




Determinants of Effective Change Management for Software Deployment Projects


Abebaw Zeleke, DBA candidate

University of Maryland Global Campus, Adelphi, Maryland, United States

 <https://orcid.org/0000-0002-6351-6501>

Walter McCollum, PhD

Vice President/Senior Associate Vice Provost, Miami Dade College Online, Miami, Florida, United States

 <https://orcid.org/0000-0003-1306-9181>

Contact: abebaw@gmail.com

Abstract

Software application deployment change management is one of the emerging research themes that is gaining increased focus day by day. Our study examined the factors that affect software application deployment change management in Agile software development settings. Our study provided a systematic review and synthesized the approaches, practices, and challenges reported for adopting and implementing deployment change management. The prime objective of our study was to systematically synthesize the data extracted and formulate evidence-based practical recommendations that are influential in software deployment change management. Six research themes are proposed to evaluate the rationale of the research question. This qualitative study and systematic review explored the pertinent research articles and key findings from prominent academic databases. Based on the selected criteria, the final screening revealed 25 articles from an immense set of publications. Key findings that emerged from these publications are correlated with the six research themes: (a) timely communication with all stakeholders; (b) the reliance of deployment approaches on past experience; (c) the importance of collaboration among team members having adequate knowledge of DevOps tools; (d) the ramification of the differences among development, test, and production environments; (e) the influential areas that reap the benefits of continuous delivery and deployment; and (f) the challenges of the effective use of containerization. We also found indications of the significance of Lewin's three-step change process model in the Agile development and deployment environment. Overall, our study deepens understanding of this thriving research area and contributes to the literature on Agile deployment and the software change management process.

Keywords: *DevOps, continuous deployment, continuous improvement, change management, software deployment*

Date Submitted: June 22, 2021 | **Date Published:** November 14, 2021

Recommended Citation

Zeleke, A., & McCollum, W. (2021). Determinants of effective change management for software deployment projects. *International Journal of Applied Management and Technology*, 20, 124–12. <https://doi.org/10.5590/IJMAT.2021.20.1.07>

I would like to thank my family for supporting the work and also Dr. Walter R. McCollum, PhD—Walden University Dean of Student Affairs/Executive Director—for co-authoring this manuscript for submission to the *International Journal of Management and Technology*.

Introduction

In an increasingly globalized world, effective software deployment has gained prime significance among industries, software companies, and research communities. The innovative ways and practices in software deployment are considered key drivers in implementing a successful business for many enterprises. Software deployment management entails building, testing, and delivering reliable services to customers. Effective software applications change management and process implementation involve multiple aspects and add value to the customers. In this perspective, the Agile manifesto has become a great indicator of reliance due to frequent software updates, embedding new features, and adding security protection shields from time to time. Agile development and deployment offer short development cycles, accommodate changes at every phase of development, provide an influential interaction of users in the development cycle, and offer a platform for change management. Recent successful software change management in Agile environments has boomed in the online business of mega companies, such as Amazon, Google, and Facebook. The implementation of fast and frequent software updates, timely handling the customers' logged complaints, dealing with online bugs, and related management issues are referred to as continuous software deployment. In this regard, software development (Dev) and IT operations (Ops), are referred to as DevOps. DevOps functions have emerged to integrate software delivery organizations and cross-functional continuous collaboration teams and customers.

Many research studies have highlighted the importance of software application deployment change management in Agile software development settings. For instance, Baouya et al. (2021) noted that managing software deployments in a coordinated and planned way plays a pivotal role in an organization's application stability. The inconsistencies in software deployment management and lack of ability to cope with the changing business and users' needs by improving software capabilities affect all areas of an organization.

Continuous delivery and deployment of software solutions and services is crucial for increasing business demands for continuous improvement. A continuous change management aims to constantly keep the software up to date, which enhances the stakeholders' visibility and empowerment. The reported studies have explained that, at the core of continuous change management, lies effective continuous software deployment. However, the evidence on Agile change management to deliver autonomous software deployment projects is limited and sporadic, and it is mostly focused on a few case studies of specific organizations (Lwakatere et al., 2019). Existing studies are mainly focused on general change management of software development with specific case studies but not directly addressing the determinants of effective change management for continuous software deployment. Such an up-front understanding is eminent from the research performed by Timans et al. (2016). Their study argued a need for a specific and practical implementation of continuous improvement change management. Likewise, research by Kamal et al. (2020) highlighted that lack of proper documentation in an Agile development process increased the complexities of the software deployment change management process.

Our study investigated the various dimensions and key factors that decisively affect the areas of change management in the continuous software deployment and provided empirical means on continuous deployment and delivery. Our study is beneficial for software organizations and practitioners to get insights on software deployment and change management process within their software development and operations departments. Our results also outlined key factors that can be applied in the three stages of Lewin's unfreezing-changing-freezing change model; i.e., code development, testing, and production.

Research by Rousseau (2020) suggested the key components of a well-formulated review question, including context, intervention, mechanisms, and outcomes. Our research looked at the significant factors affecting effective change management in software deployment and explored the answer to the following research questions:

1. How does timely communication with all stakeholders affect successful software deployment?
2. Which software deployment approaches heavily rely on past project experience?
3. What is the impact of collaboration and knowledge sharing among existing team members and those with relevant experience in new DevOps tools for effective change management?
4. Which areas in the development, testing, and production environment (hardware and software) reap the benefits of continuous delivery and deployment?
5. Which elements and dependencies between applications limit the adoption of continuous deployment?
6. What are the dimensions and challenges of effective use of containerization?

Given the nature of the study and the objectives, a qualitative research methodology was adopted. From the preliminary review of 370 research articles, we focused on a total of 25 articles that pertained to software deployments and change management practices. The selected articles also included some case studies on software deployment and proposed constructive recommendations. Some of the focused studies were related to the effective change management beyond the scope of software deployment. Despite the importance of the presented topic, to the best of our knowledge, there are no research studies that went beyond the explanation of the software deployment knowledge management and its tools or DevOps scenarios. The limited studies and lack of specific focus on global continuous Agile software deployment challenges have prompted the need for this research (Efe & Demirors, 2019). Prototyping methods and iterative life cycle models were developed and applied, but the results were not as expected. Our report aims to bridge the gap in this important topic.

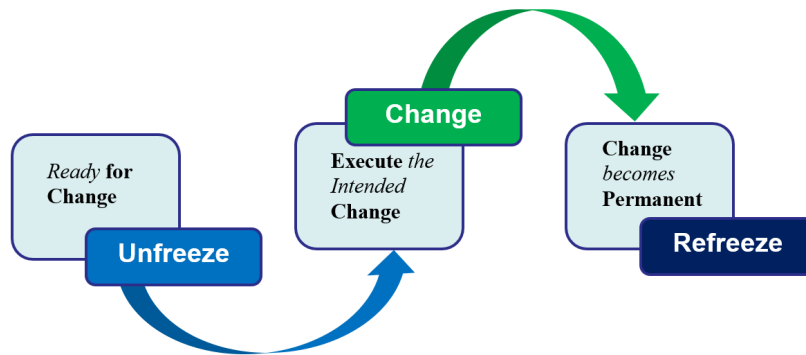
Our research was structured to (a) outline a brief introduction and importance of our research study, which was reported in the preceding section; (b) present the conceptual framework and literature review in the next section; (c) describe the research methods applied, main aspects, and a detailed description of the qualitative analysis in this study; (d) discuss the implications for software deployment practice; and (e) conclude with limitations and (f) suggestions for future research.

Literature Review and Conceptual Framework

Due to the growing importance of software deployment management in Agile environments, an increasing amount of literature is being reported describing approaches, innovative tools, and technologies (such as big data, cloud computing, and Internet of Things [IoTs]), up-to-date practices, and challenges under diverse scenarios. However, global continuous Agile software deployment challenges are still underexplored. The faster-paced software development and deployment are linked with some stressing factors, such as iterative development cycles, prioritizing new releases, accelerated productivity, and prompt coordination with customers. In this scenario, predicting and diagnosing the shortfalls of any accomplished tasks and deliverables become ambiguous and rely only on self-evaluation practices. The self-evaluation practices involve systematic appraisal of all factors and variables that are associated with systems-level outputs. Shogren et al. (2018) argued that, for evaluating the optimal deployment management, the individual-level outcomes metered and valued are important for the next execution. An exhaustive body of literature is available that supports the software deployment outcomes by using logical models. These models facilitate the alignment of support delivery development, implementation, and evaluation.

In our present study, we have used a three-step change model. This model explains a theoretical conceptual framework that was initially proposed by Kurt Lewin (Lewin, 1947) and later improved by Schein (1996/1999). The model is shown in Figure 1. This framework supports the whole system architecture and examines the interdependencies within the system. The three-step framework involves unfreezing, changing, and then freezing the model. Rajan & Ganesan (2017) also support Lewin's change management framework and suggest that change management processes are a sheer necessity for organizational emancipation, sustenance, and growth and demonstrate the competence level of an organization.

Figure 1. A Conceptual Framework of Change Model (Lewin, 1947; Schein, 1999)



The dynamic nature of software application changes and its subsequent deployment scheduling affects all stakeholders. Akbar et al. (2019) mentioned that one of the most challenging issues in such projects is the dynamic nature of change process requirements. The new technological means, big data analytics, and cloud computing are providing prompt solutions to the customers' changing requirements; however, the change management process can be made more predictable if the factors affecting the change process are identified. Considering this, Lewin's change model provides an ideal framework to establish the key parameters for software change management and deployment.

Research Methodology

Our study aimed to fill the gap of the approaches, tools, challenges, and practices of software deployment change management in an Agile environment by means of a Systematic Literature Review (SLR). This SLR is adopted because it provides a detailed understanding of the challenges and helps to identify the optimum parameters and areas of improvement. The SLR is preferred for this study because it provides a comprehensive view of the efficacy of software deployment management for different types of organizations and software-intensive applications.

SLRs frequently consider framing the highest point of the Hierarchy of Evidence, especially in the applied sciences. For instance, Popay et al. (2006) explained that systematic reviews use the Hierarchy of Evidence to decide the nature of examination considered. As explained in Figure 2 (adopted from the University of Illinois), articles and well-qualified assessments form the base of the pyramid; case series and case reports come next; case-control studies are next; the cohort studies are set in the center; the Randomized Control Trials take the second position from the top; and systematic reviews are set at the highest point of the pyramid.

Figure 2. *Hierarchy of Evidence (Systematic Reviews)*

Another aspect of adopting a systematic review for this study was to examine the consequences of past investigations within a selected time. The traditional literature reviews just sum up a given point without disclosing the models used for methodological examination of the pertinent investigations. Popay et al. (2006) explained that conventional reviews frequently extract useless detail from research perspectives. The systematic reviews specifically and artfully concentrate on methodological quality and need to be straightforward. Research by Sorrell (2007) points out that systematic reviews, in general, answer micro inquiries with respect to specialized proficiency instead of large-scale strategy questions.

The area of software deployment and change management is becoming rich with publications and reports; therefore, our study has focused only on the most recent research articles, those published within the last 6 years. Dingsøyr & Lassenius (2016) highlighted that the study of Agile software application development is more reliable when a diverse number of case studies are included in the research theme, especially the studies representing the changing speed of delivery, data security, and ecosystems. Considering this, the presented review has explored the disparity in change management and software deployment by undertaking different documents, including case studies, advanced application tools, and software change management approaches. In our review, we have systematically identified and precisely reviewed 370 relevant papers and analyzed the data obtained to answer a set of research questions (described in the Introduction). The University of Maryland Global Campus (UMGC) electronic database was used in this research. This database covers more than 50 libraries, including OneSearch, Scopus, ABI/Inform, ProQuest Dissertations, and Google Scholar. Appropriate search strings, operators, and strategies were adopted to extract the precise publications from well-renowned databases. For instance, the search string used the keywords 'DevOps Software Delivery' OR 'software deployment' OR 'continuous software deployment' OR 'effective software change management' AND 'software change management' including OneSearch, Scopus, ABI/Inform, ProQuest Dissertations, and Google Scholar. We applied snowballing technique (Budgen et al., 2008) to choose the references of the selected papers and cited the relevant publication in this research. The initial search resulted in more than 370 articles from different databases. After a detailed analysis and study, 178 articles were excluded, which were not highly pertinent to the scope of the topic. From the remaining articles, 46 were screened to be relevant to the factors that affect software deployment studies. In the second phase, we applied TAPUPAS multiple methods (Pawson et al., 2003) and Weight of Evidence (WoE, Gough, 2007) to critically evaluate the selected articles. By applying these methods, only 25 articles satisfied the eligibility criteria. For the WoE

method, we used a score of 1 to 3 (1 = low and 3 = high). This is demonstrated in Appendix A. To be more precise with WoE score, an aggregated score was evaluated to estimate how the articles are rated for each measurement.

To cope with the reliability and validity issues, our presented study has applied the TAPUPAS method. The findings of this are explained and presented in Appendix B. This method critically appraises, evaluates, and systematically examines the selected articles to evaluate their trustworthiness and their relevance in the context of the research questions. To further support the reliability of selected articles, Appendix C explains the coding of 25 articles and their relation to each other. The data are analyzed as part of the process to identify key study themes by searching and identifying concepts and finding relations between them. To ensure the reliability, we have followed three complementary analyses of each article to establish how past research relates software change management and presents a comprehensive picture of software deployment in Agile environments. The reliability is enhanced through coding and comparing individual results. Moreover, consistency was checked by conducting two pilot coding rounds.

Appendix B and C indicate that the selected articles are highly reliable under the proposed criteria.

Results and Analysis

Moreover, the findings are expected to be used as guidelines for practitioners to become more aware of the approaches, tools, and challenges and to implement appropriate practices that suit their industrial arrangements. This section presents the findings to research questions based on the SLR. The SLR has identified the main determinants of effective change management in global continuous Agile software deployments. The key findings are categorized into six themes, as presented in Appendix D (CERQual). These are explained and discussed in the following sections.

Timely Communication With All Stakeholders

The SLR has identified some imperative findings, which are associated with timely communication and feedback from all stakeholders. A set of papers discussed that, when deployment preparation and postdeployment training is scheduled, there should be coordinated communication for geographically dispersed stakeholders (Anwer et al., 2019). The review also highlighted that information technology managers and stakeholders must focus on the deployment plan. Shifting or changing the initially proposed software deployment methodologies in the middle of a project resulted in added costs, hampered the project schedule, and reduced the overall quality (Gablas et al., 2018). The results also indicated that, when a project involves hybrid teams, clear communication becomes necessary to keep everyone on board while implementing the change management plan. Research by Zasa et al. (2021) advocated that hybrid teams must include internal and external stakeholders to get an appraisal and clear visibility on the project's progress and its deployment. A set of papers has indicated that, when application deployment is scheduled as a continuous deployment Agile project plan, the communication strategy can evolve as a retrospective of the previous sprint. The SLR highlighted the importance of user training on software deployment. The study by Volker and Prostean (2016) suggested that a successful software deployment and change management project is guaranteed with all stakeholders' satisfaction on the training plans. In cases where there is a lack of relevant training, software deployment projects are adversely affected, even in the presence of skillful teams (Akbar et al., 2020). Overall, the key findings on this theme indicated that timely communication and appropriate training at all levels is crucial to a successful deployment and adoption of the deployed change.

Deployment Approaches Heavily Rely on Past Project Experience

Our analysis has revealed that new IT projects often follow similar approaches as those that were adopted in accomplished projects. Extensive experience enables professionals to understand business requirements and

transform these into Agile system settings (Palacios et al., 2018). Some studies considered the preference of new approaches if the project team experience can be leveraged to the new project. However, this approach may involve a resistance to adopting new deployment approaches (Tüzün et al., 2019). Our review has identified that both business and IT stakeholders involved in the change management process rely on established approaches rather than newly invented methods (Jayatilleke et al., 2018). The findings revealed that previous experience is coupled with meeting tight project timelines, efficient use of development resources, and preference to adopt existing deployment approaches.

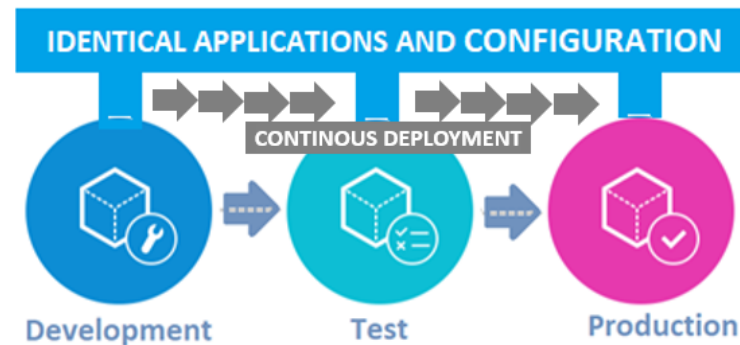
Collaboration and Knowledge Sharing Among Existing Team Members and Those With Relevant Experience in New DevOps Tools

Our analysis identified the practices that are common in successfully adopting and implementing software deployment. A set of papers argued that achieving real benefits of successful practices required developers and testers to be more responsible in the production environment to fix problems that appear after deployment. Akbar et al. (2019) supported this approach by arguing that, during the real production environment, silo work arrangements are broken and developers foresee the issues with IT operations, and deployment teams realize the limitations of IT operations while dealing with technical constraints. The results of this SLR indicated that DevOps has a significant influence on the success of practicing deployment management. DevOps is a set of activities that integrates software development and IT operations. Its main goal is to shorten the development cycle and provide high-quality software delivery on a continuing basis to users. The review also indicated that DevOps is a useful addition to Agile software development. Lwakatare et al. (2019) argued that DevOps is not well understood among software practitioners and lacked top management support. Adequate support was important to transfer knowledge of development resources and new DevOps tools. Some studies have indicated that adopting DevOps was decisive for improving cycle times to deliver optimum deployment applications to production and meet overall system quality. The outcomes of this research theme reflected that knowledge sharing between development and operations played a significant role in supporting the application (Efe & Demirors, 2019; Schuh et al., 2017).

Development, Test, and Production Environment (Hardware and Software) Differences

The results of this SLR indicated that, if proper DevOps practices are not followed, then the application code deployed for testing environment for Quality Assurance may fail. The main cause of this failure was the incompatibility of the test environment adapted to the environment of the application that was initially developed (Leonardo et al., 2019). In DevOps practice, application developers usually develop tasks that are operationalized by IT staff, including server management (Ali, 2021). To overcome these issues, the testing process is performed repeatedly. The developers write test cases during the development environment. This is the idea as explained in Figure 3.

Figure 3. Continuous Deployment Requirement to Have Identical Development, Test, and Production Servers.



This coherence provides efficient working and stability when the application code is tested and deployed during a development and test environment. Sailer and Petric (2019) suggested that coherence is the main factor that contributes to the successful execution of DevOps. It is reported that identical hardware and software setups among development, testing, and production successfully accept all deployment functional requirements (Rodríguez et al., 2017). These include servers, coding platform, tester bridge, etc.

Dependencies Among Applications Limited the Adoption of Continuous Deployment

Some papers defined the concept of continuous deployment. The review revealed that IT organizations are focused on the Continuous Delivery Practice where a development application is set, ready to be deployed to production when needed. The study by Shahin et al. (2017) elaborated that, in conventional deployment practices, no automation and dependency were associated with systems and applications. A manual verification and coordination were done to check the compliance of systems and applications. In Continuous Deployment (CD) approach, however, deployment steps are fully scripted, and all conditional logic is handled by automated processes. A CD framework produced a more structured and managed environment for Agile development. The main limitations of a CD are knowing all dependencies for an automation task (Arulkumar & Lathamaju, 2019; Luz et al., 2019; Morris et al., 2017). The dependency with hardware and compatibility with multiple versions posed a major challenge for an automatic deployment of software into customer environments.

Challenges of Effective Use of Containerization

We found several studies discussing that inappropriate workstations and application architectures created hindrance in a smooth transition toward CD practices. The highly coupled architectures can cause severe challenges for deployment and change management systems. An application developed on a local computer or cloud environment may take an enormous amount of time and setup when deployed through a different platform. Parra et al. (2018) indicated an application developed in a complex development environment may require similar setup when the application moves to a test environment. To avoid this issue, containers are introduced with all embedded dependencies for coding, testing, and deployment. Containerization enables packaging an application with self-contained units, such as Docker and Kubernetes. Investigations by Zhang et al. (2018) showed that 45.8% of the respondents have changed from one form of the container flow to another and centralized logging allows containers to share information with an entire set of all components. Among the popular containers, Docker runs on a single node, whereas Kubernetes is designed to run across a cluster. Because the centralized logging in a container is sustained for only a short time, the log messages are susceptible to being lost when a container is redeployed (Poniszewska-Marańda et al., 2021). The SLR

highlighted the importance of identifying the appropriate type of container for the long-term setup. An organization must perform a comparative analysis to find the optimal container tools and workflow approaches (Timans et al., 2016).

Discussion and Conclusions

Our work has presented a rigorous analysis of software deployment challenges and systematic synthesis of change management from selected publications through a SLR. All important approaches, tools, challenges, and practices identified through the SLR were discussed in the form of key findings. Overall, this study provided insights into software development, deployment, and change management issues. The results of the study showed that there is a positive correlation between software deployment and change management if all the dependencies are known to stakeholders. The results of the study showed that software application development managers and stakeholders must consider the key factors that affect the successful execution of continuous software deployment. The three-step process we discussed must be practiced during the change management process. The research identifies some challenges that are associated with environments. For instance, with a layered architecture, the complexity of software at several levels, including many modules or packages, made it difficult to adopt a change (Stojanov et al., 2018). In contrast, in data-driven software applications, the applied methods and tools can reduce the application complexity and simplify the management of changes.

Our study illustrated the key change management practices that lead to efficient development, testing, delivery, and operations. We also highlighted the importance of new methods, concepts, and procedures, like DevOps. The study manifested that a change (unfreezing) triggered through a change system management allowed the organization to reevaluate it and take remedial actions for future occurrences. Once the unfreezing was done, the next step was to ensure cognitive redefinition, restructuring, and learning (changing) by involving all stakeholders on a timely basis. A clear and collaborative communication among all stakeholders was important in this three-step change management cycle. Once the change was tested and implemented, its status was changed to permanent (freezing).

The SLR revealed some imperative factors that were not properly valued during the conventional deploy, test, and production environment. For instance, the successful implementation of software deployment changes was also associated with compatibility of hardware and software configurations and system architecture. Differences among development, testing, and production impact the integration and software deployment process. Due to the scope of the study and limited availability of published studies on this topic, only 25 articles satisfied the proposed criteria. However, from the findings of the selected articles, all the research questions were answered convincingly.

Limitations

Our research has undertaken the case studies of organizations that have openly shared their software deployment challenges and experiences in the form of publications. Some organizations have not shared the challenges with their software deployment process due to the proprietary nature of the software development and delivery business. Considering this, it can be viewed that a biased sample has been selected that does not cover all CD circumstances included in the software change management process. However, the selected sample provided adequate external resources and databases within the specified timeframe.

Implications for Practice

Our findings are directly relevant to the IT sector that implements continuous software development and deployment. The practice of software change management keeps evolving with time due to the dynamic nature of information technology, business, and the stakes of customers involved in this process. Continuous

software change delivery can be achieved when all the challenges for software change management are addressed.

Future Research

The preceding sections have highlighted the importance of adopting identical environments for successful software change management during all phases, starting from the development and following through until production and the quality assurance process. However, our study has identified that only limited publications are available that explain the key factors for the entire change management process, including automating build tests. To minimize the impact of manual deployment build test errors, we intend to undertake future research studies focusing on automated build, test errors, the impact of code repository choice, and deployment tests.

References

*Denotes articles included in the systematic review (SLR).

- *Akbar, M. A., Naveed, W., Alsanad, A. A., Alsuwaidan, L., Alsanad, A., Gumaei, A., Shafiq, M., & Riaz, M. T. (2020). Requirements change management challenges of global software development: An empirical investigation. *IEEE Access*, 8, 203070–203085. <https://doi.org/10.1109/ACCESS.2020.3035829>
- *Akbar, M. A., Sang, J., Nasrullah, Khan, A. A., Mahmood, S., Qadri, S. F., Hu, H., & Xiang, H. (2019). Success factors influencing requirements change management process in global software development. *Journal of Computer Languages*, 51, 112–130. <https://doi.org/10.1016/j.cola.2018.12.005>
- Ali, B. J., & Anwar, G. (2021). The mediation role of change management in employee development. *International Journal of English Literature and Social Sciences*, 6(2), 361–374.
- *Anwer, S., Wen, L., Wang, Z., & Mahmood, S. (2019). Comparative analysis of requirement change management challenges between in-house and global software development: Findings of literature and industry survey. *IEEE Access*, 7, 116585–116611. <https://doi.org/10.1109/ACCESS.2019.2936664>
- Arulkumar, V., & Lathamaju, R. (2019). Start to finish automation achieve on cloud with build channel: By DevOps method. *Procedia Computer Science*, 165, 399–405. <https://doi.org/10.1016/j.procs.2020.01.032>
- Baouya, A., Mohamed, O. A., Ouchani, S., & Bennouar, D. (2021). Reliability-driven automotive software deployment based on a parametrizable probabilistic model checking. *Expert Systems With Applications*, 174, 114572. <https://doi.org/10.1016/j.eswa.2021.114572>
- Budgen, D., Turner, M., Brereton, & Kitchenham, B. (2008). *Using mapping studies in software engineering*. Proceedings of the 20th Annual Meeting of the Psychology of Programming Interest Group (PPIG), 2008, pp. 195–204.
- Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, 77, 56–60. <https://doi.org/10.1016/j.infsof.2016.04.018>
- *Efe, P., & Demirors, O. (2019). A change management model and its application in software development projects. *Computer Standards & Interfaces*, 66, 103353. <https://doi.org/10.1016/j.csi.2019.04.012>
- *Gablas, B., Ruzicky, E., & Ondrouchova, M. (2018). The change in management style during the course of a project from the classical to the Agile approach. *Journal of Competitiveness*, 10(4), 38–53. <https://doi.org/10.7441/joc.2018.04.03>
- Gough, D. (2007). Weight of evidence: A framework for the appraisal of the quality and relevance of evidence. *Research papers in education*, 22(2), 213–228. <https://doi.org/10.1080/02671520701296189>
- *Jayatilleke, S., Lai, R., & Reed, K. (2018). Managing software requirements changes through change specification and classification. *Computer Science and Information Systems*, 15(2), 321–346. <https://doi.org/10.2298/CSIS161130041J>
- Kamal, T., Zhang, Q., Akbar, M. A., Shafiq, M., Gumaei, A., & Alsanad, A. (2020). Identification and prioritization of Agile requirements change management success factors in the domain of global software development. *IEEE Access*, 8, 44714–44726. <https://doi.org/10.1109/ACCESS.2020.2976723>

- Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies, *Information and Software Technology*, *11*, 217-230.
- Lewin, K. (1947). Frontiers in group dynamics: Concept, method and reality in social science; social equilibria and social change. *Human Relations*, *1*(1), 5–41. <https://doi.org/10.1177/001872674700100103%20>
- Leite, L., Rocha, C., Kon, F., Milojcic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. *ACM Computing Surveys (CSUR)*, *52*(6), 1-35.
- Luz, W. P., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the real world: A theory, a model, and a case study. *Journal of Systems and Software*, *157*, 110384.
- *Morris, D., Voutsinas, S., Hambly, N. C., & Mann, R. G. (2017). Use of Docker for deployment and testing of astronomy software. *Astronomy and Computing*, *20*, 105–119. <https://doi.org/10.1016/j.ascom.2017.07.004>
- *Palacios, R. C., Fernandes, E., Soto-Acosta, P., & Larrucea, X. (2018). A case analysis of enabling continuous software deployment through knowledge management. *International Journal of Information Management*, *40*, 186–189. <https://doi.org/10.1016/j.ijinfomgt.2017.11.005>
- *Parra, P., da Silva, A., Polo, Ó. R., & Sánchez, S. (2018). Agile deployment and code coverage testing metrics of the boot software on-board Solar Orbiter's Energetic Particle Detector. *Acta Astronautica*, *143*, 203–211. <https://doi.org/10.1016/j.actaastro.2017.11.037>
- Pawson, R., Boaz, A., Grayson, L., Long, A. & Barnes, C. (2003). Types and quality of knowledge in social care. Knowledge review 3. London Social Care Institute of Excellence. <https://www.scie.org.uk/publications/knowledgereviews/kro3.asp>
- Poniszewska-Marańda, A., Czechowska, E., & Chen, Y.-S. (2021). Kubernetes cluster for automating software production environment. *Sensors*, *21*(5), 1910. <https://doi.org/10.3390/s21051910>
- Popay, J., Roberts, H., Sowden, A., Petticrew, M., Arai, L., Rodgers, M., Britten, N., Roen K., & Duffy, S. (2006). Guidance on the conduct of narrative synthesis in systematic reviews. A product from the *ESRC methods programme* Version, 1, b92.
- Rajan, R., & Ganesan, R. (2017). A critical analysis of John P. Kotter's change management framework. *Asian Journal of Research in Business Economics and Management*, *7*(7), 181–203. <https://doi.org/10.5958/2249-7307.2017.00106.2>
- *Rodríguez, P., Haghightkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., Karvonen, T., Kuvaja, P., Verner, J. M., & Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. *The Journal of Systems and Software*, *123*, 263–291. <https://doi.org/10.1016/j.jss.2015.12.015>
- Rousseau, D. M. (2020). The realist rationality of evidence-based management. *Academy of Management Learning & Education*, *19*(3), 415–424. <https://doi.org/10.5465/amle.2020.0050>
- *Sailer, A., & Petric, M. (2019). Automation and testing for simplified software deployment. *EPJ Web of Conferences*, *214*, 1–7. <https://doi.org/10.1051/epjconf/201921405019>
- Schein, E. H. (1996/1999). Kurt Lewin's change theory in the field and in the classroom: Notes toward a model of managed learning. *Reflections*, *1*(1): 59-74. <https://doi.org/10.1162/15241739957028>
- Shogren, K. A., Schalock, R. L., & Luckasson, R. (2018). The use of a context-based change model to unfreeze the status quo and drive valued outcomes. *Journal of Policy and Practice in Intellectual Disabilities*, *15*(2), 101–109. <https://doi.org/10.1111/jppi.12233>

- Stojanov, Z., Dobrilovic, D., & Stojanov, J. (2018). Extending data-driven model of software with software change request service. *Enterprise Information Systems*, 12(8/9), 982–1006. <https://doi.org/10.1080/17517575.2018.1445296>
- Sorrell, S. (2007). The Rebound Effect: an assessment of the evidence for economy-wide energy savings from improved energy efficiency.
- Schuh, G., Anderl, R., Gausemeier, J., ten Hompel, M., & Wahlster, W. (2017). Industrie 4.0 maturity index. Managing the digital transformation of companies. Munich: Herbert Utz.
- Shahin, M., M. Babar, A., & Zhu, L. (2016). The intersection of continuous deployment and architecting process: Practitioners' perspectives, *ESEM'16, September 8–9, 2016*, Ciudad Real, Spain. <https://doi.org/10.1145/2961111.2962587>
- Smite, D., Moe, N.B., Sablis, A., Wohlin, C. (2017). Software teams and their knowledge networks in large-scale software development. *Information and Software Technology*, 86, 71-86.
- *Timans, W., Ahaus, K., van Solingen, R., Kumar, M., & Antony, J. (2016). Implementation of continuous improvement based on Lean Six Sigma in small- and medium-sized enterprises. *Total Quality Management & Business Excellence*, 27(3/4), 309–324. <https://doi.org/10.1080/14783363.2014.980140>
- *Tüzün, E., Tekinerdogan, B., Macit, Y., & İnce, K. (2019). Adopting integrated application lifecycle management within a large-scale software company: An action research approach. *The Journal of Systems and Software*, 149, 63–82. <https://doi.org/10.1016/j.jss.2018.11.021>
- Volker, S., & Prostean, G. (2016). Research of automotive change management and combined risk-management models. *Procedia—Social and Behavioral Sciences*, 221, 395–404. <https://doi.org/10.1016/j.sbspro.2016.05.129>
- *Zasa, F. P., Patrucco, A., & Pellizzoni, E. (2021). Managing the hybrid organization: How can Agile and traditional project management coexist? *Research Technology Management*, 64(1), 54–63. <https://doi.org/10.1080/08956308.2021.1843331>
- *Zhang, Y., Vasilescu, B., Wang, H., & Filkov, V. (2018, October). *One size does not fit all: An empirical study of containerized continuous deployment workflows*. In Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 295–306).

Appendix A

Weight of Evidence

Study	Author & Year	Coherence	Appropriateness	Relevance	Average
1	Akbar et al. (2019)	3	3	3	3.00
2	Ali (2021)	3	3	3	3.00
3	Arulkumar & Lathamaju (2019)	3	3	3	3.00
4	Palacios et al. (2018)	3	3	3	3.00
5	Efe & Demirors (2019)	2	1	1	1.67
6	Gablas et al. (2018)	3	3	3	3.00
7	Jayatileke et al. (2018)	3	3	3	3.00
8	Leonardo et al. (2019)	3	3	3	3.00
9	Luz et al. (2019)	3	2	1	1.33
10	Lwakatare et al. (2019)	3	3	3	3.00
11	Morris et al. (2017)	3	3	3	3.00
12	Akbar et al. (2020)	3	2	2	2.33
13	Parra et al. (2018)	3	3	3	3.00
14	Poniszewska-Marańda et al. (2021)	3	3	3	3.00
15	Rodríguez et al. (2017)	3	3	3	3.00
16	Sailer & Petric (2019)	3	3	3	3.00
17	Anwer et al. (2019)	3	3	3	3.00
18	Schuh et al. (2017)	3	2	2	2.33
19	Shahin et al. (2017)	3	3	3	3.00
20	Šmite et al. (2021)	3	3	3	3.00
21	Timans et al. (2016)	3	3	1	2.33
22	Tüzün et al. (2019)	3	3	3	3.00
23	Volker & Prosteian (2016)	3	3	3	3.00
24	Zasa et al. (2021)	3	3	3	3.00
25	Zhang et al. (2018)	3	3	3	3.00

Note: No explicit evidence = 0, Methodology briefly explained = 1. Considerable complementary works written = 2, Extensive justification and supporting body of knowledge = 3. High WoE (2.5 to 3.0), Medium (1.5 to 2.5), or Low (0 to 1.5) for each article. Adapted from Gough (2007).

Appendix B

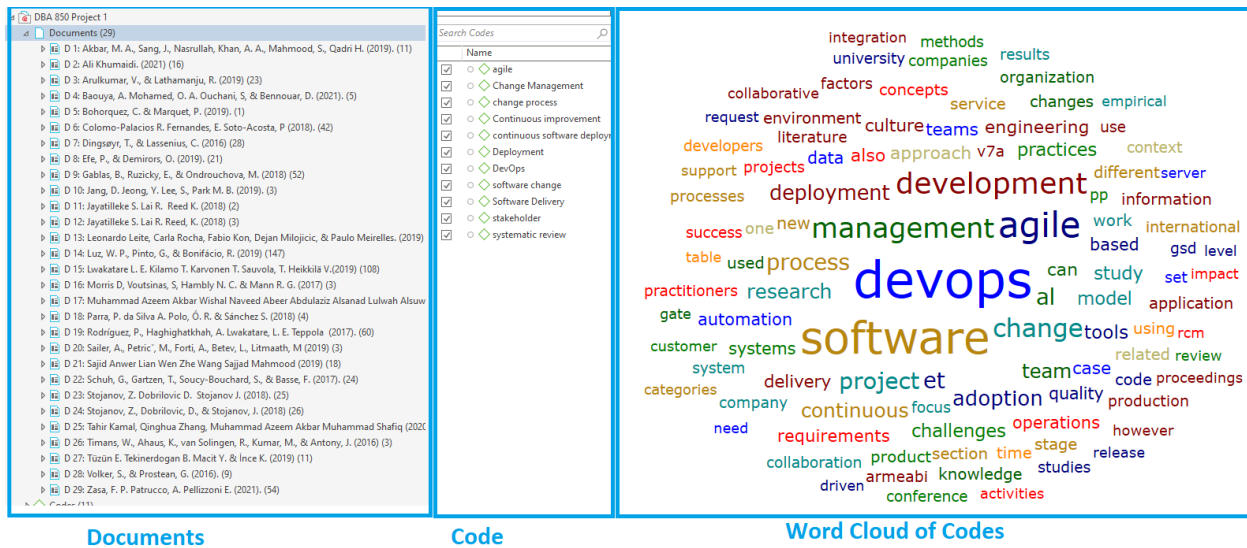
TAPUPAS

Study	Author & Year	Transparency	Accuracy	Purposivity	Utility	Proprietary	Accessibility	Specificity	Average
1	Akbar et al. (2019)	3	3	3	3	3	3	3	3.00
2	Ali (2021)	3	3	3	3	3	3	3	3.00
3	Arulkumar & Lathamaju (2019)	3	3	3	3	3	3	3	3.00
4	Palacios et al. (2018)	3	3	3	3	3	3	3	3.00
5	Efe & Demirors (2019)	2	1	2	1	2	3	2	1.86
6	Gablas et al. (2018)	3	3	3	3	3	3	3	3.00
7	Jayatileke et al. (2018)	3	3	3	3	3	3	3	3.00
8	Leonardo et al. (2019)	3	3	3	3	3	3	3	3.00
9	Luz et al. (2019)	3	2	2	3	2	3	3	1.86
10	Lwakatare et al. (2019)	3	3	3	3	3	3	3	3.00
11	Morris et al. (2017)	3	3	3	3	3	3	3	3.00
12	Akbar et al. (2020)	3	2	2	1	2	3	3	2.29
13	Parra et al. (2018)	3	3	3	3	3	3	3	3.00
14	Poniszewska-Marańda et al. (2021)	3	3	3	3	3	3	3	3.00
15	Rodríguez et al. (2017)	3	3	3	3	3	3	3	3.00
16	Sailer & Petric (2019)	3	3	3	2	2	3	3	2.71
17	Anwar et al. (2019)	3	3	3	3	3	3	3	3.00
18	Schuh et al. (2017)	3	2	2	1	2	3	3	2.29
19	Shahin et al. (2017)	3	3	3	3	3	3	3	3.00
20	Šmite et al. (2021)	3	3	3	3	3	3	3	3.00
21	Timans et al. (2016)	3	3	1	1	2	3	3	2.29
22	Tüzün et al. (2019)	3	3	3	3	3	3	3	3.00
23	Volker & Prosteian (2016)	3	3	3	3	3	3	3	3.00
24	Zasa et al. (2021)	3	3	3	3	3	3	3	3.00
25	Zhang et al. (2018)	3	3	3	3	3	3	3	3.00

Note: Note: Scoring: 3 = Highest standards met, 2 = Most standards met and 1 = Some standards Adapted from Pawson et al. (2003).

Appendix C

Word Cloud of Codes



Appendix D

CERQual

Summary of review finding	Studies contributing	Methodological limitations	Coherence	Adequacy	Relevance	CERQual assessment
Timely communication with all stakeholders.	6, 12, 17, 23, 24	12 used survey data from a total of 31 RCM challenges. Others were qualitative review software deployment and change management practices.	No concern.	Minor concern on 12 since it only included a total of 31 RCM challenges.	Minor concern on 12 regarding relevance since the study focuses on software development rather than software deployment factors, which is the focus of this study.	All studies were detailed and to the point to address the research question. All studies were detailed and especially study 4 demonstrated about orgs that are trying to embrace DevOps principles by using a widespread of knowledge-based tools. Indeed, the results from this study show that DevOps is more a cultural shift for IT than a process tools shift. All studies were adequately researched and analyzed. Collaboration related software engineering projects. Study 10 findings
Deployment approaches heavily rely on past project experience.	4, 7, 22	None.	No concern.	No concern.	No concern.	All studies were adequately researched and analyzed. Collaboration related software engineering projects. Study 10 findings
Collaboration and knowledge sharing between existing team members and those with relevant experience in new DevOps tools.	1, 5, 10, 18, 20	5, 18, and 20 embrace the change and use it as an opportunity but did not demonstrate tools that exist for plan-driven project management, specifically for	No concern.	No concern.	No concern.	All studies were adequately researched and analyzed. Collaboration related software engineering projects. Study 10 findings

Development, test, and production environment (hardware and software)	2, 8, 15, 16,	<p>depicting the status quantitatively and establishing future estimates. Overall, 1 and 10 provided best of both worlds traditional project management methods, tools, and techniques. 1, 15, and 16 demonstrated a balanced qualitative approach and 8 indicated limitations when building software for government since it involves more bureaucratic processes; requirements and prioritization can often change due to political reasons.</p>	No concern.	No concern.	No concern.	<p>show toolchain use and support for the activities of the deployment pipeline in all cases.</p>
Dependencies between applications limited the adoption of continuous deployment.	3, 9, 11, 19	<p>Study 9 authors pointed out that the first four concepts are related to the CAMS framework, proposed by Willis (2010). Studies 11 and 19 conclude that there is a great opportunity for empirical researchers to study organizations experimenting with DevOps.</p>	No concern.	<p>Minor concern on 9 since the authors pointed out that the first four concepts are related to the CAMS framework, proposed by Willis (2010).</p>	No concern.	<p>All studies were detailed and to the point to address the research question, but study 8 compared building for the government and addressed bureaucratic processes, requirements, and prioritization. All studies were detailed and to the point to address the research question. The authors pointed out that the first four concepts are related to the CAMS framework, proposed by Willis (2010). The paper concludes that there is a great opportunity for empirical researchers to study organizations experimenting with DevOps.</p>

Challenges of effective use of containerization.	13, 14, 21, 25	Study 13 focuses on embedded software development, which is commonly accepted that the use of virtual platforms is essential, especially for hardware-dependent software development. 14, 21, and 25 also complement 13 to justify the challenges and benefit of monetarization by sampling case studies.	No concern.	No concern.	No concern.	The research studies 13, 14, 21, and 25 demonstrated the dependencies of software development on hardware availability and facilitate the use of Agile methodologies. All studies were detailed and to the point to address the research question.
--	----------------	---	-------------	-------------	-------------	--

Note: Adapted from Lewin et al. (1947).



IJAMT

The *International Journal of Applied Management and Technology (IJAMT)*, sponsored by Walden University's College of Management and Technology, is a peer-reviewed, online journal that addresses contemporary national and international issues related to management and technology.