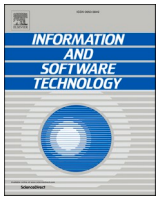
Information and Software Technology 147 (2022) 106894

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Toward successful DevSecOps in software development organizations: A decision-making framework

Muhammad Azeem Akbar [a,*], Kari Smolander [a], Sajjad Mahmood [b,c], Ahmed Alsanad [d]

[a] Software Engineering, LUT University, Lappeenranta, 53851, Finland
[b] Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
[c] Interdisciplinary Research Center for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
[d] STC's Artificial Intelligence Chair, Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia

ARTICLE INFO

Keywords:
DevOps
DevSecOps
Challenges
Multivocal literature review
Fuzzy analytical hierarchy process

ABSTRACT

Context: Development and Operations (DevOps) is a methodology that aims to establish collaboration between programmers and operators to automate the continuous delivery of new software to reduce the development life cycle and produce quality software. Development, Security, and Operations (DevSecOps) is developing the DevOps concept, which integrates security methods into a DevOps process. DevSecOps is a software development process where security is built in to ensure application confidentiality, integrity, and availability.
Objective: This paper aims to identify and prioritize the challenges associated with implementing the DevSecOps process.
Method: We performed a multivocal literature review (MLR) and conducted a questionnaire-based survey to identify challenges associated with DevSecOps-based projects. Moreover, interpretive structure modeling (ISM) was applied to study the relationships among the core categories of the challenges. Finally, we used the fuzzy technique for order preference by similarity to an ideal solution (TOPSIS) to prioritize the identified challenges associated with DevSecOps projects.
Results: We identified 18 challenges for the DevSecOps process and mapped them to 10 core categories. The ISM results indicate that the "standards" category has the most decisive influence on the other nine core categories of the identified challenges. Moreover, the fuzzy TOPSIS indicates that "lack of secure coding standards," "lack of automated testing tools for security in DevOps," and "ignorance in static testing for security due to lack of knowledge" are the highest priority challenges for the DevSecOps paradigm.
Conclusion: Organizations using DevOps should consider the identified challenges in developing secure software.

1. Introduction

DevOps is making a significant inroad into a range of IT organizations. Early-stage startups to "unicorns" like Facebook, Amazon, etc., are adopting DevOps somewhere in their organization. For example, at Flickr, "the close communication and collaboration between the development and operations teams enhanced the release time of code tenfold [1]. Similarly, Rembetsy and McDonnell [2] reported EtsDesy's success story of transferring to the DevOps culture. Erich et al. [3] claimed that most organizations are keen to initiate the DevOps program based on its known" benefits.

DevOps focuses on rapid software development and delivery through agile practices to improve collaboration between development and operation teams to reduce inconsistencies between development, operations, and releases. In this paper, we adopt Leite et al. [1] definition as follows: "DevOps is a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software versions while guaranteeing their correctness and reliability." The development team, "customers, operations, and quality assurance stakeholders collaborate to continuously deliver a software product, take advantage of market opportunities, and reduce the time needed to consider customer feedback [4,5].

Recently, DevSecOps evolved from the DevOps model as software development teams realized the importance of addressing security

* Corresponding author.
E-mail addresses: azeem.akbar@lut.fi (M.A. Akbar), kari.smolander@lut.fi (K. Smolander), smahmood@kfupm.edu.sa (S. Mahmood), aasanad@ksu.edu.sa (A. Alsanad).

https://doi.org/10.1016/j.infsof.2022.106894
Received 8 October 2021; Received in revised form 17 February 2022; Accepted 21 February 2022
Available online 27 February 2022

concerns early in the development cycle. DevSecOps integrates security management throughout the development process to coordinate activities among the trio of development, operations, and security teams [5,6]. DevSecOps strives to ensure that software applications are secure before being delivered to the user and are continuously secure during application updates. In this paper, we adopted the definition as "DevSecOps is a model on integrating the software development and operational process considering security activities: requirements, design, coding, testing, delivery, deployment and incident response"[7]. Mature DevOps practices are constantly testing, deploying, and validating that software meets every requirement and allows for fast recovery in the event of a problem. Hence, we can easily say, "DevSecOps is DevOps done right"[7].

Despite the importance of DevSecOps in software industry, little research has been done to understand the challenges faced by practitioners. Moreover, no empirical study has been conducted to prioritize and analyze the inter-relationships between the identified challenges. This paper aims to identify and prioritize challenges associated with the implementation of DevSecOps. In this study, we aim (1) to explore the DevSecOps challenges reported in the multivocal literature and industry practices, (2) to study the relationship relationships between core categories of the identified challenges, and (3) to prioritize the identified challenges concerning to their importance for DevSecOps. Prioritization of the challenging factors assists industry practitioners and academic researchers to focus on the most critical areas of the DevSecOps process. Understanding the DevSecOps challenging factors helps the industry experts develop new and effective strategies for the successful execution of DevSecOps based projects. Moreover, the detailed review and analysis of DevSecOps process challenges assist the researchers in considering the most critical challenges in their future research. We address the following research questions in this study:

[RQ1]: "What are the challenges, as identified in the literature and industry, related to DevSecOps projects?"

[RQ2]: "What are the relationships between core categories of the identified challenges?"

[RQ3]: "What is the prioritization-based taxonomy of the challenges related to DevSecOps projects?"

The rest of the paper is organized as study background is given in Section 2. Research methodology is discussed in Section 3. Results and analysis of this study are discussed in Section 4. Section 5 contains the summary of the research findings. Implications of the study results are discussed in Section 6; study limitations are expressed in Section 7, and the conclusion is provided in Section 8.

2. Background

The goal of DevOps is to facilitate collaboration between development and operations teams so that practitioners can deploy and update software in an efficient way with minimal disruption to the user experience [8,9]. In a DevOps model, both development and operation teams have independent tools, processes, and knowledge bases [8,10]. DevOps allows the development team to add new features into production continuously, and the operational teams operate the latest version to maintain project quality and other non-functional requirements [11,12]. In a nutshell, DevOps aims to define a route that allows the development and operation stakeholders to collaboratively send new features to the production environment [13].

DevSecOps automates security integration at all phases of the DevOps life cycle, from initial design to integration, testing, deployment, and delivery. However, integration of security throughout the DevOps process has its own set of challenges [6]. One of such challenges is making security adapt to the DevOps processes, which will mean the security methods need to be highly agile, and the procedures must be accepted and understood by the security, development, and operations teams [14]. Another challenge is how the organization will start the DevSecOps practices by readily adopting the change in skills, tools, standards, and processes to successfully implement security into their culture [15]. Additionally, automation and being conversant with relevant tools and technology is an ongoing challenge for DevSecOps projects. The dynamic nature of the environment implies that the necessary security functionality should be ready for use in tools that work on the right platforms [16,17].

McKay [18] suggested a list of recommendations to assist the integration of security teams in the DevOps process as follows: (1) the challenge should be first identified; (2) security teams should be exposed to DevOps early; (3) a centralized solution should be adopted; (4) the security team should transform their security practices into templates rather than reviewing new environments; and (5) the software defects should be spotted as many as possible using attempts to break the software. Similarly, Goldschmidt and McKinnon [14] recommended three layers to be taken into consideration when creating a DevSecOps environment in a cloud-based deployment as follows: (1) the infrastructure layer: there should be the inclusion of security expertise in the configuration and deployment of infrastructure to reduce impending risks; (2) the platform layer: changes should not be done to the platform directly, instead of on the settings that contain the automation and control; and (3) the application layer: the focus should be given to some processes to improve application security. Such processes are continuous integration, configuration management, validating security capabilities before enabling communication among multiple systems, understanding log files and monitoring output in a production system, cross-team communication, and collaboration.

Besides the significance of integrating security into the DevOps model, few studies has been done to understand the factors that could negatively influence the DevSecOps processs. The increasing importance of understanding and addressing security concerns during DevOps projects motivated us to develop a taxonomy of DevSecOps challenges. The taxonomy will be based on the challenges identified via a multivocal literature review and questionnaire survey conducted with the industry experts. The key objective of the industrial study is to know the perceptions and opinions of experts working with the DevSecOps paradigm. Quantitative prediction is challenging for humans (DevSecOps team), as they could more perfectly convey the feelings verbally (qualitatively). However, based on the expert's opinions, the ranking of challenging factors is complex due to the uncertainties and vagueness of the expert's opinions. Therefore, in this study, we use the fuzzy TOPSIS process to translate the qualitative prediction of the DevSecOps practitioners into quantitative prioritization values. It is a well-known approach usually used to evaluate multi-criteria decision-making problems."

3. Research design

This study explores the challenges of DevSecOps reported in state-of-the-art literature and industry practices. Multivocal literature review and questionnaire survey approaches were used to address the study's objectives. In addition, Interpretive Structure Modeling (ISM) was applied to study the relationships between core categories of the DevSecOps challenges. At the final stage, fuzzy TOPSIS was used to prioritize the identified challenges regarding their criticalities for DevSecOps projects. The details are as follows:

3.1. Multivocal literature review

The MLR study includes a literature review from formal academic publications and the gray literature. Formal academic literature consists of the pre-reviewed published research articles in conferences, journals, and workshops. The gray literature includes white papers, organization reports, blog posts, webpages, magazines, etc. To effectively perform the literature search and data extraction process, we have developed the step-by-step protocols for MLR following the guidelines of Garousi et al. [19] Fig. 1. presents an overview of the MLR process.

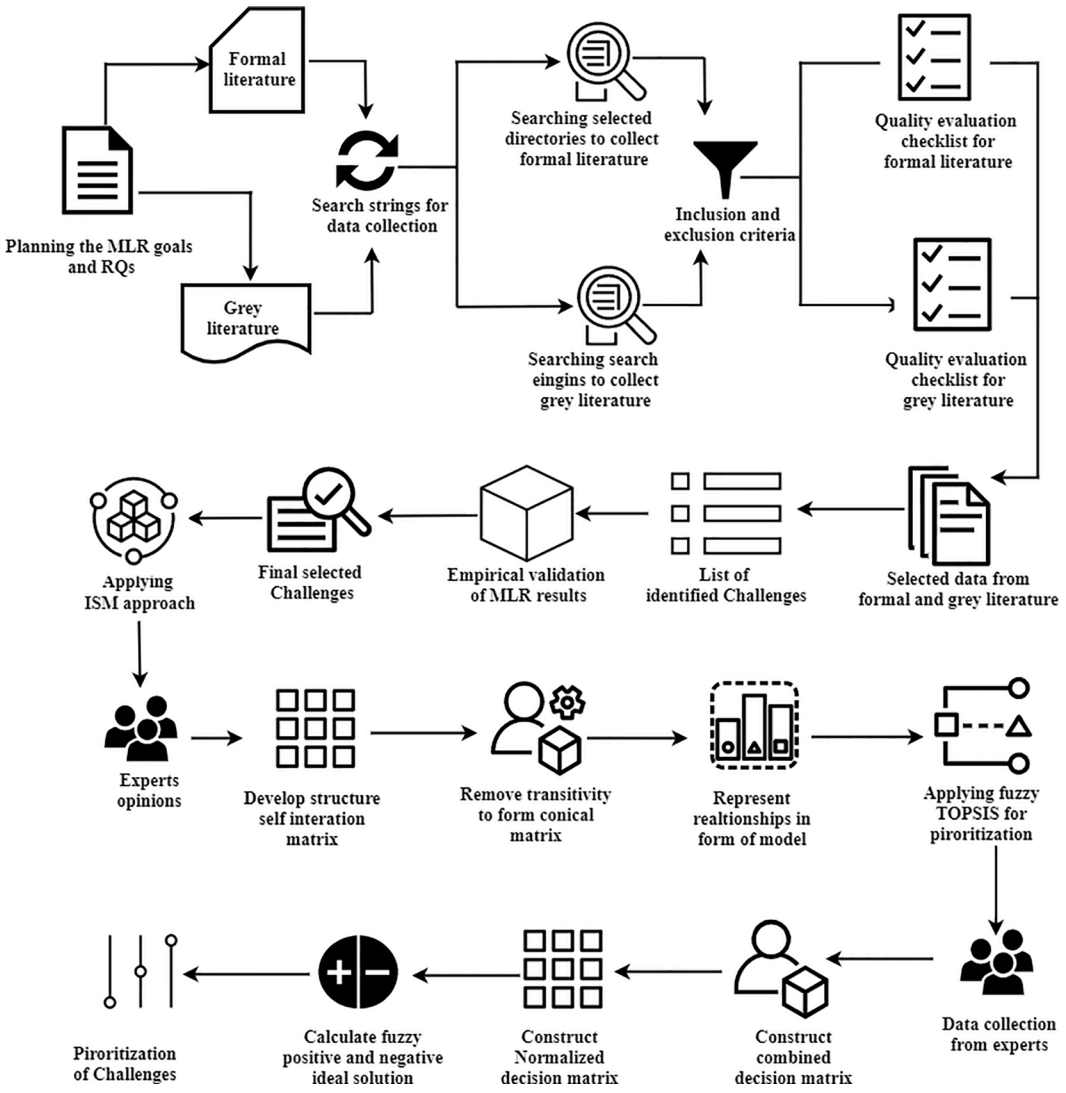


Fig. 1. Research approach.

3.2. Literature search process

We have used the automated search process to cover the vast body of literature. To execute the automated search process, we have developed the search string considering the scope of MLR (i.e., RQ1), using the PICO criteria [20] ("Population, Intervention, Comparison, Outcomes"). The search string was developed using keywords and their alternative reported in the literature [21,22]. The keywords (Table 1) were concatenated using the Boolean operator "OR" and "AND" and executed the complete search sting on scientific digital repositories, as well as Google Scholar and Google. The literature was extracted in two steps; firstly, the developed search string was executed on the scientific databases and collected literature. Secondly, we apply the search on the Google search engine and collect the gray literature.

Table 1
search string.

Related topics	Used keywords and alternatives
SS1 (Outcomes)	("barriers" OR "obstacles" OR "hurdles" OR "difficulties" OR "impediments" OR "hindrance" OR "challenges" OR "limitations")
SS2 (Intervention)	"devsecops" OR "secdevops" OR "devopssec" OR "secops" OR ("secur" AND "devops") OR ("secure" AND "continuous software engineering") OR ("secure" AND "continuous delivery") OR ("secure" AND "continuous deployment") OR ("secure" AND "continuous integration")
SS3 (Population)	Software development community
SS4 (Experimental)	("grounded theory" "interviews" "case studies," "questionnaire survey", "theoretical studies", "content analyses", "action research").
"Final search string=	(SS1) AND (SS2) AND (SS3) AND (SS4)"

3.3. Collection and selection of formal literature

To collect the formal literature, we have executed the developed search string on six scientific digital repositories by considering the suggestion of Chen et al. [23], Niazi et al. [21], Afzal et al. [24]. The selected repositories are presented in Fig. 2. The search process of the selected databases has different syntax; so, to collect the potential and a good representative set of data, we format the search string according to the requirements of digital repositories.

After collecting the literature in response to the execution of search string on the selected databases, we initially performed the literature inclusion and exclusions process; using the following protocols [21,25,26]:

- The chosen piece of literature should be in English.

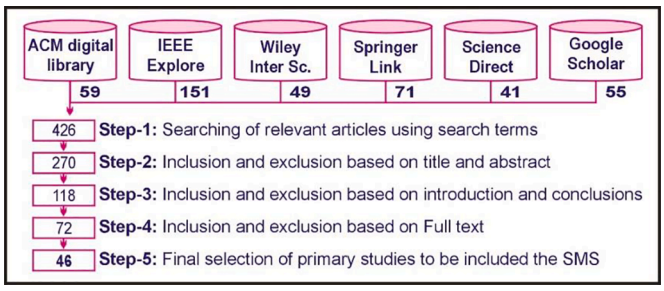


Fig. 2. Refinement of formal studies.

- The collected literature should be a journal paper, conference proceedings, or a book chapter.
- The selected work of literature should contain a detailed description of DevSecOps.
- The article should describe the challenging factors of the DevSecOps paradigm.
- The findings of the studies should be based on empirical evaluation.

Moreover, we have considered the following protocols to exclude the extracted literature:

- If two "studies found with similar nature of findings or from the same research group, then we only considered the final version."
- Short papers which lack empirical evaluation.
- A study that does not provide detailed information about the challenges of DevSecOps.

To collect the most potential literature according to the research question of this study, we further refine the selected formal studies by applying the tollgate approach proposed by Jabbari et al. [10]. The tollgate approach consists of five phases and assists in refining the collected literature concerning the study's research question (Fig. 2). Initially, we have collected the 426 studies in response to the execution of developed search strings on the selected scientific digital repositories. To carry the toll-gating process first, three authors of this study continuously participated, and authors number 4 and 5 validated each step of the tollgate approach. By carefully performing the tollgate approach steps, we have finally selected the 46 studies for the data extraction process.

The quality evaluation (QE) of the selected primary studies was performed to assess the suitability concerning the research objective. The QE score expresses the degree of appropriateness of the selected literature to answer the proposed research questions. To evaluate the quality of the selected primary studies, we followed the existing systematic literature review studies [27–29]. A formating checklist was developed for quality evaluation, as present in Table 2. The checklist includes three questions, and each question was assessed using the Likert scale given in Table 2. Using the questions of the format list, all the selected studies were evaluated, and the score is presented in Appendix-A.

Table 2
quality assessment criteria formal primary studies for.

QA Questions	Checklist Questions	Likert scale
QA1	"Does the used research approach address the research questions?"	"Yes=1, Partial=0.5, NO=0″
QA2	"Does the study discuss the challenges of DevSecOps?"	"Yes=1, Partial=0.5, NO=0″
QA3	"Are the identified results related to the justification of the research questions?"	"Yes=1, Partial=0.5, NO=0″

3.4. Collection and selection of gray literature

We explored the eight search engines to collect the most related and a good representative set of gray literature (Fig. 3). We executed the search string given in the research design section to extract the data from selected search engines.

By following the guidelines of Garousi et al. [19], we developed the inclusion and exclusion criteria that include: "the published data is publicly available on the Internet, the published data is a standalone publication written under a real name or the name of an organization, the publication content is original, and it should be more than 250 words, and the published data should cover the DevSecOps paradigm". For excluding the data, we used the following protocol: "data collected from un-authentic sources, data not in the English language, the data that do not provide detailed information about the indicated challenges."

By executing the developed search string on the selected search engines, we first collected 228 potential studies from the gray literature. For the further refinement of gray literature, considering the study's research question, we have used the five steps of the tollgate approach [24], as presented in Fig. 3. Finally, 41 pieces of gray literature were selected for the data extraction process.

The quality evaluation process was conducted to check the suitability of the collected data concerning the study's research question. The quality evaluation criteria were developed considering the recommendations of Garousi et al. [19] and given in Table 3. The final score of gray literature collected through web engines is presented in Appendix-A.

3.5. Data extraction and synthesis

The coding scheme approach [30] was applied to analyze the final collected literature (formal and gray literature). The coding scheme is an effective way to explore the analytical data to identify the specific information related to the study subject. All the collected data were reviewed, and the findings from the selected literature were extracted and labeled. The frequency of all the challenges in both data sets was also recorded. In the initial phase, 41 categories (ideas, statements, etc.) of the challenges were recorded using a Microsoft Excel Sheet after carefully examining each piece of the collected data. We systematically compared similar challenges in the second phase and merged the related challenges into 18 final challenge categories.

We performed an inter-rater reliability test to assess and remove the researcher bias [31]. Three independent experts were requested to participate in the inter-rater reliability test. They selected ten literature pieces (5 formal studies and five gray literature). All the phases of data extraction and literature selection were performed. By considering the findings of the independent experts, Kendall's nonparametric coefficient

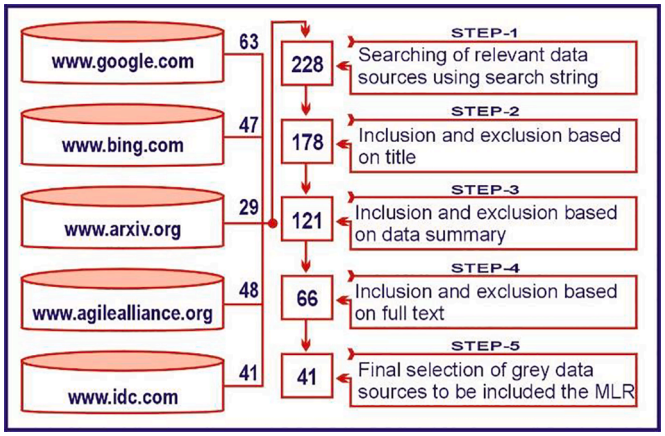


Fig. 3. Steps of tollgate gray literature.

Table 3
Quality assessment criteria for gray literature.

Criteria type	Questions of QA	Likert scale
Authority of the producer	"C1: Is the publishing organization reputable?"	"Yes=1, Partial=0.5, NO=0″
	"C2: Is an individual author associated with a reputable organization?"	"Yes=1, Partial=0.5, NO=0″
Methodology	"C3: Does the source have a clearly stated aim?"	"Yes=1, Partial=0.5, NO=0″
	"C4: Does the source have a stated methodology?"	"Yes=1, Partial=0.5, NO=0″
	"C5: Is the source supported by authoritative, contemporary references?"	"Yes=1, Partial=0.5, NO=0″
	"C6: Are any limits clearly stated?"	"Yes=1, Partial=0.5, NO=0″
	"C7: Does the work cover a specific question?"	"Yes=1, Partial=0.5, NO=0″
	"C8: Does the work refer to a particular population or case?"	"Yes=1, Partial=0.5, NO=0″
Objectivity	"C9: Does the work seem to be balanced in the presentation?"	"Yes=1, Partial=0.5, NO=0″
	"C10: Is the statement in the sources as objective as possible? Or is the statement a subjective opinion?"	"Yes=1, Partial=0.5, NO=0″
	"C11: Are the conclusions supported by the data?"	"Yes=1, Partial=0.5, NO=0″
Date	"C12: Does the item have a clearly stated date?"	"Yes=1, Partial=0.5, NO=0″
Novelty	"C13: Does it enrich or add something unique to the research?"	"Yes=1, Partial=0.5, NO=0″
	"C14: Does it strengthen or refute a current position?"	"Yes=1, Partial=0.5, NO=0″

of concordance (W) was calculated to determine the inter-rater agreement between the results of the researcher and the independent experts. The value of $W = 1$ indicates the perfect agreement, and $W = 0$ indicates the complete disagreement between the findings of researchers and independent experts. The results $W = 0.82$, $p = 0.004$ show the favorable agreement between the findings of researchers and independent experts. We used the following code to determine the "Kendall's nonparametric coefficient of a concordance" (W), and the detailed results are given in Table 4.

```
"library(DescTools)
DevSecOps <- data.frame
(external_ex = c(3,3,3,4,3,4,2,3,3,2),
external_ex = (3,4,3,5,3,4,2,4,3,3),
external_ex3=(3,3,4,4,3,4,1,3,3,3)
authors_abc = (2,3,3,4,2,4,3,3,2,3)
)
KendallW(DevSecOps, TRUE)
KendallW(DevSecOps, TRUE, test = TRUE)
)
KendallW(t(d.att[, -1]), test = TRUE)
friedman.test(y = as.matrix(d.att[,-1]), groups = d.att$id)"
```

Table 4
Kendall's coefficient of concordance test.

Data Set	Kendall Chi-Squared	df	Subjects	Raters	*p*-value	W
DevSecOps	35.344	14	10	3	0.004267	0.8246765

3.6. Industrial empirical study

The questionnaire survey approach effectively collects the data from a large and targeted population [32]. The following steps were adopted to perform the empirical investigation:

3.7. Survey questionnaire development and pilot evaluation

The evaluation of literature findings with industry experts is vital to understand practitioners. A total of 18 challenges were enlisted by conducting the literature review process. We developed the questionnaire into two parts to verify the literature survey findings and collect the additional challenges associated with DevSecOps projects. The first part is closed-ended, and it contains the list of 18 challenges identified during the literature review study. To get the opinions of survey participants, we used the five-point Likert scale that includes ("strongly agree, agree, neutral, disagree, and strongly disagree"). The neutral option is vital to collect unbiased data on the Likert scale. It gives a chance to the survey respondents to mark 'neutral' if participants don't have enough knowledge about a particular factor. The second part of the survey questionnaire is open-ended. This section enables the survey participants to add any success factors not enlisted in the closed-ended part.

3.8. Pilot assessment

To check the readability and understandability of the requested queries and questionnaire variables, we performed the pilot assessment of the questionnaire. The pilot assessment was performed with two postdoctoral researchers ("Nanjing University of Aeronautics and Astronautics, China" and "Griffith University, Australia") and three industrial experts (Virtual force-Pakistan, Integrio Systems, Canada and, Startup Development House, Poland).

All participants were requested to fill the questionnaire and give recommendations to improve the understand-ability of the survey questionnaire. The participants recommended putting all variables in a tabular form and adding more questions regarding the information of survey participants, i.e., organization size, organization type, and nature of participant organizations' business. We addressed all their recommendations, and the final survey instrument was used to perform the data collection process. The sample of the used questionnaire is given in Appendix B.

3.9. Data sources of industrial survey study

To approach the potential population, an online link to the survey questionnaire was developed using the services of Google form (docs.google.com/forms). To invite the targeted survey participants, we sent the invitation via personal Email, organizational Email, and LinkedIn (www.linkedIn.com). Furthermore, to collect the representative data sample, we used the concept of the snowball technique [27,33,34]. In snowball sampling, the participants are requested to share the survey questionnaire with their in-contact practitioners and R&D members. Snowball sampling is an efficient and cost-effective way to collect data from the physically distributed population. Snowball sampling is also beneficial for collecting data from a large and dispersed targeted population [38]. The data collection process was carried out from January 2021 to April 2021. During the data collection process, 121 responses were collected. We manually reviewed all the responses, and eight responses were found uncompleted; the final 113 responses were used for further analysis.

3.10. Analysis of the empirical data

We used the frequency analysis approach to analyze the data, which is suitable for analyzing the descriptive types of data [35]. It "comparatively analyzes the survey variables and computes the agreement level between the survey participants based on the selected Likert scale. The frequency analysis method has also been adopted in other empirical studies in software engineering [21,36-39].

3.11. Phase 2: ISM approach

Interpretive Structure Modeling (ISM) was defined by Sage [40] as "a methodology that helps to impose order and direction on the complex relationship among factors and system that results into a holistic, systematic model." ISM is an interactive learning technique that assists in structuring the related factors directly or indirectly in a holistic model. The model gives a clear and conceptual picture in a graphical format [40,41]. ISM helps to fix the complication faced concerning the relationship among different factors; hence, ISM gives a clear understanding of such types of factor relationships. Several existing studies have used this approach to develop conceptual models for the understanding of relationships between factors (Kannan et al. [42], Sharma et al. [43], Agarwal and Vrat [44]).

Consequently, the inter-personal bias in experts' opinions might affect the results of ISM. The ISM approach does not give weights to analyze the ranking of each factor at a level. We have adopted the fuzzy TOPSIS approach in later steps to avoid this concern, prioritizing the DevSecOps challenging factors based on their relationship with ten knowledge areas of DevSecOps. The steps adopted in the ISM approach to finding the interaction between DevSecOps knowledge areas (Secure Testing, Standards, Compliance and policy, Strategy and matrices, Requirements, Training, Security feature, and design, Architecture analysis, Configuration management, and Software Environment) Fig. 1 presents the detailed steps adopted to perform the ISM approach, and the steps are considered from the study of Raj and Attri [45].

3.12. Phase 3: fuzzy TOPSIS

In 1981 the TOPSIS was developed by Hwang and Yoon [46], aiming to define the positive and negative ideal solutions. The positive ideal solution maximizes benefit criteria and minimizes the cost criteria. In contrast, the negative ideal solution maximizes the cost criteria and reduces the benefit criteria [46]. The solution is considered the best if it is near a positive ideal solution and far from a negative ideal solution.

Chen and Tsao [47] developed fuzzy TOPSIS intending to fix the multi-criteria decision-making by collecting experts' opinions on a specific subject matter.

The decision-makers mark each criterion's weights in linguistic variables, and then the responses are converted into fuzzy triangular numbers (TNFs). The TNF is useful for managing the vagueness of linguistic terms mentioned by the decision-makers [48–50]. The fuzzy TOPSIS algorithms for fixing the multi-criteria decision-making are provided below.

Step-1. Suppose the problem with m alternative and n criteria can be expressed in matrix form, where A_1, A_2,….A_m are the alternatives, E_1, E_2,…, E_n is the evaluation criteria, F_{ij} is the performance rating judged by decision-makers to the alternative F_i against the criterion E_j, and Wj of each criterion Ej represents weight.

$$E_1 E_2 \ldots E_n$$

$$D = (\mathrm{Fij})\mathrm{m} * n = \begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{matrix} \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1n} \\ A_{21} & A_{22} & \ldots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \ldots & A_{mn} \end{pmatrix} \quad (1)$$

Step-2. Assemble the alternatives and their weighted criteria according to Eq. (1). Assign the ratings to each defined criterion and their alternatives using Bozbura et al. [51] fuzzy triangular scale as shown in Table 5.

Step-3. Determine the aggregate fuzzy rating of K decision-makers for each criterion by using equations Eqs. (2) and (3).

$$\mathrm{A_{ij}} = K^{min}\{x_{ijk}\},\ b = \frac{1}{K}\sum_{k=1}^{K} y_{ijK},\ c = K^{max}\{z_{ijK}\} \quad (2)$$

$$\mathrm{W_{j1}} = K^{min}\{x_{jK1}\},\ b = \frac{1}{K}\sum_{k=1}^{K} y_{jK2},\ c = K^{max}\{z_{jK3}\} \quad (3)$$

Where $A_{ij} = (x_{ij}, y_{ij}, z_{ij})$ and $i = 1,2,3,\ldots m$ and $j = 1,2,3\ldots,n$ and weight of each criterion is calculated as

$W_j = (W_{j1}, W_{j2}, W_{j3})$.

Step-4. Calculate normalized decision matrix of 'R' using linear scale transformation. The matrix after normalization will present as:

$$\mathrm{R^{-}} = [\mathrm{rij}]_{\mathrm{m*n}} \quad (4)$$

Eqs. (5) and (6) given below are used to calculate each alternative's cost and benefit criteria.

$$\mathrm{r_{ij}} = \left(\frac{x_{ij}}{z_j^+}, \frac{y_{ij}}{z_j^+}, \frac{z_{ij}}{z_j^+}\right) \text{and } z_j^+ = \text{ maxi zij (benefit criteria)} \quad (5)$$

$$\mathrm{r_{ij}} = \left(\frac{x_j^-}{z_{ij}}, \frac{x_j^-}{y_{ij}}, \frac{x_j^-}{x_{ij}}\right) \text{ and } x_j^- = \text{ maxi lij (cost criteria)} \quad (6)$$

Step-5. Compute the weighted normalized fuzzy decision matrix V∼, by multiplying the weights of each criteria W_j with calculated values of the normalized fuzzy decision matrix.

$$\mathrm{V} \cong \left[v_{ij}\right]_{\mathrm{m*n}},\ \text{where } v_{ij} \text{ is calculated by using Eq 8} \quad (7)$$

$$v_{ij} = \mathrm{A_{ij}} * \mathrm{W_j} \quad (8)$$

Step-6. In this step, the fuzzy positive and negative ideal solution is calculated as shown in Eqs. (9) and (10).

$$\mathrm{A^{+}} = \left[v_{1,}^+ v_{j,}^+ \ldots . v_m^+\right] \quad (9)$$

$$\mathrm{A^{-}} = \left[v_{1,}^- v_{j,}^- \ldots . v_m^-\right] \quad (10)$$

The values of positive and negative ideal solutions range between [0.1].

Step-7. Compute the distance of each alternative from a positive and negative ideal solution.

$$D_i^+ = \sum_{j=1}^{n} D_v\left(v_{ij}, v_j^+\right) \quad (11)$$

$$D_i^- = \sum_{j=1}^{n} D_v\left(v_{ij}, v_j^-\right) \quad (12)$$

Where D represents the distance between two fuzzy numbers.

Step-8. To compute the ranks of alternatives CCi value is calculated by considering a fuzzy positive and negative ideal solution.

Table 5
Triangular-fuzzy scale [51].

"Linguistic terms"	Triangular Fuzzy Scale
"Just Equal (JE)"	"(1,1,1)"
"Equally Important (EI)"	"(0.5,1,1.5)"
"Weakly Important (WI)"	"(1,1.5,2)"
"Strongly More Important (SMI)"	"(1.5,2,2.5)"
"Very Strongly More Important (VSMI)"	"(2,2.5,3)"
"Absolutely more important (AMI)."	"(2.5,3,3.5)"

$$\text{CCi} = \frac{D_i^-}{D_i^+ + D_i^-} \quad (13)$$

Step-9. Define the priority-wise ranking of all alternatives according to CCi value. The higher the CCi value, the greater is the rank of that alternative.

4. Study findings

The results and analysis of the study are elaborated in the following sections.

4.1. Findings of MLR study

By executing the step-by-step protocols of MLR, we identified the list of 18 DevSecOps challenging factors that are important to address for the successful implementation of the DevSecOps process (Table 6). The identified list of challenging factors was further mapped in the core ten categories of software security [52]. This mapping is used later in the paper (Section 4.5) to develop the holistic model of DevSecOps challenges and its core categories.

The concept of a coding scheme was used to map the identified challenges into core categories (i.e., general categorization, sub-categorization, and theoretical framework) [53]. The identified challenges were labeled and grouped by the mapping team, considering the impact of each challenge on the DevSecOps process. In the mapping process, the first and third authors of this study continually participated, the second author arbitrarily involved verifying the mapping process. Various existing studies of other software engineering domains have considered these mapping practices [21,25]. Furthermore, we performed an inter-rater reliability test to verify the mapping results. We invited two external experts, and they performed the mapping steps. Considering the mapping results of study authors and external experts, we determined the non-parametric Kendall's coefficient of concordance (W), [31], and the results ($W = 0.96$) show the significant agreement between study authors and external experts. Hence, this shows that the mapping process is consistent and unbiased.

Table 6
List of challenging factors.

Sr. No	Challenges	Categories
Ch1	Lack of automated testing tools for security in DevOps	Secure Testing (C1)
Ch2	Ignorance in static testing for security due to lack of knowledge	
Ch3	Communicate security standard to the DevOps team	Standards (C2)
Ch4	Lack of secure coding standards	
Ch5	Inconsistence security policies design and performance measures	Compliance and policy (C3)
Ch6	Developers' resistance to integrating security protocols	
Ch7	Neglecting change control in security	
Ch8	Using unstable performance metrics for security evaluation	Strategy and matrices (C4)
Ch9	Translate Compliance constraints to requirements	Requirements (C5)
Ch10	Lack of software security awareness	Training (C6)
Ch11	Integrate and deliver security features	Security feature and design (C7)
Ch12	Threat modeling scalability issue	
Ch13	Perform security feature review	Architecture analysis (C8)
Ch14	Periodic penetration test for application coverage	Configuration management (C9)
Ch15	Define secure deployment parameters and configurations	
Ch16	Identify software defects found in operations monitoring and feed them back to the development	
Ch17	Use application behavior monitoring and diagnostics	Software Environment (C10)
Ch18	Use of immature automated deployment tools	

4.2. Secure testing

The testing is vital since it discovers defects/bugs before the delivery to the client, which guarantees the quality of the software. DevOps process of continuous development involves automated testing tools and know-how about security facts to valid software builds to a repository. Hsu [54] stated that most of the security problems in DevOps software development occur due to a lack of automated testing tools and ignorance in static testing for security. Two challenges are reported in the literature to perform the secure testing, i.e., Ch1 (Lack of automated testing tools for security in DevOps) and Ch2 (Ignorance in static testing for security due to lack of knowledge). The organizations must focus on Ch1 and Ch2 for secure testing process execution (Table 6).

4.3. Standards

The organization creates security standards for technologies in use. These security standards are the organization's demands to meet security policy while carrying out operations. These standards are a part of the job that must be communicated to the DevOps team while working on security [16]. The standards are deployed from the application's specific terms to secure coding standards. DevOps team's ignorance of security standards can make development tasks less actionable [6]. Under this category, we found Ch3 (Communicate security standard to DevOps team) and Ch4 (Lack of secure coding standards), pointing to the key areas that need consideration of software development organizations for the successful execution of the DevSecOps paradigm (Table 6).

4.4. Compliance and policy

The compliance and policies are contractual agreements to control service-level software risks. According to Farroha and Farroha [55], in DevOps, there is still a debate on "Inconsistence security policies design and performance measures. Sánchez-Gordón and Colomo-Palacios [56] also claim this fact as "Developers' resistance to integrating security protocols" as their main focus is to deliver the product to a customer on time. To secure software from risks, there should be proper security policies. Proper auditing is required against security policies to manage change control in an organization. The literature reported three challenging points that need to consider by an organization for compliance and policy aspects, i.e., Ch5 (Inconsistence security policies design and performance measures), Ch6 (Developers' resistance to integrating security protocols), and Ch7 (Neglecting change control in security) (Table 6).

4.5. Strategy and matrices

The strategy and matrices encompass assigning roles and responsibilities to team members, identifying security goals, estimating software budgets, and identifying matrices and gates [56]. The software development team using unstable performance metrics for security evaluation will be challenging for both the organization and security landscape [57]. The organizations need to focus on Ch8 (Using unstable performance metrics for security evaluation) to implement strategies and matrices in the DevSecOps paradigm successfully (Table 6).

4.6. Requirements

The requirements involve eliciting security requirements from the organization and building standards and controls, i.e., input validation, authentication, security guidance, etc. [58]. The DevOps team should translate compliance constraints into software requirements for each

project. This approach is helpful for an organization to codify the requirements to reusable codes or containers [57]. To address this aspect of DevSecOps, the software development organization should consider Ch9 (Translate Compliance constraints to requirements) (Table 6).

4.7. Training

Training provides awareness to team members about organization history and use of tools and the technology stack [58]. On-demand individual or group training about security awareness will support the DevOps team in maintaining the security standards of an organization [59]. To address the Ch10 (lack of software security awareness), the organizations need to arrange workshops and seminars that could be handy to educate the practitioner regarding the DevSecOps paradigm (Table 6).

4.8. Security features and design

The security features and design involve building middleware frameworks and designing usable security patterns to control security [60]. The security features must be integrated with all phases of software development to resolve challenges like threat modeling, scalability issue, etc. [61]. Ch11 (Integrate and deliver security features) and Ch12 (Threat modeling scalability issue) are the key challenges highlighted by the literature in the context of security features and design (Table 6).

4.9. Architecture analysis

Architecture analysis is required to review the performance of security features [61]. During this process, all security features are observed to study the design problems that could cause a failure in the software development process [62]. For this, literature reported Ch13 (Perform security feature review), and organizations need to consider this factor to cover the architectural-based issues of DevSecOps (Table 6).

4.10. Configuration management

Configuration management involves updating software, version control, incident handling, and defect tracking. Penetration testing must be performed to identify an application's vulnerabilities [63]. After vulnerabilities are detected, they are fed back to the development team to defect management [64]. The organizations must practice secure deployment parameters and configurations to control security operations. For configuration management, three challenges are reported in the literature, i.e., Ch14 (Periodic penetration test for application coverage), Ch15 (Define secure deployment parameters and configurations), and Ch16 (Identify software defects found in operations monitoring and feed them back to development) (Table 6).

4.11. Software environment

The software environment deals with the platform where all installation, configuration, management, monitoring, and code signing are performed [65]. The organization monitors the behavior of an application to spot attacks. Immature automated deployment tools can misguide the team about firewalls and configuring services. A proper installation guide of software with license agreements must be distributed to customers to make the software environment understood by humans and not just by machines [66]. In the software environment aspect, we identified Ch17 (Use application behavior monitoring and diagnostics) and Ch18 (Use of immature automated deployment tools) from the literature and key areas that need to focus by software organizations for DevSecOps success (Table 6).

4.12. Findings of empirical study

This study contains the summarized data results collected via a questionnaire survey study.

4.12.1. Demographic data analysis of survey participants

The first part contains the queries about the demographic information of survey participants. This information is essential to assess the effectiveness of collected data concerning the objectives. According to Patten [67], "the demographic data provides information about survey respondents and is essential for the determination of whether the participants in a particular study are a representative sample of the target population for results generalization purposes or not." Moreover, Finstad [34] mentioned that the analysis of demographic information is essential to assess the maturity level of collected data sets. Altman et al. [68] noted that the demographic data is vital to determine "what your target population is and what they are thinking about." Hence, considering the importance of demographic information of survey participants, we analyze collected data regarding the participant's organization size, designation, and experience. The analyzed results are given in the following sections

4.12.2. Respondent's designation

Finstad [34] mentioned that the importance and applicability of the factors vary concerning the expert's designations. Furthermore, Niazi et al. [21] underlined that the impact of the challenges varies concerning the participants' designation. They further mentioned that a factor could be ranked correctly if the respondents experienced that challenge. The designation of survey participants is presented in Fig. 4. The results show that most survey participants include "project managers" and "software developers."

4.12.3. Respondent's experience

We also analyzed the experience of the survey respondents. Considering the responses mentioned in the questionnaire, we determined the medium and mean (medium = 5.5 and mean 7.5). There is a significant variation in the survey participants' experiences, as presented in Fig. 5.

4.12.4. Organization size

We also collected the data from the survey participants concerning the size of their organizations. To classify the organizations, we used the definition of the Australian Bureau of Statistics [69], i.e. "(SMALL, 0–19 employees), (MEDIUM, 20–200 employees), and (LARGE, $\geq$200 employees)"[69]. Akbar et al. [70] mentioned that the survey participants' size is essential to assess their exposure and maturity levels. According to the results given in Fig. 6, 31(33%), 37 (40%), and 25 (27%) participants belong to large, medium, and small-scale organizations. There is a good mix of survey participants from all sizes of organizations.

4.12.5. Summarized results of empirical study

The empirical study was conducted to get the experts' perceptions working in the DevOps environment. To collect the opinions of experts, we developed an online survey instrument. The opinions of survey participants were collected in the form of a five-point Likert scale that includes "strongly agree, agree, strongly disagree, disagree, and neutral." We broadly classified the responses into three categories: positive ("strongly agree and agree"), negative (strongly disagree and disagree), and "neutral." The summarized result of the positive category presents the opinions of those survey participants who agree that the identified factors could negatively impact the DevSecOps practices. The negative category shows the opinions of those participants who did not consider the identified factors as a challenge for DevSecOps. Lastly, the score of the neutral category presents those respondents who are not sure about the impact of identified factors on the DevSecOps paradigm. The bold rows of Table 7 show the responses of survey participants

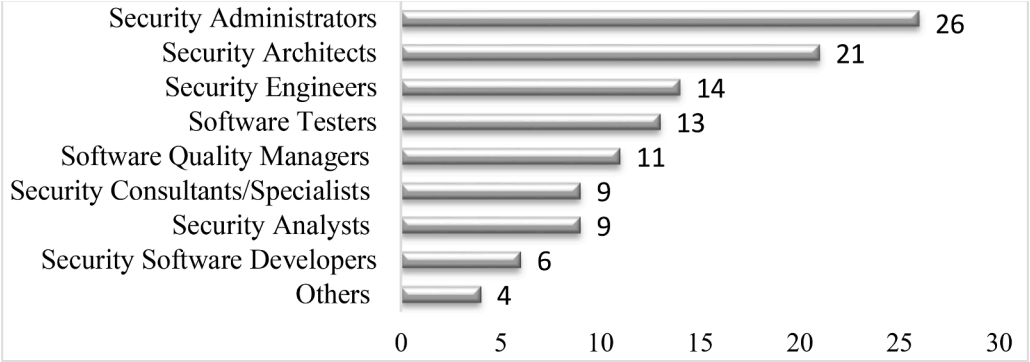


Fig. 4. Analysis of the survey participant's designations.

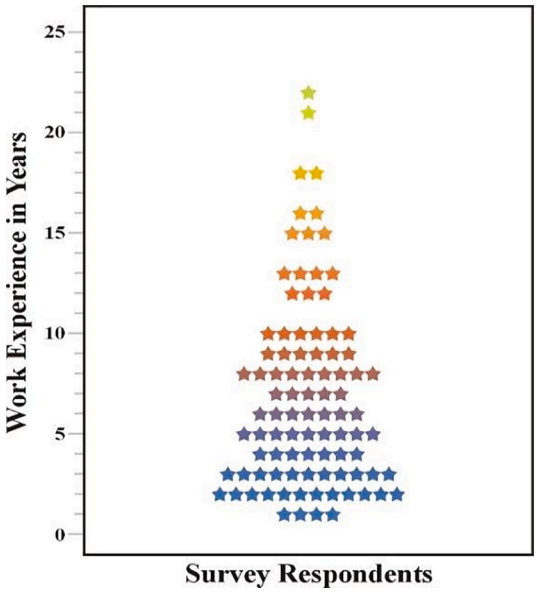


Fig. 5. Experience of survey respondents.

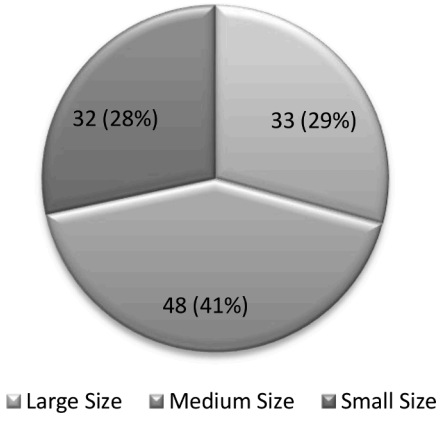


Fig. 6. Participants organizations size.

against the core categories of DevSecOps challenges. The summarized results are given in Table 7.

According to the results (Table 7), most of the survey respondents are agreed that the identified list of challenging factors negatively influenced the DevSecOps process. All the identified challenging factors scored ≥70% in the frequency analysis results.

The highest reported challenging factors were Ch1 (Lack of automated testing tools for security in DevOps, 85%) and Ch16 (Identify software defects found in operations monitoring and feed them back to development, 85%). Furthermore, Ch12 (Threat modeling scalability issues, 84%) and Ch8 (Using unstable performance metrics for security evaluation, 83%) was declared as the third and fourth most scored challenging factors for DevSecOps practices Table 7.

Based on the frequency analysis results, C2 (Standards, 90%) and C7 (Security feature and design, 89%) are declared the highest reported categories of the DevSecOps process. We also observed that C3 (Compliance and policy, 87%), C1 (Secure Testing, 86%), and C4 (Strategy and matrices, 86%) are the second most critical knowledge area of DevSecOps practices.

Considering the frequency analysis (Table 7), Ch9 (Translate Compliance constraints to requirements), Ch10(Lack of software security awareness), and Ch11 (Integrate and deliver security features) scored 13% in the negative category. This shows that 13% of the survey participants do not consider Ch9, Ch10, and Ch11 challenging for the DevSecOps process. Furthermore, Ch10 (Lack of software security awareness, 15%) and Ch13 (Perform security feature review, 15%) are declared the highest scored challenges in the neutral category. This renders that 15% of the survey participants are not sure about the impact of Ch10 and Ch13.

4.13. Results of ISM approach

The ISM approach has been used to examine the interaction between the key knowledge areas of DevSecOps challenging factors. Various researchers have applied the same approach to studying elements' contextual interaction [42–44]. Ergo, to develop the contextual interaction between the criteria, creating a structural-self-interaction matrix (SSIM) is essential, as presented in the following sections.

4.14. Structural-self-interaction matrix (SSIM)

The expert's opinions were used to apply the ISM approach for examining the contextual relationship between the key categories of DevSecOps. To get the insight of experts, we developed an experts' group to apply ISM. The participates were invited from the first survey using an invitation letter. Ten experts agreed to participate in the decision-making process. The participants are from the industry practices and R&D areas, where DevOps is prepared. A brief demographic of participants is given in Appendix D. From the expert's opinions; we developed the SSIM matrix. The sample size might limit the generalization of the study. However, we found that Kannan et al. [42] used five experts' opinions to select reverse logistic providers.

Similarly, Soni [71] established nine members to analyze the factors that caused complexities in an urban rail transit system. We further found that Attri et al. [72] used the data of five experts to determine a decision on the success factors for total productive maintenance. Ergo, considering the other existing studies, the opinions of ten experts are

Table 7
Summary of empirical investigations.

S. No.	List of Challenging Factors	Empirical Investigations ($n = 113$)							
		Positive			Negative			Neutral	
		S.A	A	%	D	S.D	%	N	%
C1	**Secure Testing**	**40**	**57**	**86**	**2**	**5**	**6**	**9**	**8**
Ch1	Lack of automated testing tools for security in DevOps	40	56	85	4	1	4	12	11
Ch2	Ignorance in static testing for security due to lack of knowledge	51	42	82	5	5	9	10	9
C2	**Standards**	**58**	**44**	**90**	**0**	**2**	**2**	**9**	**8**
Ch3	Communicate security standard to DevOps team	41	47	78	7	6	12	12	11
Ch4	Lack of secure coding standards	37	51	78	6	4	9	15	13
C3	**Compliance and policy**	**51**	**47**	**87**	**2**	**3**	**4**	**10**	**9**
Ch5	Inconsistence security polices design and performance measures	37	48	75	6	7	12	15	13
Ch6	Developers' resistance to integrate security protocols	31	61	81	3	6	8	12	11
Ch7	Neglecting change control in security	46	39	75	9	3	11	16	14
C4	**Strategy and matrices**	**40**	**57**	**86**	**2**	**6**	**7**	**8**	**7**
Ch8	Using unstable performance metrics for security evaluation	30	64	83	2	6	7	11	10
C5	**Requirements**	**41**	**54**	**84**	**3**	**6**	**8**	**9**	**8**
Ch9	Translate Compliance constraints to requirements	39	46	75	8	7	13	13	12
C6	**Training**	**39**	**48**	**77**	**6**	**8**	**12**	**12**	**11**
Ch10	Lack of software security awareness	30	51	72	10	5	13	17	15
C7	**Security feature and design**	**54**	**47**	**89**	**1**	**3**	**3**	**8**	**7**
Ch11	Integrate and deliver security features	39	46	75	8	7	13	13	12
Ch12	Threat modeling scalability issues	42	53	84	6	2	7	10	9
C8	**Architecture analysis**	**39**	**56**	**84**	**8**	**3**	**10**	**7**	**6**
Ch13	Perform security feature review	49	34	73	7	6	12	17	15
C9	**Configuration management**	**45**	**48**	**82**	**4**	**4**	**7**	**12**	**11**
Ch14	Periodic penetration test for application coverage	33	56	79	5	4	8	15	13
Ch15	Define secure deployment parameters and configurations	39	53	81	8	5	12	8	7
Ch16	Identify software defects found in operations monitoring and feed them back to development	41	55	85	5	6	10	6	5
C10	**Software Environment**	**31**	**61**	**81**	**2**	**5**	**6**	**14**	**12**
Ch17	Use application behavior monitoring and diagnostics	43	44	77	7	4	10	15	13
Ch18	Use of immature automated deployment tools	35	50	75	5	8	12	15	13

S.*A*= strongly agree, *A*=Agree, *D*=disagree, S.*D*=strongly disagree, *N*=Neutral

sufficient for building the ISM model.

The following symbols are used to indicate the direction of a relationship between the DevSecOps enabler (m and n).

(1) "V" indicates the relationship from m enabler to n enabler.
(2) "A" indicates the relationship from n enabler to m enabler.
(3) "X" when both enablers' m and n reach each other.
(4) "O" presents the situation when there is no relationship between enabler m and enabler n.

Depending on experts' opinions, we have developed SSIM presented in Table 8.

According to the results presented in Table 8, there is no relationship between C1 (Secure Testing) and C10 (Software Environment), as there's relationship is presented with 'O'. Similarly, it is noted that C2 (Standards) helps to improve the C10 (Software Environment), as 'V' denoted the relationship between both the enablers. Furthermore, it is noted that C3 (Compliance and policy) helps improve the C10 (Software Environment); because, according to the experts' opinions, C3 and C10 have 'A' types of relationship.

4.15. Reachability matrix

To develop the reachability matrix, we changed the values of V, A, X, and O in binary digits (0, 1). We consider the following protocols to develop the reachability matrix.

(1) If the value of m and n in SSIM is V, we replace it with 1; otherwise, the assigned value is 0.
(2) If the value of m and n in SSIM is A, it is replaced with 0; else, it becomes 1.
(3) If the value of m and n in SSIM is X, it is replaced with 1; and gives 1 to n and m entry.
(4) If the value of m and n in SSIM is O, it will replace with 0; for n and m, the assigned value is 0.

By applying the above-discussed protocols, the reachability matrix is presented in Table 9. The final reachability matrix was developed by determining the transitivity, as discussed in Section 3.3. The 1* value is used to incorporate the transitivity. This assists in filling the gaps in the data collected from experts while developing SSIM. The incorporation of transitivity check is given in Table 10.

Table 10 presents all criteria' driving, dependence power, and ranks. The determined driving power indicates all the criteria to achieve that DevSecOps knowledge area (criteria). The dependence power shows the criteria that may help in achieving it. This dependence and driving power will help in MICMAC analysis, where we divide the criteria into four clusters, i.e., autonomous, dependent, linkage, and independence cluster.

Table 8
SSIM matrix.

	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1
C1	O	V	V	O	V	V	X	O	V	*
C2	V	O	O	O	O	O	O	O	*	*
C3	A	O	O	O	O	O	O	*	*	*
C4	O	O	V	X	X	V	*	*	*	*
C5	O	V	O	V	V	*	*	*	*	*
C6	X	O	O	X	*	*	*	*	*	*
C7	O	X	O	*	*	*	*	*	*	*
C8	O	O	*	*	*	*	*	*	*	*
C9	V	*	*	*	*	*	*	*	*	*
C10	*	*	*	*	*	*	*	*	*	*

Table 9
Reachability matrix.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	1	1	0	1	1	1	0	1	1	0
C2	0	1	0	0	0	0	0	0	0	1
C3	0	0	1	0	0	0	0	0	0	0
C4	1	0	0	1	1	1	1	1	0	0
C5	0	0	0	0	1	1	1	0	1	0
C6	0	0	0	1	0	1	1	0	0	1
C7	0	0	0	1	0	1	1	0	1	1
C8	0	0	0	0	0	0	0	1	0	0
C9	0	0	0	0	0	0	1	0	1	1
C10	0	0	1	0	0	1	0	0	0	1

4.16. Partitioning the reachability matrix

Warfield [73] stated that "the reachability set consists of the element itself and other elements, which it may help to achieve, whereas the antecedent set consists of the element itself and other elements, which may help achieve it. After that, the intersection of these sets is derived for all the elements. The elements for which the reachability and the intersection sets are the same occupy the top level in the ISM hierarchy. The top-level element in the hierarchy would not help achieve any other element above its level. Once the top-level element is identified, it is separated from the other elements. Then, the same process is repeated to find out the elements in the next level. This process is continued until the level of each element is found. These levels help in building the diagraph and the ISM model".

In this study, we have ten criteria (DevSecOps knowledge areas), and their reachability set, antecedent set, intersection set, and levels are presented in Table 11.

4.17. Interpretation of ISM model

The final ISM model was developed based on the reachability matrix results. The inter-relationships of the criteria are presented using the arrows pointing from criterion to other. The transitivity analysis was performed to check the vagueness in the data once the diagraph was finally converted to the ISM model (Fig. 7). C2 (Standards) category stands on the top for selecting DevSecOps challenging factors. This shows C2 (Standards) is an independent area of identified DevSecOps challenges. All the other categories are dependent on C2. As Fig. 7 shows, categories of level 4, C1(Secure Testing), C4 (Strategy and matrices), C5 (Requirements), and C6 (Training) are dependent only on level 5, C2 (Standards), but all the coming areas of level 3, 2 and 1 depends on level-4, categories. The results show that C3 (Compliance and policy) depends on all other categories.

4.17.1. MICMAC analysis

MICMAC is abbreviated by matrix cross-impact matrix multiplication applied to classification. The MICMAC analysis assists in examining the key categories that drive the system. According to Attri et al. [72], the MICMAC "is an analysis to examine categories' drive power and dependence power." The enablers are classified into four clusters based on their driving and dependence power.

(1) Autonomous cluster: the category that belongs to this cluster has weak driving and dependence power. They are mostly disconnected from the system due to weak links. Hence, these categories have a minor impact on the whole system.
(2) Linkage cluster: the categories belonging to this cluster have strong driving and dependence power and affect other enablers due to strong linkage.
(3) Dependent cluster: the enablers belonging to this cluster have strong dependence power but have weak driving power.
(4) Independent cluster: the enablers belonging to this cluster have weak, dependent power but have strong driving power; they are also known as "key enablers."

4.18. Development of conical matrix

The key objective of conical matrix development is to perform the

Table 11
Leveling of final reachability matrix.

	LEVEL PARTITIONS ITERATION ONE REACHABILTY SET	Antecedent Set	Intersection set	Level
C1	1,2,4,5,6,7,8,9,10	1,4	1,4	
C2	2,3,6,10	1,2	1,2	
C3	3	2,3,7,8,10	3	1
C4	1,4,5,6,7,8,9,10	1,4,5,6,7,10	1,4,5,6,7,10	
C5	4,5,6,7,9,10	1,4,5	4,5	
C6	4,6,7,9,10	1,2,4,5,6,7,10	4,6,7,10	
C7	3,4,6,7,8,9,10	1,4,5,6,7,9,10	4,6,7,9,10	
C8	8	1,4,7,8	8	2
C9	3,7,9,10	1,4,5,6,7,9,10	7,9,10	
C10	3,4,6,7,9,10	1,2,4,5,6,7,9,10	4,6,7,9,10	
	ITERATION two			
C1	1,2,4,5,6,7,9,10	1,4	1,4	
C2	2,6,10	1,2	1,2	
C4	1,4,5,6,7,9,10	1,4,5,6,7,10	1,4,5,6,7,10	
C5	4,5,6,7,9,10	1,4,5	4,5	
C6	4,6,7,9,10	1,2,4,5,6,7,10	4,6,7,10	
C7	4,6,7,9,10	1,4,5,6,7,9,10	4,6,7,9,10	
C9	7,9,10	1,4,5,6,7,9,10	7,9,10	3
C10	4,6,7,9,10	1,2,4,5,6,7,9,10	4,6,7,9,10	
	ITERATION three			
C1	1,2,4,5,6	1,4	1,4	
C2	2,6	1,2	1,2	
C4	1,4,5,6	1,4,5,6	1,4,5,6	4
C5	4,5,6	1,4,5	4,5	
C6	4,6	1,2,4,5,6	4,6	
	ITERATION Four			
C2	2	2	2	5

Table 10
Transitivity check.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Driving power	Rank
C1	1	1	0	1	1	1	1*	1	1	1*	9	6
C2	0	1	1*	0	0	1*	0	0	0	1	4	2
C3	0	0	1	0	0	0	0	0	0	0	1	1
C4	1	0	0	1	1	1	1	1	1	1*	8	5
C5	0	0	0	1*	1	1	1	0	1	1*	6	3
C6	0	0	0	1	0	1	1	0	1*	1	6	3
C7	0	0	1*	1	0	1	1	1	1	1	7	4
C8	0	0	0	0	0	0	0	1	0	0	1	1
C9	0	0	1*	0	0	0	1	0	1	1	4	2
C10	0	0	1	1*	0	1	1*	0	1*	1	6	3
Dep power	2	2	5	6	4	7	7	4	7	8	52	
Rank	1	1	3	4	2	5	5	2	5	6		

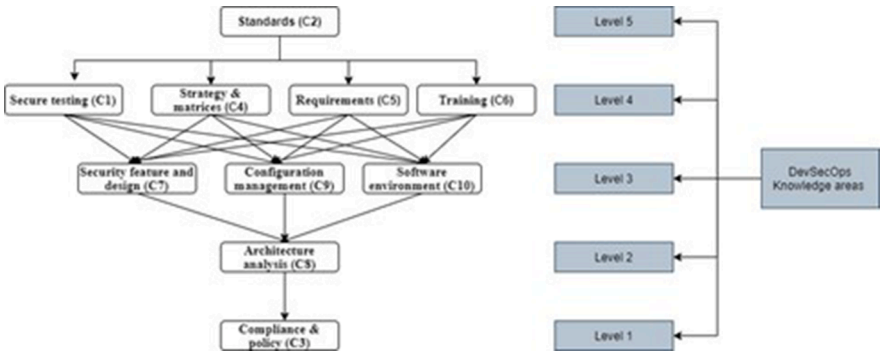


Fig. 7. Leveling of DevSecOps core categories.

MICMAC analysis. The conical matrix (Table 9) was developed considering the data given in Tables 10 and 11. Firstly, all the criteria were ordered concerning their level number (Table 11). Secondly, the values of each criterion were considered from Table 10. For example, the value of C3 across rows and columns of transitivity matrix (Table 10) shows the '0′ relationship between all other criteria, except C3. Similarly, the value of C7 shows a '1′ relationship with C3, C8, C7, C9, C10, C4, and C6; and for the rest of the criteria, the relationship of C7 is '0′. Thus, considering the same procedure (Table 12).

We considered the classification approach of Kannan et al. [42] and preset the MICMAC analysis results in Fig. 8. All the DevSecOps criteria were classified into four clusters of MICMAC analysis Fig. 8. shows the classification of DevSecOps knowledge areas into four clusters. The first cluster includes (autonomous enablers), the second cluster (dependent enablers), third and fourth cluster includes (independent enablers). According to the results, C3 (Compliance and policy) and C8 (Architecture analysis) criteria are part of autonomous elements; and this indicated that these both disconnected from the system due to weak link with other criteria. It is noted that C4 (Strategy and matrices), C6 (Training), C7 (Security feature and design), and C10 (Software Environment) have strong driving and dependence power and affect other enablers due to strong linkage. C1 (Secure Testing) and C5 (Requirements) belong to an independent cluster, indicating that these criteria have weak, dependent, and strong driving power; they are also known as key enablers. According to the results, C2 (Standards) and C9 (Configuration management) belong to the dependent cluster, and this renders that C2 and C9 have strong dependence power but have weak driving power.

4.19. Application of fuzzy topsis

The findings of ISM only level the key categories of DevSecOps challenges. To address the uncertainties and vagueness in consideration of DevSecOps challenging factors, we applied the fuzzy TOPSIS approach. The fuzzy TOPSIS approach gives the ranks of each challenging factor that assists the practitioners in considering the highest priority challenging factors for the success and progression of

Table 12
Conical matrix after clustering enablers.

	C3	C8	C7	C9	C10	C1	C4	C5	C6	C2
C3	1	0	0	0	0	0	0	0	0	0
C8	0	1	0	0	0	0	0	0	0	0
C7	1	1	1	1	1	0	1	0	1	0
C9	1	0	1	1	1	0	0	0	0	0
C10	1	0	1	0	1	0	1	0	1	0
C1	0	1	1	1	1	1	1	1	1	1
C4	0	1	1	1	1	1	1	1	1	0
C5	0	0	1	1	1	0	1	1	1	0
C6	0	0	1	1	1	0	1	0	1	0
C2	1	0	0	0	1	0	0	0	1	1

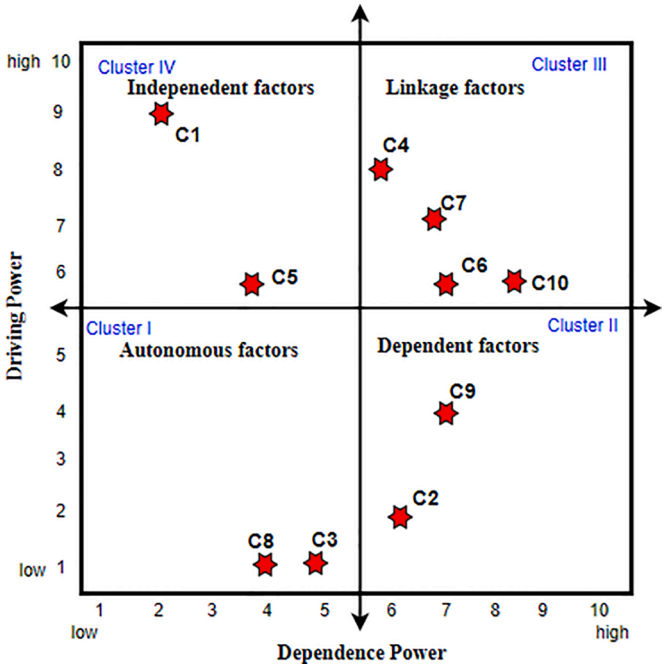


Fig. 8. Graphical view of MICMAC analysis.

DevSecOps. The fuzzy-TOPSIS approach has been used in studies of other engineering domains to address multi-criteria decision-making problems [48,74–76].

To perform the fuzzy-TOPSIS approach, we requested ten experts (who participated in ISM, appendix-C) to mark each challenging factor concerning their own experience and understanding. We developed a questionnaire as presented in Appendix C. Each participant could consult other colleagues while ranking the complex factors to get a more representative response from experts. The following steps were applied to calculate the fuzzy-TOPSIS results.

Step 1& 2: To get the insights of experts regarding the effectiveness of DevSecOps challenging factors, we used the fuzzy triangular scale, which gives linguistic values(Table 5) [42–44].

Step 3: Using Eq. (1) (Section 3.4), we compute the combined decision matrix. In this paper, we study a total of 18 challenging factors that are related to the ten core criteria of DevSecOps Table 13. shows the combined decision matrix, which shows the collective opinion of all the experts involved in decision-making.

Step 4: In this step, we calculated the normalized decision matrix, and to achieve this, we used Eqs. (2) and (3) (Section 3.4). To normalize the decision matrix, we must consider cost and beneficial criteria [77]. The cost and benefit criteria is a systematic process to measure the strengths and weaknesses of alternatives used to determine options that

Table 13
Combined decision matrix.

		Weight	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8	Ch9	Ch10	Ch11	Ch12	Ch13	Ch14	Ch15	Ch16	Ch17	Ch18
Secure Testing	Combined decision matrix	1.0	0.5	0.5	0.5	1.5	0.5	0.5	0.5	0.5	0.5	0.5	1.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
		1.2	2.5	1.6	1.4	2.5	2.0	2.0	1.8	1.3	1.2	1.2	2.1	1.0	1.4	1.0	1.2	1.2	1.3	1.3
		2.0	3.5	3.0	3.0	3.0	3.0	3.0	3.5	2.0	2.5	2.5	3.0	1.5	2.5	1.5	2.0	2.0	2.0	2.5
Standards		1.2	1.5	0.5	1.0	1.5	0.5	0.5	1.5	1.5	0.5	0.5	1.0	0.5	1.0	0.5	1.0	1.0	0.1	0.5
		2.5	2.4	2.1	1.2	2.1	1.2	1.2	2.3	1.8	1.2	1.2	2.0	1.4	1.4	1.4	1.4	1.2	1.8	1.0
		3.5	3.5	3.0	2.5	3.0	2.0	2.5	3.0	3.0	2.5	2.0	3.0	2.5	2.0	2.5	2.0	2.0	2.0	1.5
Compliance and policy		0.5	1.5	1.5	0.5	2.2	1.5	1.5	0.5	1.5	0.5	1.5	0.5	1.5	1.5	1.5	0.5	1.5	0.5	0.5
		1.0	2.5	2.4	2.3	3.0	2.4	2.0	1.7	2.3	2.1	2.3	2.1	2.4	2.4	2.3	1.4	2.3	1.7	1.4
		2.0	3.5	3.5	3.5	3.5	3.5	3.0	3.0	3.5	3.5	3.5	3.5	3.5	3.5	3.5	2.5	3.5	3.0	2.5
Strategy and matrices		0.5	1.5	0.5	1.0	0.5	1.5	0.5	1.5	0.1	0.5	0.1	0.5	0.5	0.5	0.1	1.0	0.1	1.5	1.0
		1.4	2.4	2.1	1.2	1.3	1.2	1.6	2.3	1.8	1.2	1.8	1.2	1.4	1.4	1.8	1.4	1.8	2.3	1.4
		2.5	3.5	3.0	2.5	3.0	3.0	2.5	3.0	3.0	2.5	2.0	2.5	2.5	2.5	3.0	2.0	3.0	3.0	2.0
Requirements		0.5	1.5	0.5	0.5	1.3	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
		1.0	2.5	1.3	1.4	3.0	2.0	1.0	1.8	1.3	1.2	1.3	1.2	1.0	1.0	1.3	1.2	1.3	1.8	1.2
		1.5	3.0	1.5	3.0	3.5	3.0	1.5	3.5	2.0	2.5	2.0	2.5	1.5	1.5	2.0	2.0	2.0	3.5	2.0
Training		0.5	1.5	1.5	0.5	1.5	1.5	1.5	0.5	1.5	0.5	0.5	0.5	1.5	1.5	1.5	0.5	0.5	1.5	0.5
		1.4	2.5	2.4	2.3	2.5	2.2	2.4	1.7	2.3	2.1	2.1	1.5	2.4	2.3	2.4	1.4	1.4	2.3	1.0
		2.5	3.5	3.5	3.5	3.5	3.5	3.5	3.0	3.5	3.5	3.0	3.0	3.5	3.0	3.5	2.5	2.5	3.0	1.5
Security feature and design		0.5	0.5	0.5	0.5	1.5	0.5	1.5	0.5	0.5	1.0	0.5	1.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
		1.2	1.0	1.6	1.3	2.4	1.0	1.2	1.7	1.0	1.3	1.0	1.3	1.0	1.0	1.0	1.5	1.2	1.4	1.0
		2.0	1.5	2.5	2.5	3.5	1.5	2.5	3.0	1.5	2.0	1.5	2.0	1.5	2.0	1.5	2.5	2.0	2.0	1.5
Architecture analysis		0.5	1.5	1.5	0.5	2.2	1.5	1.5	0.5	1.5	0.5	1.5	0.5	1.5	1.5	1.5	0.5	1.5	0.5	0.5
		1.3	2.5	2.4	2.3	3.0	2.4	2.0	1.7	2.3	2.1	2.3	2.1	2.4	2.4	2.3	1.4	2.3	1.7	1.4
		3.0	3.5	3.5	3.5	3.5	3.5	3.0	3.0	3.5	3.5	3.5	3.5	3.5	3.5	3.5	2.5	3.5	3.0	2.5
Configuration management		0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
		1.5	1.6	1.7	1.4	2.1	1.2	1.0	1.0	1.0	2.0	1.0	2.0	1.0	1.0	1.0	1.2	1.0	1.0	1.2
		2.5	3.5	3.0	3.5	3.0	2.5	2.5	1.5	1.5	3.0	1.5	3.0	1.5	1.5	1.5	2.5	1.5	1.5	2.5
Software Environment		0.5	1.5	0.5	1.0	0.5	1.5	0.5	1.5	0.1	0.5	0.1	0.5	0.5	0.5	0.1	1.0	0.1	1.5	1.0
		1.2	2.4	2.1	1.2	1.3	1.2	1.6	2.3	1.8	1.2	1.8	1.2	1.4	1.4	1.8	1.4	1.8	2.3	1.4
		2.5	3.5	3.0	2.5	3.0	3.0	2.5	3.0	3.0	2.5	2.0	2.5	2.5	2.5	3.0	2.0	3.0	3.0	2.0

provide the best approach to achieving benefits while performing a specific task [77]. Hence, to complete the normalization process, we used the 'requirements' as a cost criterion as it includes the challenging factors related to 'translate compliance constraints to requirements. The results of the normalized decision matrix are presented in Table 14.

Step 5: in this step, we calculated the weighted decision matrix based on the weights assigned by the group of experts for each criterion were multiplied with their alternative (i.e., challenging factors) using Eq. (4), and the determined results are given in Table 15.

Step 6: In this step, we execute Eqs. (5) and (6) and calculate the fuzzy ideal solutions (positive and negative). The fuzzy positive and negative solutions are given in Tables 16 and 17.

Step 7: In this step, we determined the distance Di* and Di- for each DevSecOps challenging factor for criteria (Eq. (7) and Eq. (8)), and the results are given in Tables 16 and 17. Whereas: ST = Secure Testing, S = Standards, C = Compliance, and policy, SM = Strategy and matrices, R = Requirements, T = Training, SF = Security feature and design, A = Architecture analysis, C = Configuration management, SE = Software Environment.

Step 8: Using Eq. (9), the closeness coefficient was calculated, and the results are given in Table 18.

Step 9: Considering the closeness coefficient values, the ranks for each challenging factor were determined. Higher the cci value of a challenging factor shows their higher ranking (Table 18). For example, Ch4 (Lack of secure coding standards, CCi = 0.78) is the highest challenging factor for the DevSecOps paradigm. We further noted that Ch1 (Lack of automated testing tools for security in DevOps, CCi = 0.77), Ch2 (Ignorance in static testing for security due to lack of knowledge, CCi = 0.74) are ranked as the 2nd and 3rd most critical challenging factors for the DevSecOps process.

According to the final determined ranks presented in Table 18, "Ch4 (Lack of secure coding standards, CCi=0.78) is ranked as the highest priority challenging factor. For example, Prates et al. [78] stated that standards can be positive ("do it this way") or negative ("do not use this API"), but they must be enforced to be useful. Furthermore, Rahul et al. [60] defined that an obligatory phase of a firm's secure coding standards mostly begins as sufficient grounds for rejecting a piece of code. Other useful coding standard topics might include proper use of cloud APIs, approved cryptography, memory sanitization, and many others. They further indicated that the code review against standards must be objective: it should not become a debate about whether the non-compliant code is exploitable. Tony [16] underlined that the coding standards are specific to language constructs and enforced with tools (e.g., codified into SAST rules) in some situations. In other cases, published coding standards are specific to technology stacks and enforced during the code review process" or using automation. Ch1 (Lack of automated testing tools for security in DevOps, CCi = 0.77) is ranked as the 2nd most critical challenging factor for the DevSecOps paradigm. In DevSecOps, the adoption of black-box security testing tools is vital for the quality assessment process. Automated testing tools are also essential to encapsulate the attacker [14]. Ch2 (Ignorance in static testing for security due to lack of knowledge, CCi = 0.74) is ranked as the 3rd most significant challenging factor for the successful execution of the DevSecOps paradigm.

An interesting observation is that the Ch5 (Inconsistency security policies design and performance measures, CCi = 0.56), Ch6 (Developers' resistance to integrating security protocols, CCi = 0.56), Ch8 (Using unstable performance metrics for security evaluation, CCi = 0.56) and Ch9 (Translate Compliance constraints to requirements, CCi = 0.56) are ranked as 5th most important challenging factors for DevSecOps. According to the experts, this renders that these four challenging factors are equally crucial for DevSecOps process execution.

4.20. Holistic model of devsecops challenging factors

Finally, we developed a holistic model of DevSecOps challenging factors and their leveling determined via the ISM approach. The developed model (Fig. 9) indicated the global ranking (GR) and local ranking (LR) of each of the challenging factors. The global ranking presents the priority order of each challenging factor by applying the step-by-step protocols of fuzzy-TOPSIS. It indicates the priority order of the challenging factors compared to all the reported challenging factors. The local ranking then presents the priority order of each challenge within its respective category, which is useful when checking the impact of a challenging factor locally (within the category).

Figs. 7 and 8 show that the C2 (Standards) category stands on the top for selecting DevSecOps challenging factors categories. This indicated that C2 is an independent category of the DevSecOps challenging factor, which shows that all the other challenging categories are dependent on C2. Considering the local rankings (Fig. 9,Table 18), Ch4 (Lack of secure coding standards) is the highest-ranked challenging factor for the DevSecOps paradigm. We further noted that Ch4 is also declared the highest priority challenging factor for global ranking. Practitioners need to consider Ch4 as a priority for the success and progression of DevSecOps in real-world projects.

Moreover, the results (Fig. 9 and Table 18) show that Ch3 (Communicate security standard to DevOps team) is ranked as the 2nd category, i.e., C2 (Standards) category. Still, considering the global ranking, it stands as the 4th priority challenging factor for DevSecOps. This analysis shows that the global ranking of the challenging factors could vary by comparing the weights of all 18 identified challenging factors. Thus, the practitioners need to pay due attention to considering the priority of the DevSecOps challenging factors concerning their designation (nature of work) and interest.

Furthermore, the results (Figs. 7 and 9) presented that C1 (Secure Testing), C4 (Strategy and matrices), C5 (Requirements), and C6 (Training) are ranked on level 4, and these categories depend on only one category, C2 (Standards), i.e., level 5. According to the ISM analysis results, the rest of all: levels 3, level 2, and level 1, categories depend on level 4.

The fuzzy-TOPSIS results indicated that Ch1 (Lack of automated testing tools for security in DevOps), Ch2 (Ignorance in static testing for security due to lack of knowledge) are ranked as the 2nd and 3rd most critical challenging factors for DevSecOps. The practitioners should consider the challenging and individual factors, considering their category level and local and global rankings.

5. Summary of study findings

This study explores and analyzes the challenges faced while adopting DevSecOps, as reported by academic and industry researchers and faced by real-world practitioners. Firstly, to investigate the challenging factors of DevSecOps, we did a multivocal literature review. We collected most potential literature published as scientific research and gray literature, including experience reports, blogs, white papers, case studies, etc. With the protocols of multivocal literature review, 46 formal and 41 gray literature studies were selected for the final data extraction process. By carefully reviewing the final selected literature, we identified the list of 18 challenging factors that are related to 10 key categories of DevSecOps. Secondly, the questionnaire survey was conducted with experts; it evaluated the significance of the identified challenging factors in real-world practices. During the survey data collection process, 113 complete responses were collected. The frequency analysis showed that the identified list of 18 challenging factors and their core categories are related to industry practices.

We further applied the ISM approach to examine the relationships between the ten core categories of DevSecOps. The results show that C3 (Compliance and policy) and C8 (Architecture analysis) criteria are autonomous elements. This indicated that these both had only weak links to other criteria. C4 (Strategy and matrices), C6 (Training), C7 (Security feature and design), and C10 (Software Environment) have strong driving and dependence power and affect other enablers due to

Table 14
Normalized decision matrix.

		Weights	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8	Ch9	Ch10	Ch11	Ch12	Ch13	Ch14	Ch15	Ch16	Ch17	Ch18
Secure Testing	Normalized decision matrix	1.0	0.1	0.1	0.1	0.4	0.1	0.1	0.1	0.1	0.1	0.1	0.4	0.1	0.1	0.1	0.1	0.1	0.1	0.1
		1.2	0.7	0.5	0.4	0.7	0.6	0.6	0.5	0.4	0.3	0.3	0.6	0.3	0.4	0.3	0.3	0.3	0.4	0.4
		2.0	1.0	0.9	0.9	0.9	0.9	0.9	1.0	0.6	0.7	0.7	0.9	0.4	0.7	0.4	0.6	0.6	0.6	0.7
Standards		1.2	0.4	0.1	0.3	0.4	0.1	0.1	0.4	0.4	0.1	0.1	0.3	0.1	0.3	0.1	0.3	0.3	0.0	0.1
		2.5	0.7	0.6	0.3	0.6	0.3	0.3	0.7	0.5	0.3	0.3	0.6	0.4	0.4	0.4	0.4	0.3	0.5	0.3
		3.5	1.0	0.9	0.7	0.9	0.6	0.7	0.9	0.9	0.7	0.6	0.9	0.7	0.6	0.7	0.6	0.6	0.6	0.4
Compliance and policy		0.5	0.4	0.4	0.1	0.6	0.4	0.4	0.1	0.4	0.1	0.4	0.1	0.6	0.4	0.4	0.1	0.4	0.1	0.1
		1.0	0.7	0.7	0.7	0.9	0.7	0.6	0.5	0.7	0.6	0.7	0.6	0.7	0.7	0.7	0.4	0.7	0.5	0.4
		2.0	1.0	1.0	1.0	1.0	1.0	0.9	0.9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	1.0	0.9	0.7
Strategy and matrices		0.5	0.4	0.1	0.3	0.1	0.4	0.1	0.4	0.0	0.1	0.0	0.1	0.1	0.1	0.0	0.3	0.0	0.4	0.3
		1.4	0.7	0.6	0.3	0.4	0.3	0.5	0.7	0.5	0.3	0.5	0.3	0.4	0.4	0.5	0.4	0.5	0.7	0.4
		2.5	1.0	0.9	0.7	0.9	0.9	0.7	0.9	0.9	0.7	0.6	0.7	0.7	0.7	0.9	0.6	0.9	0.9	0.6
Requirements		0.5	0.2	0.3	0.2	0.1	0.2	0.3	0.1	0.3	0.2	0.3	0.2	0.3	0.3	0.3	0.3	0.3	0.1	0.3
		1.0	0.2	0.4	0.4	0.2	0.3	0.5	0.3	0.4	0.4	0.4	0.4	0.5	0.5	0.4	0.4	0.4	0.3	0.4
		1.5	0.3	1.0	1.0	0.4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Training		0.5	0.4	0.4	0.1	0.4	0.4	0.4	0.1	0.4	0.1	0.1	0.1	0.4	0.4	0.4	0.1	0.1	0.4	0.1
		1.4	0.7	0.7	0.7	0.7	0.6	0.7	0.5	0.7	0.6	0.6	0.4	0.7	0.7	0.7	0.4	0.4	0.7	0.3
		2.5	1.0	1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0	0.9	0.9	1.0	0.9	1.0	0.7	0.7	0.9	0.4
Security feature and design		0.5	0.1	0.1	0.1	0.4	0.1	0.4	0.1	0.1	0.3	0.1	0.3	0.1	0.1	0.1	0.1	0.1	0.1	0.1
		1.2	0.3	0.5	0.4	0.7	0.3	0.3	0.5	0.3	0.4	0.3	0.4	0.3	0.3	0.3	0.4	0.3	0.4	0.3
		2.0	0.4	0.7	0.7	1.0	0.4	0.7	0.9	0.4	0.6	0.4	0.6	0.4	0.6	0.4	0.7	0.6	0.6	0.4
Architecture analysis		0.5	0.4	0.4	0.1	0.6	0.4	0.4	0.1	0.4	0.1	0.4	0.1	0.4	0.4	0.4	0.1	0.4	0.1	0.1
		1.3	0.7	0.7	0.7	0.9	0.7	0.6	0.5	0.7	0.6	0.7	0.6	0.7	0.7	0.7	0.4	0.7	0.5	0.4
		3.0	1.0	1.0	1.0	1.0	1.0	0.9	0.9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	1.0	0.9	0.7
Configuration management		0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
		1.5	0.5	0.5	0.4	0.6	0.3	0.3	0.3	0.3	0.6	0.3	0.6	0.3	0.3	0.3	0.3	0.3	0.3	0.3
		2.5	1.0	0.9	1.0	0.9	0.7	0.7	0.4	0.4	0.9	0.4	0.9	0.4	0.4	0.4	0.7	0.4	0.4	0.7
Software Environment		0.5	0.4	0.1	0.3	0.1	0.4	0.1	0.4	0.0	0.1	0.0	0.1	0.1	0.1	0.0	0.3	0.0	0.4	0.3
		1.2	0.7	0.6	0.3	0.4	0.3	0.5	0.7	0.5	0.3	0.5	0.3	0.4	0.4	0.5	0.4	0.5	0.7	0.4
		2.5	1.0	0.9	0.7	0.9	0.9	0.7	0.9	0.9	0.7	0.6	0.7	0.7	0.7	0.9	0.6	0.9	0.9	0.6

Table 15
Weighted normalized decision matrix.

		WEIGHTS	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8	Ch9	Ch10	Ch11	Ch12	Ch13	Ch14	Ch15	Ch16	Ch17	Ch18	A+	A-
Secure Testing	Weighted Normalized decision matrix	1.0	0.1	0.1	0.1	0.4	0.1	0.1	0.1	0.1	0.1	0.1	0.4	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.4	0.1
		1.2	0.9	0.5	0.5	0.9	0.7	0.7	0.6	0.4	0.4	0.4	0.7	0.3	0.5	0.3	0.4	0.4	0.4	0.4	0.9	0.3
		2.0	2.0	1.7	1.7	1.7	1.7	1.7	2.0	1.1	1.4	1.4	1.7	0.9	1.4	0.9	1.1	1.1	1.1	1.4	2.0	0.9
Standards		1.2	0.5	0.2	0.3	0.5	0.2	0.2	0.5	0.5	0.2	0.2	0.3	0.2	0.3	0.2	0.3	0.3	0.0	0.2	0.5	0.0
		2.5	1.7	1.5	0.9	1.5	0.9	0.9	1.6	1.3	0.9	0.9	1.4	1.0	1.0	1.0	1.0	0.9	1.3	0.7	1.7	0.7
		3.5	3.5	3.0	2.5	3.0	2.0	2.5	3.0	3.0	2.5	2.0	3.0	2.5	2.0	2.5	2.0	2.0	2.0	1.5	3.5	1.5
Compliance and policy		0.5	0.2	0.2	0.1	0.3	0.2	0.2	0.1	0.2	0.1	0.2	0.1	0.3	0.2	0.2	0.1	0.2	0.1	0.1	0.3	0.1
		1.0	0.7	0.7	0.7	0.9	0.7	0.6	0.5	0.7	0.6	0.7	0.6	0.7	0.7	0.7	0.4	0.7	0.5	0.4	0.9	0.4
		2.0	2.0	2.0	2.0	2.0	2.0	1.7	1.7	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.4	2.0	1.7	1.4	2.0	1.4
Strategy and matrices		0.5	0.2	0.1	0.1	0.1	0.2	0.1	0.2	0.0	0.1	0.0	0.1	0.1	0.1	0.0	0.1	0.0	0.2	0.1	0.2	0.0
		1.4	1.0	0.8	0.5	0.5	0.5	0.6	0.9	0.7	0.5	0.7	0.5	0.6	0.6	0.7	0.6	0.7	0.9	0.6	1.0	0.5
		2.5	2.5	2.1	1.8	2.1	2.1	1.8	2.1	2.1	1.8	1.4	1.8	1.8	1.8	2.1	1.4	2.1	2.1	1.4	2.5	1.4
Requirements		0.5	0.1	0.2	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.2	0.1
		1.0	0.2	0.4	0.4	0.2	0.3	0.5	0.3	0.4	0.4	0.4	0.4	0.5	0.5	0.4	0.4	0.4	0.3	0.4	0.5	0.2
		1.5	0.5	1.5	1.5	0.6	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	0.5
Training		0.5	0.2	0.2	0.1	0.2	0.2	0.2	0.1	0.2	0.1	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.2	0.1	0.2	0.1
		1.4	1.0	1.0	0.9	1.0	0.9	1.0	0.7	0.9	0.8	0.8	0.6	1.0	0.9	1.0	0.6	0.6	0.9	0.4	1.0	0.4
		2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.1	2.5	2.5	2.1	2.1	2.5	2.1	2.5	1.8	1.8	2.1	1.1	2.5	1.1
Security feature and design		0.5	0.1	0.1	0.1	0.2	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.1
		1.2	0.3	0.5	0.4	0.8	0.3	0.4	0.6	0.3	0.4	0.3	0.4	0.3	0.3	0.3	0.5	0.4	0.5	0.3	0.8	0.3
		2.0	0.9	1.4	1.4	2.0	0.9	1.4	1.7	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.4	1.1	1.1	0.9	2.0	0.9
Architecture analysis		0.5	0.2	0.2	0.1	0.3	0.2	0.2	0.1	0.2	0.1	0.2	0.1	0.2	0.2	0.2	0.1	0.2	0.1	0.1	0.3	0.1
		1.3	0.9	0.9	0.9	1.1	0.9	0.7	0.6	0.9	0.8	0.9	0.8	0.9	0.9	0.9	0.5	0.9	0.6	0.5	1.1	0.5
		3.0	3.0	3.0	3.0	3.0	3.0	2.6	2.6	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.1	3.0	2.6	2.1	3.0	2.1
Configuration management		0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
		1.5	0.7	0.7	0.6	0.9	0.5	0.4	0.4	0.4	0.9	0.4	0.9	0.4	0.4	0.4	0.5	0.4	0.4	0.5	0.9	0.4
		2.5	2.5	2.1	2.5	2.1	1.8	1.8	1.1	1.1	2.1	1.1	2.1	1.1	1.1	1.1	1.8	1.1	1.1	1.8	2.5	1.1
Software Environment		0.5	0.2	0.1	0.1	0.1	0.2	0.1	0.2	0.0	0.1	0.0	0.1	0.1	0.1	0.0	0.1	0.0	0.2	0.1	0.2	0.0
		1.2	0.8	0.7	0.4	0.4	0.4	0.5	0.8	0.6	0.4	0.6	0.4	0.5	0.5	0.6	0.5	0.6	0.8	0.5	0.8	0.4
		2.5	2.5	2.1	1.8	2.1	2.1	1.8	2.1	2.1	1.8	1.4	1.8	1.8	1.8	2.1	1.4	2.1	2.1	1.4	2.5	1.4

Table 16
Fuzzy positive ideal solution.

	ST	S	C	SM	R	T	SF	A	C	SE	di*
Ch1	0.16	0.00	0.10	0.00	0.60	0.00	0.72	0.12	0.12	0.00	1.84
Ch2	0.29	0.37	0.11	0.23	0.07	0.02	0.38	0.14	0.23	0.23	2.08
Ch3	0.32	0.77	0.18	0.50	0.10	0.09	0.40	0.21	0.17	0.48	3.22
Ch4	0.16	0.31	0.00	0.34	0.57	0.00	0.00	0.00	0.21	0.31	1.90
Ch5	0.25	1.02	0.11	0.35	0.15	0.07	0.72	0.14	0.47	0.31	3.60
Ch6	0.25	0.79	0.24	0.46	0.00	0.02	0.41	0.33	0.49	0.45	3.44
Ch7	0.22	0.29	0.30	0.21	0.14	0.29	0.23	0.40	0.87	0.21	3.15
Ch8	0.57	0.38	0.13	0.27	0.07	0.05	0.72	0.16	0.87	0.26	3.49
Ch9	0.45	0.79	0.20	0.50	0.06	0.12	0.54	0.24	0.21	0.48	3.60
Ch10	0.45	1.02	0.13	0.64	0.07	0.24	0.72	0.16	0.87	0.64	4.94
Ch11	0.18	0.35	0.20	0.50	0.06	0.32	0.54	0.24	0.21	0.48	3.09
Ch12	0.74	0.74	0.10	0.48	0.00	0.02	0.72	0.14	0.87	0.46	4.28
Ch13	0.43	0.96	0.11	0.48	0.00	0.21	0.57	0.14	0.87	0.46	4.25
Ch14	0.74	0.74	0.13	0.27	0.07	0.02	0.72	0.16	0.87	0.26	3.99
Ch15	0.58	0.96	0.45	0.66	0.05	0.49	0.38	0.62	0.47	0.65	5.32
Ch16	0.58	1.00	0.13	0.27	0.07	0.49	0.56	0.16	0.87	0.26	4.40
Ch17	0.57	0.94	0.30	0.21	0.14	0.21	0.54	0.40	0.87	0.21	4.39
Ch18	0.44	1.31	0.45	0.66	0.05	0.90	0.72	0.62	0.47	0.65	6.26

Table 17
Fuzzy negative ideal solution.

	ST	S	C	SM	R	T	SF	A	C	SE	di-
Ch1	0.72	1.32	0.39	0.69	0.02	0.90	0.00	0.55	0.84	0.67	6.10
Ch2	0.51	0.98	0.38	0.46	0.59	0.89	0.35	0.55	0.64	0.45	5.80
Ch3	0.50	0.61	0.36	0.22	0.59	0.88	0.34	0.53	0.83	0.22	5.07
Ch4	0.60	1.02	0.45	1.27	0.04	0.90	0.72	0.62	0.68	0.41	6.71
Ch5	0.53	0.31	0.38	0.43	0.58	0.87	0.00	0.55	0.42	0.43	4.49
Ch6	0.53	0.59	0.21	0.23	0.61	0.89	0.34	0.29	0.41	0.22	4.33
Ch7	0.68	1.06	0.17	0.50	0.58	0.64	0.51	0.26	0.00	0.48	4.87
Ch8	0.18	0.97	0.37	0.44	0.59	0.88	0.00	0.54	0.00	0.43	4.39
Ch9	0.33	0.59	0.35	0.21	0.60	0.86	0.18	0.52	0.67	0.21	4.51
Ch10	0.33	0.31	0.37	0.14	0.59	0.67	0.00	0.54	0.00	0.12	3.07
Ch11	0.57	0.98	0.35	0.21	0.60	0.63	0.18	0.52	0.67	0.21	4.90
Ch12	0.00	0.61	0.39	0.21	0.61	0.89	0.00	0.55	0.00	0.21	3.47
Ch13	0.34	0.38	0.38	0.21	0.61	0.69	0.16	0.55	0.00	0.21	3.54
Ch14	0.00	0.61	0.37	0.44	0.59	0.89	0.00	0.54	0.00	0.43	3.86
Ch15	0.17	0.38	0.00	0.09	0.60	0.42	0.34	0.00	0.42	0.08	2.50
Ch16	0.17	0.35	0.37	0.44	0.59	0.42	0.17	0.54	0.00	0.43	3.48
Ch17	0.18	0.44	0.17	0.50	0.58	0.69	0.18	0.26	0.00	0.48	3.48
Ch18	0.34	0.08	0.00	0.09	0.60	0.00	0.00	0.00	0.42	0.08	1.60

solid linkage. C1 (Secure Testing) and C5 (Requirements) belong to an independent cluster indicating a weak dependent but strong driving power. This meant that it is not dependent on other categories but has practical significance. They can also be considered key enablers. C2 (Standards) and C9 (Configuration management) belong to the dependent cluster, rendering C2 and C9 have a strong dependence power but a weak driving power.

The uncertainty in experts' opinions can be a problem when identifying challenging factors. We applied the fuzzy TOPSIS approach to address this concern to fix the multicriteria decision-making problem. The results show that Ch4 (Lack of secure coding standards) is the highest challenging factor. Ch1 (Lack of automated testing tools for security in DevOps), Ch2 (Ignorance in static testing for security due to lack of knowledge), and Ch3 (Communicate security standard to DevOps team) are the top four challenging factors for the successful execution of DevSecOps.

6. Study implications

The study implications for researchers and practitioners are as follows:

For researchers: The study provides a state-of-the-art overview of the challenges that can impact DevSecOps processes by providing a review of both academic and gray literature. The study's findings offer a body of knowledge to researchers to develop strategies for the successful implementation of DevSecOps projects. The study also provides a rank-based framework of the identified challenging factors. The investigated factors are examined in their priority ranking and the relationship between the core categories of the identified challenging factors. We believe that prioritization-based ranking helps the researchers to consider the most significant challenging factors in their future research.

For Practitioners: The in-depth multivocal literature study and empirical investigations provide industry experts with knowledge about the challenging areas of the DevSecOps process. This study presents 18 challenges, and each challenge calls on the industry practitioners to focus on during the implementation of DevSecOps projects. The prioritization of the identified challenges will assist the practitioners in considering the most significant challenging factor on priority. Identifying challenges and their prioritization will help the practitioners revise and develop the new strategies for the successful execution of DevSecOps practices. Moreover, this study provides a holistic model of DevSecOps challenges, informing practitioners about which challenge is vital in which category. This study also induced ISM and fuzzy TOPSIS as novel approaches in this domain that assist the industry experts in fixing the uncertainty and vague opinions of DevSecOps experts.

Future work: To date, little research has been conducted on developing models and strategies for DevSecOps practices in the real-world industry. Hence, despite the importance of incorporating security into DevOps, no maturity model supports DevSecOps in software development processes by highlighting critical success factors, critical

Table 18
Closeness coefficient values and Ranks.

Categories	Sr. NO.	Challenging factors	CCi	Local Ranks	GlobalRanks
Secure Testing (C1)	Ch1	Lack of automated testing tools for security in DevOps	0.77	1	2
	Ch2	Ignorance in static testing for security due to lack of knowledge	0.74	2	3
Standards (C2)	Ch3	Communicate security standard to DevOps team	0.61	2	4
	Ch4	Lack of secure coding standards	0.78	1	1
Compliance and policy (C3)	Ch5	Inconsistence security polices design and performance measures	0.56	2	5
	Ch6	Developers' resistance to integrate security protocols	0.56	2	5
	Ch7	Neglecting change control in security	0.61	1	4
Strategy and matrices (C4)	Ch8	Using unstable performance metrics for security evaluation	0.56	1	5
Requirements (C5)	Ch9	Translate Compliance constraints to requirements	0.56	1	5
Training (C6)	Ch10	Lack of software security awareness	0.38	1	9
Security feature and design (C7)	Ch11	Integrate and deliver security features	0.61	1	4
	Ch12	Threat modeling scalability issue	0.45	2	7
Architecture analysis (C8)	Ch13	Perform security feature review	0.45	1	7
Configuration management (C9)	Ch14	Periodic penetration test for application coverage	0.49	1	6
	Ch15	Define secure deployment parameters and configurations	0.32	3	10
	Ch16	Identify software defects found in operations monitoring and feed them back to the development	0.44	2	8
Software Environment (C10)	Ch17	Use application behavior monitoring and diagnostics	0.44	1	8
	Ch18	Use of immature automated deployment tools	0.20	2	11

challenges, and best practices, and a road map. Considering the importance of security concerns in software development, we are motivated to develop a specific DevSecOps maturity model (DevSecOpsMM) that will assist the software organizations in measuring their maturity level and suggest the best practices for successful implementation execution of DevSecOps activities. DevSecOpsMM will be based on the empirically uncovered challenges, success factors, and best practices of DevSecOps and existing maturity models of other software engineering domains.

The DevSecOpsMM will be developed using the steps shown in Fig. 10. "The DevSecOpsMM model consists of three core components, i.e., maturity level component, factors component (critical success factors (CSFs), critical challenges (CCHs)) and assessment component. The current study only contributed to developing a one-factor component, i.e., critical challenging factors Fig. 10 shows the association between the key components of DevSecOpsMM. The maturity level component is considered to assess an organization's maturity level regarding the DevSecOps process. The factors component consists of the CSFs and CCHs representing the DevSecOps process's key areas. The assessment component is used to assess an organization's specific maturity level and suggest the best practices to boost the DevSecOps capabilities." The proposed model will assist in executing DevSecOps activities in the real-world industry.

7. Limitations

The MLR was performed to collect the formal and gray literature related to the study's objective. The first author of this study executed the literature searching and collection process, and the second author of this study reviewed the selected literature and performed the data extraction. Thus, there might be a threat of bias in the extracted data. Authors no-3 and 4 carried out the inclusion, exclusion, quality assessment, and data extraction process for five formal and five gray studies to address this threat. Moreover, we applied the inter-rater reliability test with external experts, and the results show that there is no significant bias and that the collected data and analysis are consistent.

Secondly, there might be a chance of missing some related literature during the data collection process. Our study has a representative set of 87 literature items, and therefore this limitation is not systematic [21, 70]. The questionnaire survey approach has been applied to investigate the identified challenging factors with industry experts empirically. There is always a threat in the survey instrument development. This threat has been addressed by piloting and assessing the development questionnaire with external experts and lab mates. The findings of ISM and the fuzzy TOPSIS approach are based on the decisions of ten experts, and this might be a small data set. These studies are subjective in types, and various existing studies [42–44] also used the small data set for such kind of analysis. Thus, the results of ISM and fuzzy TOPSIS approaches are generalizable.

8. Conclusion

Security is an essential attribute of quality software. Software is secure if it does not allow confidentiality, integrity, and availability of its data, code or service to be compromised. Security plays an integral part in the development life cycle to take full advantage of DevOps. By considering the significance of security in software development, this paper explores the challenging factors faced by practitioners while executing DevSecOps activities. A list of 18 DevSecOps challenging factors was identified related to the ten key areas of software security. We further surveyed intending to get practitioners' views of the identified challenging factors and their core categories. The frequency analysis shows that identified list of challenging factors and their categories are relevant to the industry.

We further examined the relationship between the categories of DevSecOps challenging factors using the ISM approach. The ISM analysis results show that the 'Standards' category has the highest driving power. We also applied the fuzzy-TOPSIS approach to address the multi-criteria decision-making problem of DevSecOps challenges. The results show that 'lack of secure coding standards,' 'lack of automated testing tools for security in DevOps,' 'ignorance in static testing for security due to lack of knowledge, and 'Communicate security standard to DevOps team' are declared as the highest priority challenging factors for DevSecOps paradigm.

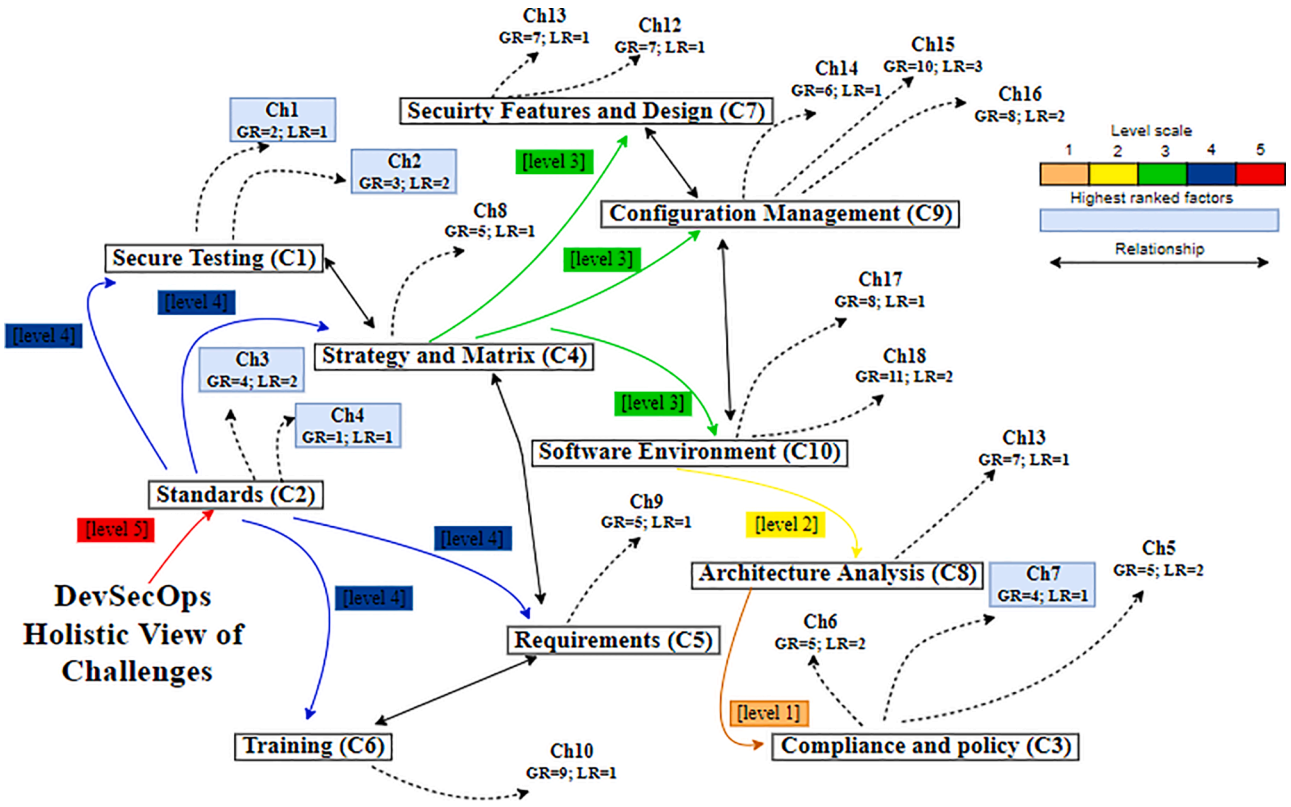


Fig. 9. Holistic model of DevSecOps challenges.

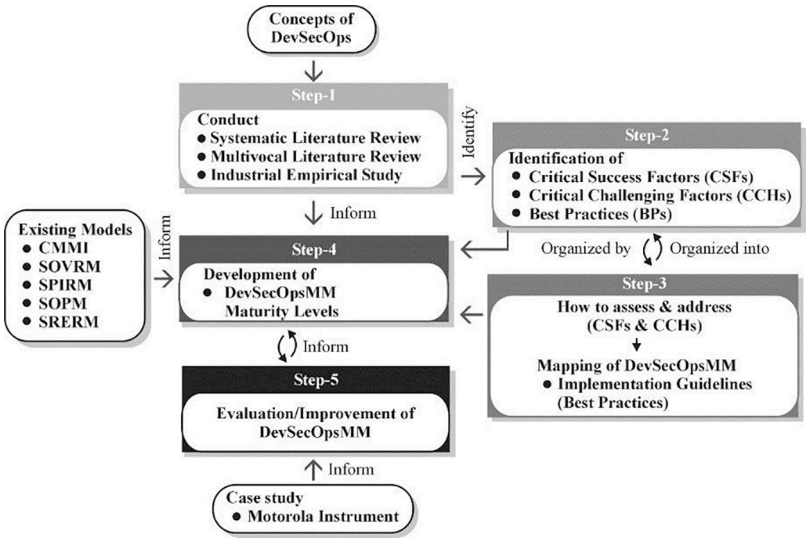


Fig. 10. Structure of proposed model.

CRediT authorship contribution statement

Muhammad Azeem Akbar: Conceptualization, Data curation. **Kari Smolander:** Methodology, Supervision. **Sajjad Mahmood:** Writing – review & editing. **Ahmed Alsanad:** Resources, Formal analysis.

Declaration of Competing Interest

All the authors of the mentioned manuscript have no conflcts of interest.

Acknowledgment

The authors are grateful to the Deanship of Scientific Research, King Saud University, for funding through Vice Deanship of Scientific Research Chairs for empirical data collection and analysis.

Appendixes

Appendix-A: Selected Formal and gray literature: https://tinyurl.com/ykzffuud

Appendix-B: Sample of Used Survey instrument: https://tinyurl.

com/y8mze9ch

Appendix-C: Sample of Used Questionnaire for Fuzzy-TOPSIS: https://tinyurl.com/wn8xmfx6

Appendix-D: Experts involved in Fuzzy TOPSIS decision making: https://tinyurl.com/ndffubdx

References

[1] L. Leite, C. Rocha, F. Kon, D. Milojicic, P. Meirelles, A survey of DevOps concepts and challenges, ACM Comput. Surv. (CSUR) 52 (6) (2019) 1–35.

[2] Rembetsy, Michael, and Patrick McDonnell. Continuously deploying culture: Scaling culture at etsy. Velocity Europe. O'Reilly (2012).

[3] F. Erich, C. Amrit, M. Daneva, Report: Devops Literature Review, University of Twente, Tech. Rep, 2014.

[4] F. Erich, C. Amrit, M. Daneva, A qualitative study of DevOps usage in practice, J. Softw. Evol. Process 29 (6) (2017) e1885.

[5] K. Carter, Francois raynaud on DevSecOps, IEEE Softw. 34 (5) (2017) 93–96.

[6] H. Myrbakken, R. Colomo-Palacios, DevSecOps: a multivocal literature review, in: Proceedings of the International Conference on Software Process Improvement and Capability Determination, Springer, 2017, pp. 17–29.

[7] Yasar, Hasan. Overcoming DevSecOps Challenges: A Practical Guide for All Stakeholders. Technical Report, CARNEGIE-MELLON UNIV PITTSBURGH PA PITTSBURGH, DEFENSE TECHNICAL INFORMATION CENTER, 8725 John J. Kingman Road, Fort Belvoir, VA 22060-6218 1-800-CAL-DTIC (1-800-225-3842) United States, 2020. https://apps.dtic.mil/sti/pdfs/AD1110359.pdf.

[8] J. Roche, Adopting DevOps practices in quality assurance, Commun. ACM 56 (11) (2013) 38–43.

[9] M. Senapathi, J. Buchan, H. Osman, DevOps capabilities, practices, and challenges: insights from a case study, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, 2018, pp. 57–67.

[10] R. Jabbari, N. bin Ali, K. Petersen, B. Tanveer, What is DevOps? A systematic mapping study on definitions and practices, in: Proceedings of the Scientific Workshop Proceedings of XP2016, 2016, pp. 1–11.

[11] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation (Adobe Reader), Pearson Education, 2010.

[12] E. Woods, Operational: the forgotten architectural view, IEEE Softw. 33 (3) (2016) 20–23.

[13] J. Humble J, Farley D. Continuous delivery: reliable software releases through build, test, and deployment automation. Addison-Wesley Signature Series, Pearson Education, 27 Jul 2010 - Computers pp. 117–512.

[14] Goldschmidt, M., McKinnon, M: Devsecops - agility with security. Technical report, Sense of Security, pp. 44 (2016).

[15] D. Shackleford, The Devsecops Approach to Securing Your Code and Your Cloud, SANS Institute InfoSec Reading Room A DevSecOps Playbook, 2017.

[16] T.H.-C. Hsu, Hands-On Security in DevOps: Ensure Continuous Security, Deployment, and Delivery With DevSecOps, Packt Publishing Ltd, 2018.

[17] D. Shackleford, A Devsecops Playbook, SANS Institute InfoSec Reading Room. A DevSecOps Playbook, 2016.

[18] J. McKay, How to use devsecops to smooth cloud deployment. SVP and Chief Technology Officer, Logicworks, IDG Communications, Inc. 2016. https://www.networkworld.com/article/3041640/how-to-use-devsecops-to-smooth-cloud-deployment.html.

[19] V. Garousi, M. Felderer, M.V. Mäntylä, Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Inf. Softw. Technol. 106 (2019) 101–121. /02/01/2019.

[20] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3, EBSE Technical Report EBSE-2007-01.

[21] M. Niazi, S. Mahmood, M. Alshayeb, A.M. Qureshi, K. Faisal, N. Cerpa, Toward successful project management in global software development, Int. J. Project Manage. 34 (8) (2016) 1553–1567.

[22] H. Zhang, M.A. Babar, P. Tell, Identifying relevant studies in software engineering, Inf. Softw. Technol. 53 (6) (2011) 625–637.

[23] Chen, Lianipng, Muhammad Ali Babar, and He Zhang. Towards an evidence-based understanding of electronic data sources. In 14th International Conference on Evaluation and Assessment in Software Engineering (EASE), pp. 1–4. 2010.

[24] W. Afzal, S. Alone, K. Glocksien, R. Torkar, Software test process improvement approaches: a systematic literature review and an industrial case study, J. Syst. Softw. 111 (2016) 1–33.

[25] A.A. Khan, J. Keung, S. Hussain, M. Niazi, S. Kieffer, Systematic literature study for dimensional classification of success factors affecting process improvement in global software development: client–vendor perspective, IET Softw. 12 (4) (2018) 333–344.

[26] S.U. Khan, M. Niazi, R. Ahmad, Barriers in the selection of offshore software development outsourcing vendors: an exploratory study using a systematic literature review, Inf. Softw. Technol. 53 (7) (2011) 693–706.

[27] A.A. Khan, J. Keung, M. Niazi, S. Hussain, A. Ahmad, Systematic literature review and empirical investigation of barriers to process improvement in global software development: client–vendor perspective, Inf. Softw. Technol. 87 (2017) 180–205.

[28] M. Niazi, et al., Challenges of project management in global software development: a client-vendor analysis, Inf. Softw. Technol. 80 (2016) 1–19.

[29] S.U. Khan, M. Niazi, R. Ahmad, Factors influencing clients in the selection of offshore software outsourcing vendors: an exploratory study using a systematic literature review, J. Syst. Softw. 84 (4) (2011) 686–699.

[30] J.M. Corbin, A. Strauss, Grounded theory research: procedures, canons, and evaluative criteria, Qual. Sociol. 13 (1) (1990) 3–21.

[31] W. Afzal, R. Torkar, R. Feldt, A systematic review of search-based testing for non-functional system properties, Inf. Softw. Technol. 51 (6) (2009) 957–976.

[32] V. Lenarduzzi, D. Taibi, MVP explained: a systematic mapping study on the definitions of minimal viable product, in: Proceedings of the 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2016, pp. 112–119.

[33] S. Easterbrook, J. Singer, M.-A. Storey, D. Damian, Selecting empirical methods for software engineering research. Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 285–311.

[34] K. Finstad, Response interpolation and scale sensitivity: evidence against 5-point scales, J. Usability Stud. 5 (3) (2010) 104–110.

[35] B. Kitchenham, S.L. Pfleeger, Principles of survey research: part 5: populations and samples, ACM SIGSOFT Softw. Eng. Notes 27 (5) (2002) 17–20.

[36] S. Ali, S.U. Khan, Software outsourcing partnership model: an evaluation framework for vendor organizations, J. Syst. Softw. 117 (2016) 402–425.

[37] M.A. Akbar, et al., Statistical analysis of the effects of heavyweight and lightweight methodologies on the six-pointed star model, IEEE Access 6 (2018) 8066–8079.

[38] I. Keshta, M. Niazi, M. Alshayeb, Towards implementation of requirements management specific practices (SP1. 3 and SP1. 4) for Saudi Arabian small and medium sized software development organizations, IEEE Access 5 (2017) 24162–24183.

[39] S. Mahmood, S. Anwer, M. Niazi, M. Alshayeb, I. Richardson, Key factors that influence task allocation in global software development, Inf. Softw. Technol. 91 (2017) 102–122.

[40] A.P. Sage, Methodology for Large-Scale Systems, McGraw-Hill, New York, 1977, pp. 165–203.

[41] V. Ravi, R. Shankar, Analysis of interactions among the barriers of reverse logistics, Technol. Forecast. Soc. Chang. 72 (8) (2005) 1011–1029.

[42] G. Kannan, S. Pokharel, P.S. Kumar, A hybrid approach using ISM and fuzzy TOPSIS for the selection of reverse logistics provider, Resour. Conserv. Recycl. 54 (1) (2009) 28–36.

[43] H. Sharma, A. Gupta, The objectives of waste management in India: a futures inquiry, Technol. Forecast. Soc. Chang. 48 (3) (1995) 285–309.

[44] A. Agarwal, P. Vrat, Modeling attributes of human body organization using ISM and AHP, Jindal J. Bus. Res. 6 (1) (2017) 44–62.

[45] T. Raj, R. Attri, Identification and modelling of barriers in the implementation of TQM, Int. J. Prod. Qual. Manag. 8 (2) (2011) 153–179.

[46] K. Yoon, C.-L. Hwang, Manufacturing plant location analysis by multiple attribute decision making: part i—single-plant strategy, Int. J. Prod. Res. 23 (2) (1985) 345–359.

[47] T.-Y. Chen, C.-Y. Tsao, The interval-valued fuzzy TOPSIS method and experimental analysis, Fuzzy Sets Syst. 159 (11) (2008) 1410–1428.

[48] S. Rafi, W. Yu, M.A. Akbar, A. Alsanad, A. Gumaei, Multicriteria based decision making of DevOps data quality assessment challenges using fuzzy TOPSIS, IEEE Access 8 (2020) 46958–46980.

[49] D. Kannan, A.B.L. de Sousa Jabbour, C.J.C. Jabbour, Selecting green suppliers based on GSCM practices: using fuzzy TOPSIS applied to a Brazilian electronics company, Eur. J. Oper. Res. 233 (2) (2014) 432–447.

[50] R.A. Krohling, V.C. Campanharo, Fuzzy TOPSIS for group decision making: a case study for accidents with oil spill in the sea, Expert Syst. Appl. 38 (4) (2011) 4190–4197.

[51] F.T. Bozbura, A. Beskese, C. Kahraman, Prioritization of human capital measurement indicators using fuzzy AHP, Expert Syst. Appl. 32 (4) (2007) 1100–1112.

[52] Gary McGraw, Ph.D., Sammy Migues, and Jacob West. Building Security In Maturity Model (BSIMM) Version 6, 171 Second Street, Suite 300, San Francisco, California, 94105, USA, 2015. https://www.inf.ed.ac.uk/teaching/courses/sp/2015/lecs/BSIMM6.pdf.

[53] S. Salinger, L. Plonka, L. Prechelt, A coding scheme development methodology using grounded theory for qualitative analysis of pair programming, Hum. Technol. 4 (1) (2008) 9–25. http://www.humantechnology.jyu.fi. URN:NBNfi:jyu-200804151350. Retrieved from.

[54] T.H.-C. Hsu, Practical Security Automation and Testing: Tools and Techniques For Automated Security Scanning and Testing in Devsecops, Packt Publishing Ltd, 2019.

[55] B.S. Farroha, D.L. Farroha, A framework for managing mission needs, compliance, and trust in the DevOps environment, in: Proceedings of the IEEE Military Communications Conference, IEEE, 2014, pp. 288–293.

[56] M. Sánchez-Gordón, R. Colomo-Palacios, Security as culture: a systematic literature review of DevSecOps, in: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, 2020, pp. 266–269.

[57] R. Kumar, R. Goyal, Modeling continuous security: a conceptual model for automated DevSecOps using open-source software over cloud (ADOC), Comput. Secur. 97 (2020), 101967.

[58] N. Tomas, J. Li, H. Huang, An empirical study on culture, automation, measurement, and sharing of devsecops, in: Proceedings of the International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2019, pp. 1–8.

[59] Z. Ahmed, S.C. Francis, Integrating security with DevSecOps: techniques and challenges, in: Proceedings of the International Conference on Digitization (ICD), IEEE, 2019, pp. 178–182.

[60] B. Rahul, K. Prajwal, M. Manu, Implementation of DevSecOps using open-source tools, Int. J. Adv. Res. Ideas Innov. Technol. 5 (3) (2019).
[61] M. Zaydi, B. Nassereddine, DevSecOps practices for an agile and secure it service management, J. Manag. Inf. Decis. Sci. 23 (2) (2020) 1–16.
[62] R. Mao, et al., Preliminary findings about DevSecOps from grey literature, in: Proceedings of the IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2020, pp. 450–457.
[63] H. Yasar, Build Secure Application With DevSecOps!, Carnegie-Mellon University, Software Engineering Institute Pittsburgh United …, 2018.
[64] T. Longstaff, H. Yasar, Build Secure Application With DevSecOps!, Carnegie-Mellon Univ Pittsburgh, PA Pittsburgh United States, 2019.
[65] A. Barabanov, A. Markov, A. Fadin, V. Tsirlov, I. Shakhalov, Synthesis of secure software development controls, in: Proceedings of the 8th International Conference on Security of Information and Networks, 2015, pp. 93–97.
[66] A.P. Heiner, N. Asokan, Secure software installation in a mobile environment, in: Proceedings of the 3rd Symposium on Usable Privacy and Security, 2007, pp. 155–156.
[67] M.L. Patten, Questionnaire Research: A Practical Guide, Routledge, 2016.
[68] D. Altman, D. Machin, T. Bryant, M. Gardner, Statistics With Confidence: Confidence Intervals and Statistical Guidelines, John Wiley & Sons, 2013.
[69] D. Trewin, Small Business in Australia: 2001, 1321.0," ed, Australian Bureau of Statistics Report, 2002.
[70] M.A. Akbar, et al., Success factors influencing requirements change management process in global software development, J. Comput. Lang. 51 (2019) 112–130.
[71] M. Soni, End to end automation on cloud with build pipeline: the case for DevOps in insurance industry, continuous integration, continuous testing, and continuous delivery, in: Proceedings of the IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), IEEE, 2015, pp. 85–89.
[72] R. Attri, S. Grover, N. Dev, D. Kumar, Analysis of barriers of total productive maintenance (TPM), Int. J. Syst. Assur. Eng. Manag. 4 (4) (2013) 365–377.
[73] J.N. Warfield, Developing interconnection matrices in structural modeling, IEEE Trans. Syst. Man Cybern. (1) (1974) 81–87.
[74] F.R.L. Junior, L. Osiro, L.C.R. Carpinetti, A comparison between fuzzy AHP and fuzzy TOPSIS methods to supplier selection, Appl. Soft Comput. 21 (2014) 194–209.
[75] C.-.N. Liao, H.-.P. Kao, An integrated fuzzy TOPSIS and MCGP approach to supplier selection in supply chain management, Expert Syst. Appl. 38 (9) (2011) 10803–10811.
[76] A. Zouggari, L. Benyoucef, Simulation based fuzzy TOPSIS approach for group multi-criteria supplier selection problem, Eng. Appl. Artif. Intell. 25 (3) (2012) 507–519.
[77] D. Yong, Plant location selection based on fuzzy TOPSIS, Int. J. Adv. Manuf. Technol. 28 (7) (2006) 839–844.
[78] L. Prates, J. Faustino, M. Silva, R. Pereira, Devsecops metrics, in: Proceedings of the EuroSymposium on Systems Analysis and Design, Springer, 2019, pp. 77–90.