous integration and deployment, DevOps facilitates faster delivery of software products while ensuring reliability and stability.

Lean requirements engineering focuses on eliminating waste and optimizing value delivery by identifying and prioritizing the most essential requirements. It emphasizes lean principles such as value stream mapping, minimizing handoffs, and empowering cross-functional teams to deliver customer value efficiently.

DevOps

The surge in DevOps adoption is driven by its capacity to expedite, secure, and ensure the reliability of developing, testing, and delivering new features to users. DevOps is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) professionals [10]. It's becoming increasingly common for companies to rely on DevOps practices. Hence, we've opted to explore this phenomenon and offer a concise overview. Our aim is to demystify DevOps and its toolset for non-technical audiences while also highlighting investment opportunities for us as Venture Capital investors. This investigation will delve into the core principles of DevOps, including continuous integration, continuous delivery, and automation. We'll also examine popular DevOps tools and platforms, such as Jenkins, Docker, and Kubernetes, explaining their roles in streamlining the software development lifecycle. By presenting these concepts in an accessible manner, we aim to provide a clear understanding of DevOps and its potential impact on business efficiency and innovation. Additionally, we'll identify emerging trends and growth areas within the DevOps ecosystem, offering insights into promising investment prospects for Venture Capital funding.

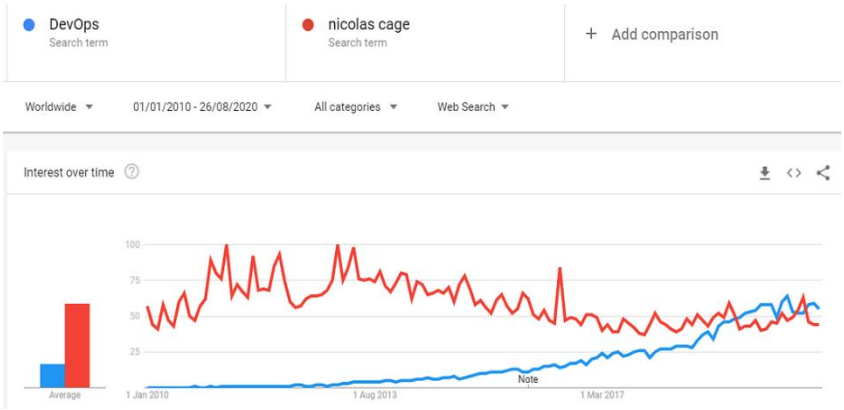

Figure 2: DevOps statistics [3]

The DevOps movement manifesto

Break down organizational barriers — Encourage collaboration between Dev and Ops teams by merging them or fostering proximity. Shared ownership and toolsets facilitate seamless cooperation.

Normalize failure — Understand that errors are inherent in development. Establish protocols for swift resolution, such as automated rollback scripts for failed deployments.

Embrace incremental change — Depart from outdated practices of infrequent, large-scale updates. Opt for smaller, more frequent deployments, following Continuous Integration/Continuous Deployment (CI/CD) principles to mitigate disruption and identify bugs promptly. DevOps, which integrates development and operations, presents a complex challenge for organizations seeking to implement it efficiently for the continuous delivery of information management systems. This paper seeks to address this complexity by conducting a systematic literature review (SLR) on DevOps. The aim of this review is to compile and analyze existing

knowledge on DevOps, providing a foundation to facilitate informed, effective, and less risky adoption of DevOps practices for information management systems [6]. Das (2024) describes set up time reduction time in an electronics industry that is our further research plan [11].

Harness automation — Replace manual, non-scalable tasks with automated processes. From package installation to monitoring and log analysis, automation enhances deployment speed, stability, and reliability, mitigating human error in repetitive tasks. Quantify everything — To validate the effectiveness of your DevOps practices, metrics are indispensable. Introduce measurement, monitoring (e.g., CPU usage), and alerting tools to track performance and detect critical service disruptions. These tools provide insights and notifications, ensuring prompt response to issues, thus reaffirming the value of adhering to the DevOps methodology.

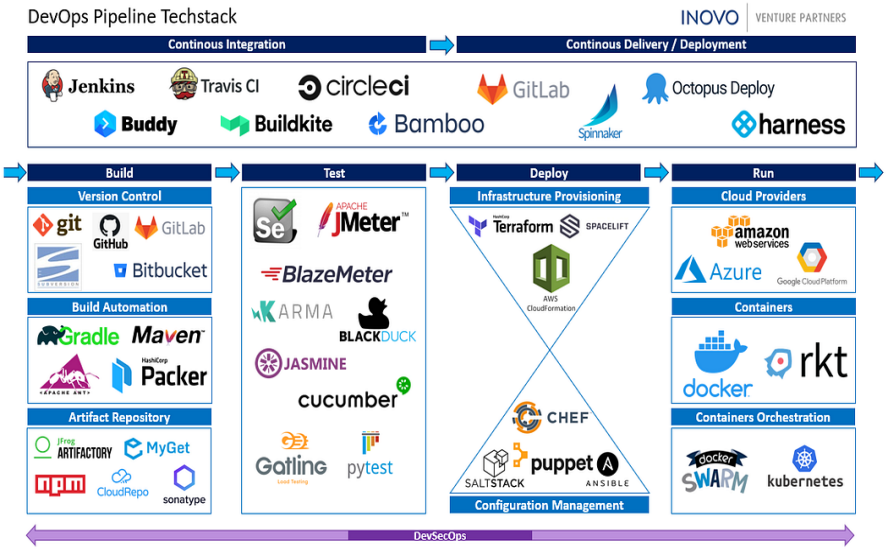

Figure 3: DevOps Pipeline Techstack.

Integration involves merging code from multiple developers into a unified codebase. The more frequent these integrations occur, the easier it is to resolve conflicts that arise, leading to a preference for integrating changes multiple times throughout the day, creating a nearly continuous process. Before code is merged into the shared repository, it undergoes a series of steps, including building, which transforms the source code into executable software, and testing to ensure its functionality. Continuous Integration (CI) tools are often paired with Continuous Delivery (CD) or Continuous Deployment (CD) tools. Continuous delivery extends the principles of Continuous Integration, ensuring that not only is the code built and tested, but it's also prepared for deployment with the simple push of a button. Continuous Deployment takes this automation further by automatically deploying the code to end-users without manual intervention. While CI/CD tools themselves may not provide substantial value independently, they serve as central hubs where various components of the development pipeline connect. Jenkins, a widely used tool in this domain, has garnered some criticism due to its complexity in setup and maintenance, despite its extensive array of plugins and integrations. As an alternative, newer entrants like Buddy and Buildkite, which recently secured a $20 million Series A funding round, offer simplified approaches and reliability, making them worth considering. In summary, CI and CD practices streamline the software development process by automating integration, testing, and deployment, with tools like Jenkins paving the way for newer, more user-friendly alternatives such as Buddy and Buildkite. DevOps, a fusion of development and operations, poses inherent complexity. Many organizations struggle to establish robust DevOps capability for consistently delivering information management systems. This paper seeks to address this challenge by conducting a systematic literature review (SLR) on DevOps. The objective is to compile and analyze insights into DevOps, leveraging the established SLR approach. Through this review, the aim is to build a knowledge repository that aids in the informed, efficient, and lower-risk adoption of DevOps for information management systems [7].

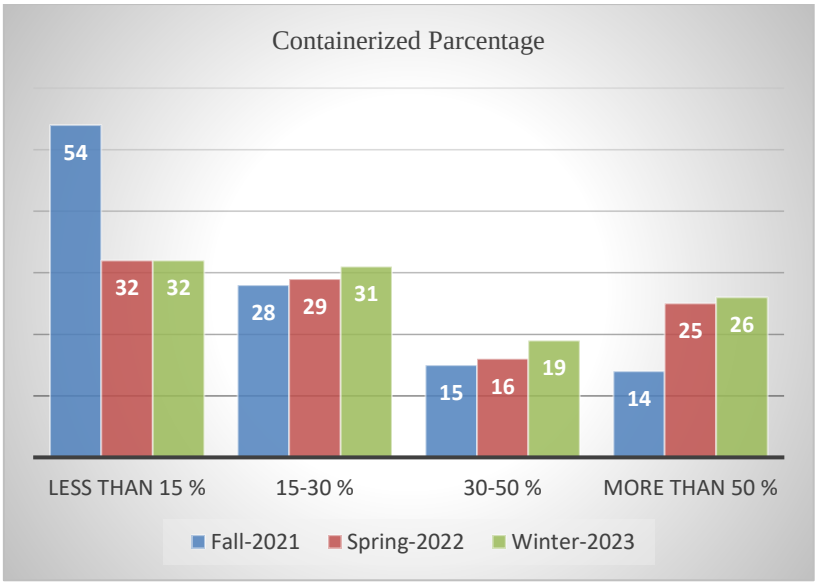

Figure 4: Containerized Percentage

When working individually, storing code in a local directory, and making copies suffices. Yet, in the context of startups and enterprises with sizable development teams, comprising hundreds or even thousands of employees, a systematic approach is imperative. Here, tools known as Version Control Systems (VCS) or Source Control Management (SCM) step in to prevent code merge conflicts and enable the retrieval of previous code versions. GitHub reigns supreme in this category, offering a far more intuitive user experience compared to older, less user-friendly solutions such as SVN. Some researchers provide a comprehensive roadmap for implementing or strengthening DevOps methodology. Additionally, it presents findings from a case study detailing the application of this roadmap within a small organization. The results indicate that the roadmap assisted the team in identifying areas for improvement in adopting a DevOps approach, enhancing their comprehension of DevOps principles, and streamlining the implementation process by furnishing a structured framework comprising tasks, roles, and performance metrics [7].

Table 1: DevSecOps according to GSA

Build		**Test**	**Deploy**	**Operations**
Plan	**Code**	**CI**	**CD**	**Security and Monitoring**
Jira	Ansible	Jenkins	Ansible	AMI
Slack	Github	Selenium	Jenkins	ClamAV
Trello	Jenkins		Terraform	CloudWatch
			CloudFormation	Nessus
				OSSEC
				SolarWinds

Build Automation tools streamline the compilation process, which essentially translates your code into functional software. Typically, you'd seek out a tool tailored to the programming language used in your project. Thankfully, there's an abundance of open-source options available. Maven stands out in this arena, boasting a market share ranging from 35% to 60%, depending on the project type. Artifact Repository management deals with the somewhat nebulous concept of artifacts, which essentially encompasses various by-products of software development, such as project source code, dependencies, or explanatory diagrams outlining software functionality. Given the variability of artifacts across software versions, effective management is crucial. JFrog stands at the forefront with its leading Artifact Repository Manager (ARM), backed by a whitepaper enumerating ten compelling reasons to opt for its services.

Testing is assuming increasing importance as DevOps practices accelerate development cycles. With rapid development comes heightened security concerns, prompting the integration of tests throughout the development pipeline rather than solely post-build. This heightened focus on testing has prompted considerable attention in recent times, warranting deeper exploration in forthcoming blog posts.

Infrastructure Provisioning (IP) tools facilitate the launch of your infrastructure, marking a significant departure from the cumbersome processes of the past. Formerly, provisioning involved the arduous task of ordering physical hardware, awaiting its delivery, and painstakingly configuring connections via wires. However, the advent of cloud computing has revolutionized this landscape, making infrastructure provisioning markedly simpler. Tools like Terraform empower you to manage your infrastructure as code (IaaC), ushering in a new era of efficiency. Building upon this foundation, Spacelift enhances collaboration among DevOps teams, bolsters security measures, introduces version control capabilities, and offers insights into infrastructure costs.

Configuration Management (CM) is an integral component of Infrastructure as Code (IaaC), ensuring that IT systems, servers, and software within your infrastructure maintain a predefined state by overseeing settings, patches, and updates. Surprisingly, 27% of DevOps teams employ custom CM solutions, while the leading commercial tool, Ansible, captures a 23% market share, particularly recommended for smaller infrastructures. Despite some overlap with Infrastructure Provisioning (IP) tools, such as Terraform, it's considered best practice to utilize separate tools for IP and CM functionalities.

4. CONCLUSION

In conclusion, this paper has endeavored to provide an extrapolatory analysis of theoretical frameworks for Requirements Engineering (RE). By delving into the foundational principles and methodologies underpinning RE, we have gained valuable insights into its multifaceted nature and the critical role it plays in software development processes. Through the exploration of various theoretical perspectives, including but not limited to, the Rational Unified Process (RUP), the V-Model, and Agile methodologies, we have elucidated the diverse approaches to eliciting, analyzing, documenting, and validating requirements. Moreover, this analysis has underscored the evolving landscape of RE, wherein emerging trends such as DevOps and Agile practices are reshaping traditional paradigms and necessitating a more dynamic and adaptive approach to requirements management. The integration of such methodologies into RE frameworks reflects a broader industry shift towards iterative development, collaboration, and rapid response to change. Furthermore, the extrapolatory nature of this analysis invites future research endeavors to delve deeper into the interplay between theoretical frameworks and practical implementation strategies within diverse organizational contexts. Additionally, the exploration of novel technologies such as artificial intelligence and machine learning presents exciting opportunities for enhancing requirements elicitation and validation processes, thereby optimizing software development outcomes.

5. FUTURE WORKS

The integration of security measures into the DevOps pipeline is gaining widespread traction, prompting a shift not only in nomenclature, from DevOps to DevSecOps, but also in mindset and the adoption of various solutions. This trend is so extensive that it warrants a dedicated blog post for comprehensive coverage. Containers are rapidly becoming ubiquitous, supplanting Virtual Machines due to their superior efficiency and cost-effectiveness. As a result, their prevalence is steadily increasing, positioning them as the new standard in software development environments.

Simplification is emerging as a key focus in DevOps pipelines, which often exhibit significant complexity, varying not only between organizations but even within a single entity. Looking ahead, there's a potential for the emergence of end-to-end tools that consolidate multiple pipeline components, streamlining setup and maintenance processes by reducing the need for extensive customization and plugin integration. A feasibility assessment can be conducted to ascertain the potential interest in the proposed MMS (Multi-Modal System). This analysis involved several steps: Conducting preliminary searches to ascertain the existence of relevant literature on the core subject of the research. Extracting keywords from the initial search results to gain insight into formulating the search criteria. Crafting the initial search string based on the identified keywords. Implementing the initial search string across bibliographic databases to determine the volume of available studies.

REFERENCES

[1]. Moisiadis, F. (2002, October). The fundamentals of prioritising requirements. In *Proceedings of the systems engineering, test and evaluation conference (SETE'2002)*.

[2]. Lubars, M., Potts, C., & Richter, C. (1993, January). A Review of the State of Practice in Requirements Modeling. In *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering* (pp. 2-14). IEEE.
[3]. https://medium.com/inside-inovo/devops-explained-venture-capital-perspective-156c5705e774.
[4]. Kabaale, E., & Kituyi, G. M. (2015). A theoretical framework for requirements engineering and process improvement in small and medium software companies. Business Process Management Journal, 21(1), 80-99.
[5]. Hernández, R., Moros, B., & Nicolás, J. (2023). Requirements management in DevOps environments: a multivocal mapping study. Requirements Engineering, 28(3), 317-346.
[6]. Qumer Gill, A., Loumish, A., Riyat, I., & Han, S. (2018). DevOps for information management systems. VINE Journal of Information and Knowledge Management Systems, 48(1), 122-139.
[7]. Laukkarinen, T., Kuusinen, K., & Mikkonen, T. (2017, May). DevOps in regulated software development: case medical devices. In 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER) (pp. 15-18). IEEE.
[8]. Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (2016, May). What is DevOps? A systematic mapping study on definitions and practices. In Proceedings of the scientific workshop proceedings of XP2016 (pp. 1-11).
[9]. Bobrov, E., Bucchiarone, A., Capozucca, A., Guelfi, N., Mazzara, M., Naumchev, A., & Safina, L. (2020). Devops and its philosophy: Education matters! Microservices: Science and Engineering, 349-361.
[10]. Bellomo, S. (2014). Architectural Implications of DevOps. Software Engineering Institute. Software Architecture: Trends and News Directions.
[11]. Das, T., (2024). SMED Techniques for Rapid Setup Time Reduction in Electronics Industry. Journal of Scientific and Engineering Research, 2024, 11(4):257-269