Contents lists available at ScienceDirect

Applied Soft Computing

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

Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis

Saima Rafi [a,*], Muhammad Azeem Akbar [b,*], Wu Yu [c], Ahmed Alsanad [d,*], Abdu Gumaei [d], Muhammad Umer Sarwar [e]

[a] *University of Murcia, Department of Informatics and Systems, 30100 Murcia, Spain*
[b] *Lappeenranta-Lahti University of Technology, Department of Information Technology, 53851 Lappeenranta, Finland*
[c] *School of Cyber security and Information Law, Chongqing University of Posts and Telecommunication Chongqing 400065, China*
[d] *STC's Artificial Intelligence Chair, Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia*
[e] *College of Computer Science and Information Studies, Government College University, Faisalabad 38000, Pakistan*

ARTICLE INFO

Article history:
Received 8 October 2020
Received in revised form 15 December 2021
Accepted 19 December 2021
Available online 28 December 2021

Keywords:
DevOps
Testing capabilities
ISM
Fuzzy TOPSIS
MICMAC

ABSTRACT

DevOps is an emerging paradigm that refer to a collaborative culture of development and operation teams aiming to develop the high quality software product. Software organizations are adopting DevOps culture for software development and easy maintenance instead of using traditional SDLC mechanism. To enter the production stage, in DevOps process, the software product have to pass through quality gates were the software are tested during development phase to meet the established targeted criteria. This indicates that the mechanism of testing in DevOps process is not straightforward, and to establish strong DevOps testing platform there is a need to explore more automated testing practices. Thus, using multivocal literature review approach, we have selected 39 studies and identify the 20 testing capabilities. Finally, the interpretive structure modeling (ISM) and fuzzy technique for order preference by similarity to ideal solution (fuzzy TOPSIS) were applied. The results shows that (C2, CCi=0.808; C6, CCi=0.720; and C3, CCi=0.705) are top ranked testing capabilities. Using analysis results, we develop a holistic structure of testing capabilities to show their inter-relationship with each other and their priorities to select the best testing capabilities for DevOps process.

1. Introduction

The software firms are searching for new ways to improve their activities of software development in order to meet the demands of business and market values [1]. To this end, various firms are adopting DevOps due to the pressure derived from rapidly changing business environment. According to the State of DevOps report 2019 [2], the DevOps is spreading worldwide with the rate of more than 10,000 industries from the year 2018 to 2019. The adoption of DevOps in software firms increase project deployment rate more than thousand times per day for example, after adopting DevOps companies like Netflix, Google and Amazon commit to deploy thousand times faster deploy time than before. This report also states that the firms using DevOps experience 60 times fewer failures and recovery time 168 times faster, as compared to the firms without DevOps cycle [2]. The adoption of DevOps assists to merge the gap between development and operations teams [3,4] and possibly will support other dimensions i.e., quality of software, security, testing, performance and stability of the services [5].

The DevOps is a collaboration of both development and operation teams aiming to develop the quality projects in-time and budget. Moreover, Jabbari et al. [6] defines DevOps as "methodology used to bridge the gap between development and operation teams to improve communication, continuous integration and delivery, quality management and automated deployment practices. The perspective of DevOps defined by Smeds et al. [7] is widely supported by research community that includes three attributes to define DevOps, namely "culture enabler, capabilities and technology enabler". They stated that "to enhance the capabilities such as continuous testing, planning, continuous deployment, continuous integration and release, continuous infrastructure monitoring, feedback and service failure rates; process management and testing strategies are required in different areas of software process improvements". Thus, proper testing strategies and process management activities are required to improve the testing and processing techniques in software firms working within DevOps environment.

* Corresponding authors.
E-mail addresses: saima.rafi@um.es (S. Rafi), azeem.akbar@lut.fi (M.A. Akbar), aasanad@ksu.edu.sa (A. Alsanad).

https://doi.org/10.1016/j.asoc.2021.108377
1568-4946/© 2021 Elsevier B.V. All rights reserved.

As noted by [6], despite the importance of DevOps in software industry, testing environment in DevOps has not yet received much attention by scientific community. As for the most important key dimensions for DevOps, some attention has been devoted to quality assessment [8,9], security [10–12] and performance measures [13,14], but limited research has been conducted to address the testing concern of DevOps process. This indicates testing as a weak area of research in DevOps and needed to be improved to make DevOps process more effective. With the motivation of this research gap, this study explore and analyze the most significant capabilities of testing that can improve DevOps testing process. We believe that better understanding of testing capabilities will help the experts to improve their testing strategies related to DevOps; for better testing performance. This study has three main objectives: (i) identification of testing capabilities for DevOps from literature (ii) analyze the interaction among the capabilities in the form of model to get holistic view of testing capabilities for DevOps process and (iii) determine the relative importance of each capability for the selection process (of best capabilities) based on their significance importance to improve DevOps process. The findings of study will assist software firms in handling problems faced by testing teams and helping them in refining testing strategies based on significant testing capabilities for DevOps process.

The rest of the paper is organized as follows: Background and motivation is presented in Section 2, Section 3 presents the detail about selected research methodologies. Section 4 contains detail about results of this study and the summary and discussion of research objectives are discussed in Section 5. The study implications and study limitation are discussed in Sections 6 and 7. Finally, conclusion and future directions are summarized in Section 8.

2. Background and motivation

Testing is conspicuous by its lack of presence from most of the existing DevOps literature. The literature focus more on development and operational activities of DevOps. Understandably, this gap is creating nervousness in testers about their role in a team. The experts involved in shaping problems of DevOps fails to highlight testing approaches because they are not from professional background of testing. This does not means that there is no professional testing activities involved in DevOps, only development team is controlling testing, but it is considered as a passive activity. Humble and Farley [15] defines testing as "a cross functional activity that involves the whole team, and should be done continuously from the beginning of the software project". Clokie [16] also pointed out that it is very difficult for the testers to identify their role in DevOps process, when testing is a part of every activity during software development process. According to Clokie, changing the whole infrastructure for DevOps testing depends on three main aspects i.e., organization vision, existing test approaches and collaboration between the teams. To perform such activities people from testing backgrounds must be a part of DevOps team.

Zimmerer [17] stated two main characteristics of continuous testing in industrial DevOps (iDevOps); (i) continuous testing from beginning to the end and (ii) strategic test automation. Bertolino et al. [18] proposed DevOpRET approach for reliability testing (as a part of acceptance testing) to make DevOps testing activities more reliable. Faber [19] explains the significance of testing in DevOps by claiming that more automated tests must be performed to make DevOps cycle strong. Pietrantuono and Russo [20] worked on software reliability testing in DevOps by integrating acceptance-testing stage before each next release to production. Furthermore, they performed case study to evaluate the reliability assessment evolves over subsequent releases. Although, there are many challenges of DevOps testing encountered by experts in software firms. Cruzes et al. [21] performed empirical investigation to investigate testing challenges and trends in DevOps. They have discussed five major concerns for testers in DevOps that are "coordination, roles and responsibilities of team members, automation and monitoring, types of tests and infrastructure", and proposed a conceptual architecture for continuous testing. However, this study has major contribution in field of testing DevOps environment but missed the major testing capabilities, which are important in improving performance of testing in DevOps. This research gap motivated us to identify testing capabilities essential to improve DevOps testing performance in software firms.

State-of-the-art review of existing literature indicated that experts are trying to bring testing to another level of automation in DevOps but still limited consideration is made to establish DevOps testing platform in software firms. In addition, to date, there is no conceptual model based on testing capabilities for DevOps that could help practitioners towards adoption of effective testing strategies for better performance in software firms. Therefore, there is a need for study that could help industrial experts to understand DevOps testing process in software firms. This motivated us to explore testing capabilities that plays an essential role in improving DevOps testing process in software development units. In this study, we have applied multivocal literature review (MLR) approach to identify the testing capabilities related to DevOps. The interaction among the testing capabilities was analyzed by using Interpretive Structural Modeling (ISM) technique and a holistic model is proposed. The model demonstrates the driving and dependence relationship between the identified capabilities. Finally, the fuzzy TOPSIS is adopted to determine the relative importance of testing capabilities for effective and efficient performance of DevOps in software firms.

Various studies has adopted ISM and multi-criteria decision-making (MCDM) approach for calculating the relationship among attributes and prioritizing the attributes for example, Kannan et al. [22] analyzed the interaction between the green suppliers (who address the environment performance) by using ISM and AHP approach. Ravi and Shankar [23] discussed eleven challenges of reverse logistic in automobile industries and analyze the interaction among challenges using ISM approach. Parthiban et al. [24] worked on factors affecting the supplier selection problem and presented conceptual model by using ISM and AHP approach. Agarwal and Vrat [25] model the attributes of human body organization using ISM and AHP approach. Kannan et al. [26] used a hybrid approach i.e. (ISM and fuzzy TOPSIS) for the selection of reverse logistic providers in fuzzy environment. We have applied the same approach as adopted by [26] to present a holistic view of testing capabilities showing their driving and dependence interaction with other capabilities. We also prioritize the testing capabilities to determine their relative importance with respect to DevOps testing performance. The prioritization of testing capabilities is based on CAMS model (principles of DevOps) [27]. This study will assist the software firms to establish new testing strategies for better testing performance in DevOps process.

3. Methodology

To fulfill the aim of this study, the research has been designed in three different steps. The multivocal literature review (MLR) was adopted to identify the capabilities of testing for DevOps process. Moreover, based on the findings of MLR we have applied ISM technique to analyze the interaction among the capabilities in form of model to get holistic view of testing capabilities for DevOps. Finally, Fuzzy TOPSIS approach was applied to determine the relative importance of these capabilities for effective testing. Fig. 1 shows the detailed overview of methodology which is briefly discussed in subsequent sections:

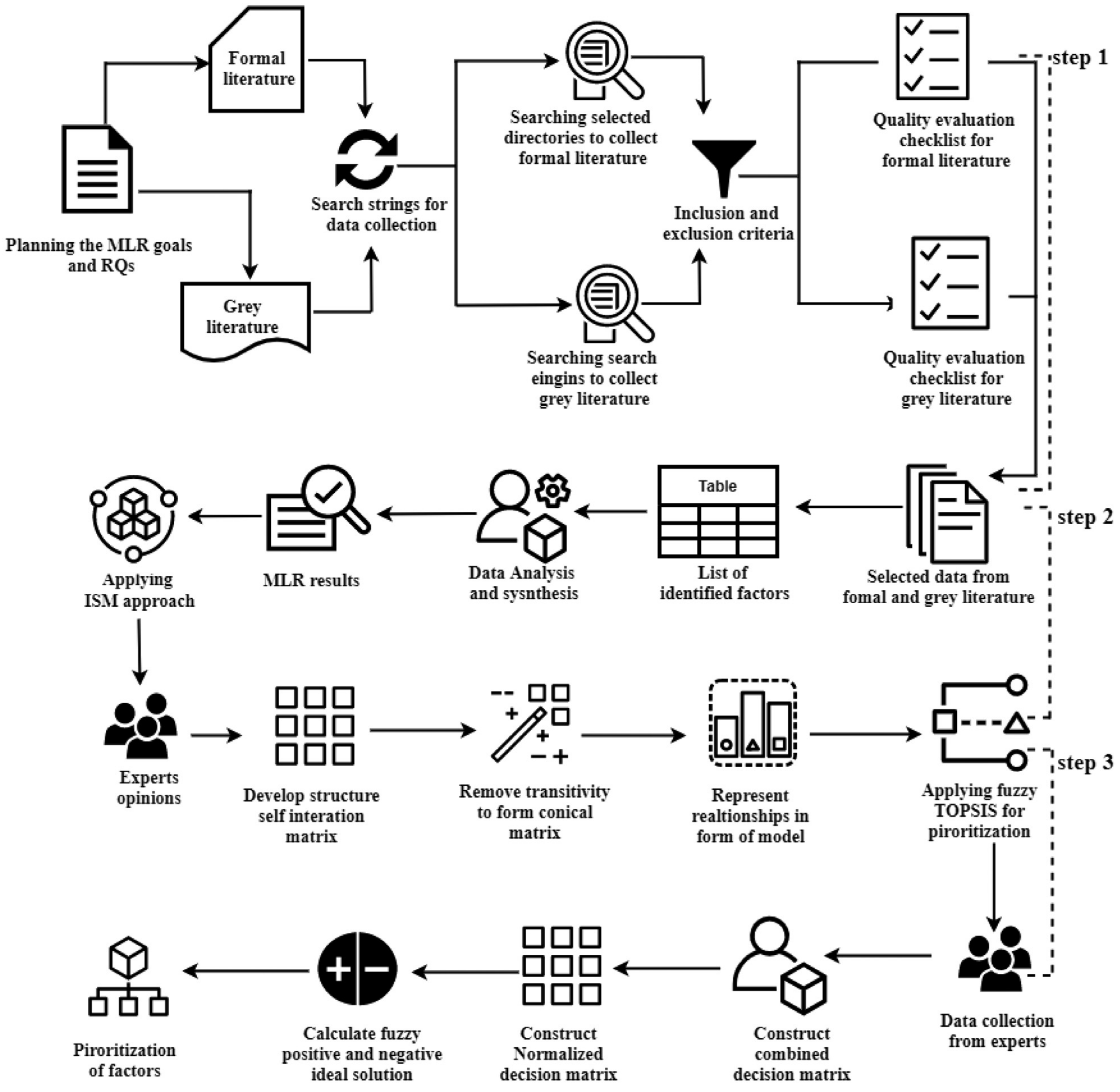
Formal literature
Planning the MLR goals and RQs
Grey literature
Search strings for data collection
Searching selected directories to collect formal literature
Searching search eingins to collect grey literature
Inclusion and exclusion criteria
Quality evaluation checklist for formal literature
Quality evaluation checklist for grey literature
step 1
step 2
Selected data from fomal and grey literature
Table
List of identified factors
Data Analysis and sysnthesis
MLR results
Applying ISM approach
Experts opinions
Develop structure self interation matrix
Remove transitivity to form conical matrix
Represent realtionships in form of model
Applying fuzzy TOPSIS for piroritization
step 3
Data collection from experts
Construct combined decision matrix
Construct Normalized decision matrix
Calculate fuzzy positive and negative ideal solution
Piroritization of factors

Fig. 1. Overview of methodology.

3.1. Multivocal literature review

MLR is considered as an effective method to explore the both types of literature (grey and formal) related to the study objectives [28,29]. The grey literature consists of white paper, blogs, interviews, case studies and experienced reports of academia and industry in both private and government sectors where publication is not the primary objectives. The formal literature refers to all pre-reviewed published studies in main stream journals or conferences. The testing in DevOps process is particularly industrial oriented activity of software engineering domain, therefore, the inclusion of grey literature is essential to explore the testing capabilities for DevOps in software firms. To perform MLR we followed the guidelines of Garousi et al. [28] that are, "to conduct MLR there are three core steps (i) planning, (ii) conducting and (iii) reporting". The brief description of all steps of MLR is explained in subsequent sections.

3.1.1. Step-1: Planning the MLR

To collect the data from literature, the following research question (RQ) was proposed.

RQ 1: What are the testing capabilities for DevOps reported in the literature, that are essential to improve DevOps testing performance?

The protocols of MLR approach were designed before starting the review process. These protocols were developed by research authors (first three authors) of this study. The pilot assessment was performed with the research team and research advisor to validate the review protocols. They recommended some changes in protocols. We update the protocols according to their recommendations; and used these protocols to conduct this study. All protocols are briefly discussed in following section.

3.1.2. Step-2: Data collection

To collect data from pre-reviewed published studies, the following steps were performed.

Table 1
Quality evaluation checklist for formal literature.

QE#	Quality evaluation questions	Grading scale
QE.1	Does the adopted research method address the proposed research question?	[agree = 1, neutral = 0.5, disagree = 0]
QE.2	Does the study discuss any testing capability for DevOps?	[agree = 1, neutral = 0.5, disagree = 0]
QE.3	Does the study discuss any approach for effective testing in DevOps?	[agree = 1, neutral = 0.5, disagree = 0]
QE.4	Is the selected study related to testing in DevOps?	[agree = 1, neutral = 0.5, disagree = 0]
QE.5	Are the identified results related to justify research question?	[agree = 1, neutral = 0.5, disagree = 0]

Table 2
Quality evaluation checklist for grey literature.

QE#	Quality evaluation questions	Grading scale
QE1	Is the publishing organization reputable? For example, the software engineering institute (SEI)	[agree = 1, neutral = 0.5, disagree = 0]
QE2	Is the author have affiliation with a reputed software organizations?	[agree = 1, neutral = 0.5, disagree = 0]
QE3	Does the author published other similar work?	[agree = 1, neutral = 0.5, disagree = 0]
QE4	Does the author has experience in subject domain?	[agree = 1, neutral = 0.5, disagree = 0]
QE5	Does the aim of study clear in literature?	[agree = 1, neutral = 0.5, disagree = 0]
QE6	Are the limitations of the data clearly stated?	[agree = 1, neutral = 0.5, disagree = 0]
QE7	Do authoritative, contemporary references support the source?	[agree = 1, neutral = 0.5, disagree = 0]
QE8	Is the work referring to a specific case or population?	[agree = 1, neutral = 0.5, disagree = 0]
QE9	Does data covers specific domain?	[agree = 1, neutral = 0.5, disagree = 0]
QE10	Does the selected data presented in balance form?	[agree = 1, neutral = 0.5, disagree = 0]
QE11	Does the statement contains subjective opinion?	[agree = 1, neutral = 0.5, disagree = 0]
QE12	Is there any tool comparison by author working in particular firm?	[agree = 1, neutral = 0.5, disagree = 0]
QE13	Does the summarized conclusion cover the data?	[agree = 1, neutral = 0.5, disagree = 0]
QE14	Is there any defined date on data?	[agree = 1, neutral = 0.5, disagree = 0]
QE15	Does the data related to the developed research question?	[agree = 1, neutral = 0.5, disagree = 0]

Selection of data sources for formal and grey literature: As defined by Afzal et al. [29] "searching of appropriate published literature in digital repositories is critical to effectively address the study objectives". For the selection of potential literature, the selection of appropriate digital libraries and search engines, the suggestion of Khan et al. [30], Akbar et al. [31] and Chen et al. [32] were considered. The selected data sources for formal literature and grey literature are presented in Fig. 2.

Development of search string

To explore the literature from the selected studies, the search string was developed using the key terms and their alternatives. The key terms and their alternatives for search strings were derived from existing research studies in field of DevOps testing [3,4,6,7,9,13,20]. The search string is divided into four parts:
Part1 (P1): Keywords and their alternates related to capability.
Part2 (P2): Keywords and their alternates related to testing.
Part3 (P3): Keywords and their alternates related to DevOps.
Part4 (P4): Keywords and their alternatives related to software development organization.

These keywords and their alternates were connected by using Boolean operators "AND" and "OR" to formulate the search strings according to the syntax requirements of particular digital library.

Search string = (P1 AND P2 AND P3 AND P4)

An example of complete search string is:
("capability" OR "ability" OR "power" OR "potential" OR "competence" OR "proficiency") AND ("testing" OR "examination" OR "assessment" OR "investigation" OR "inspection" OR "analysis" OR "screening") AND ("DevOps" OR "continuous deployment unit" OR "continuous integration" OR "development and operation" OR "continuous delivery process" OR "continuous development" OR "DevTestOps", OR "continuous testing process") AND ("software development organizations" OR "software development firms" OR "software engineering industry" OR "software team" OR "development companies" OR "software engineering firms" OR "development and processing units")

Inclusion criteria:

The primary studies articles were selected based on given inclusion criteria [29,33].

- The data must be related to selected topic.
- The data that described DevOps testing in software organizations.
- The data that described the testing capabilities for DevOps process.
- The data that described motivation to improve DevOps testing process.
- Findings of data must be based on real world experiments.
- The language of searched material must be English.

Exclusion criteria:

The exclusion criteria are given below:

- The data that does not relate to the research question.
- The data that does not described testing strategies for DevOps.
- The data that does not provide detailed information relevant to the testing capabilities for DevOps process.
- In case of similar findings, from same research group, we will consider the latest published results.
- Data without authentic source.
- Data that do not have scientific results of real-world practices.
- Data that do not provides details of findings.

Quality evaluation: the quality evaluation (QE) of formal data (primary selected studies) was performed using guidelines of Kitchenham and Charters [34] and for grey literature; we followed the guidelines of Garousi et al. [31]. To evaluate the suitability of selected literature for addressing the developed RQ we have performed QE process. The checklist designed to measure the quality of selected primary studies is summarized in Tables 1 and 2. The evaluation is made for each selected study based on response grading (against each question) mentioned in Tables 1 and 2. This criteria was also adopted in other studies for quality evaluation process [29–31]. The detail about QE outcomes of formal and grey literature are presented in Appendices A and B.

Selected studies:

The selected studies were further processed through tollgate approach [35], which consists of five phases as discussed in Fig. 2.

We collected total of 50 published articles for the formal literature and 30 pieces of the grey literature using the designed

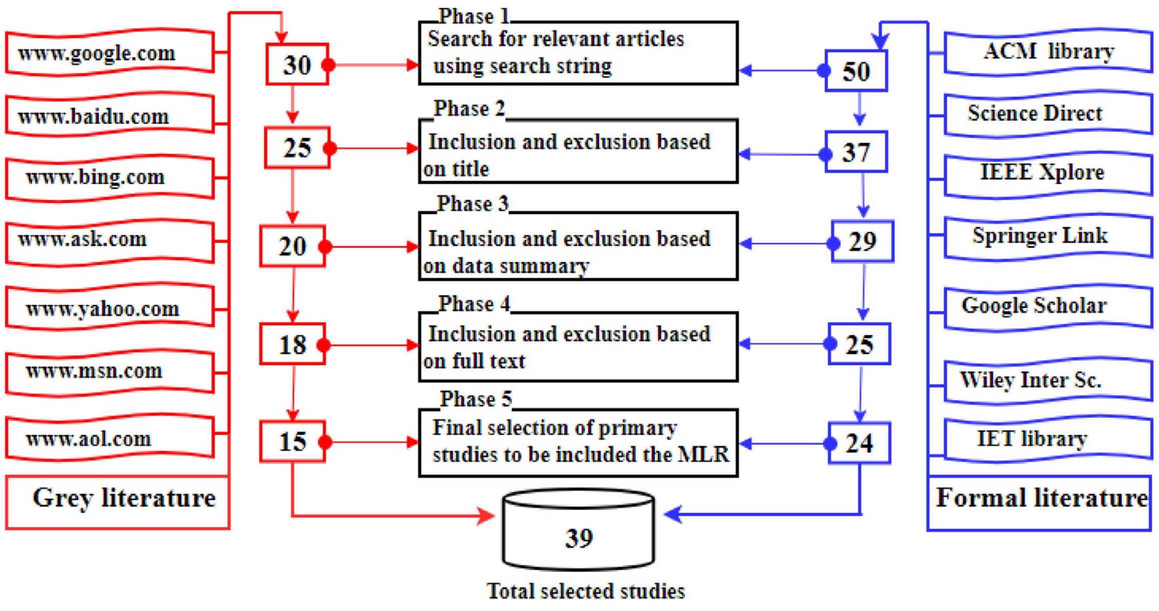
www.google.com
www.baidu.com
www.bing.com
www.ask.com
www.yahoo.com
www.msn.com
www.aol.com
Grey literature
30
25
20
18
15
Phase 1 Search for relevant articles using search string
Phase 2 Inclusion and exclusion based on title
Phase 3 Inclusion and exclusion based on data summary
Phase 4 Inclusion and exclusion based on full text
Phase 5 Final selection of primary studies to be included the MLR
50
37
29
25
24
ACM library
Science Direct
IEEE Xplore
Springer Link
Google Scholar
Wiley Inter Sc.
IET library
Formal literature
39
Total selected studies

Fig. 2. Selected primary studies from formal and grey literature.

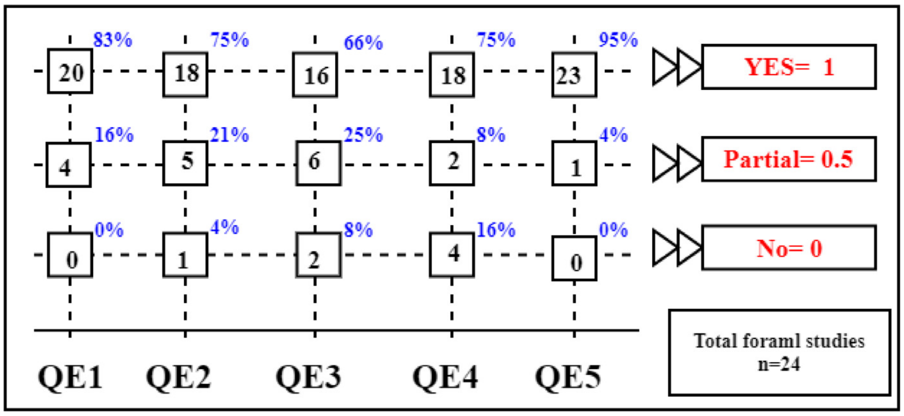

Fig. 3. QE results of formal literature.

search string and by following the inclusion and exclusion criteria. We performed all phases of tollgate approach suggested by Afzal et al. [30], to refine the primary studies. The shortlisted 39 primary studies after performing all phases are presented in Fig. 2. Moreover, the shortlisted data was evaluated further using QE criteria. The final list of shortlisted studies is presented in Appendices A and B, we used the code "SP" to represent each selected primary study.

Data extraction and synthesis: For data extraction process after the final selection of literature, the basic (ideas, concepts, findings etc.) for each study is collected. To answer the research question, we considered total of 39 studies for data extraction process.

Data extraction is performed by following the Kitchenham and Charters [34] instructions. To extract testing capabilities for DevOps, we have used grounded theory approach [36]. The codes, categories and sub-categories were carefully defined from the selected studies to systematically record the extracted data. The list of testing capabilities extracted from both set of data (formal and grey literature) are presented in Table 4. In execute this process, first three authors of this study participated and the list of testing capabilities were defined. The other two counter check the data extraction process to remove bias and to validate the testing capabilities for DevOps process. Furthermore, we have applied inter-rater reliability test to remove inter-person bias. To perform this process we invited team of external experts (four members). The team members were from well know labs of "COMSAT University Lahore, Pakistan" and "King Saud University Saudi Arabia". After performing all steps of tollgate approach on eight randomly selected studies, we have calculated a non-parametric Kendall coefficient of concordance (W) [30,35]. The value of $W = 1$ indicates "perfect agreement" while the value of $W = 0$ indicates "no agreement" between external and study authors. Though, the outcomes of $W = 0.83$ where $p = 0.003$ for eight randomly selected studies, shows that there is acceptable agreement among externals and authors of study.

3.1.3. Step 3: Reporting the review

Quality evaluation of the selected literature: The QE score is calculated for each shortlisted primary study based on QE checklist developed in Section 3.1.2. The list of primary studies with QE outcomes are presented in Appendices A and B. To exclude the study, we have used 50% quality evaluation value as a threshold. The summarized QE results are shown in Figs. 3 and 4. The Fig. 3 indicates that QE1 is scored 1 by 80% of studies, 0.5 by 16% studies and scored 0 by none of the formal studies. Similarly, Fig. 4 indicates that QE1 is scored 1 by 80% of grey studies, 0.5 by 13% of grey studies and scored 0 by none of the grey studies. The same process is adopted to check the percentage of each

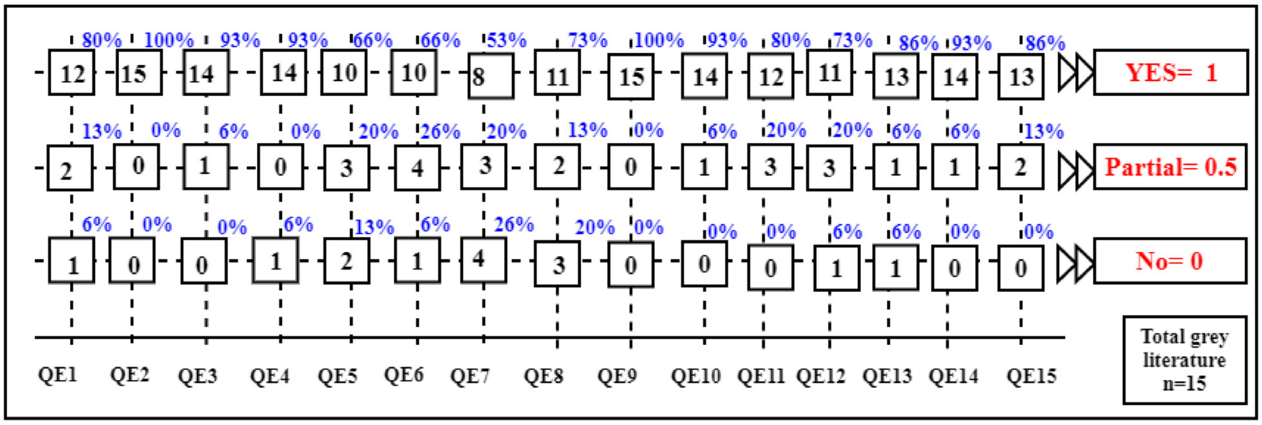

Fig. 4. QE results of grey literature.

QE (checklist question). The cumulative quality score shows that all the studies score greater than 50%, which indicates that the selected literature is relevant to address the research question of the study.

3.2. ISM methodology

ISM approach has been applied to analyze and summarize the interdependencies between different testing capabilities for DevOps that were identified from literature. This approach is developed by Warfield [37] to help individual and groups to construct a roadmap of complex relationships between multiple elements, involved in a complex situation. The general concept of ISM is "to decompose the complicated system into several subsystems (elements) by using experts' practical experience and knowledge in order to construct a multilevel structural model" [38]. Kannan et al. [26] used ISM approach for selection of reverse logistics providers. Ararwal and Vrat [25] used ISM for modeling the attributes of human body organization. Kannan and Haq [39] applied the ISM technique to analyze the relationship between criteria and sub-criteria which influence the supplier selection to develop supply chain environment. Various other researchers [40–42] have deployed ISM technique in their research successfully for analyzing the problems and developing the conceptual frameworks.

The ISM technique has some limitations. The contextual relationship among the variables mostly depends on user knowledge and their work experience in particular domain [26]. Therefore, this personal bias of judgment might influence the outcome. ISM does not give any weighting associated with variables to analyze their ranking at particular level. This section explains all steps adopted for ISM approach to find the relationship between identified testing capabilities for DevOps in a case to improve DevOps testing performance. The steps involved in the ISM technique are discussed as follows:

Step1. Identify and define the variables (capabilities) to making DevOps testing strategies more organized, controlled and directed for better performance measures.
Step2. The variables identified in step 1 are analyzed further to develop a relationship among variables in order to categories the variables into groups for evaluation.
Step3. Compute a structural self-interaction matrix (SSIM) for variables under consideration through opinion of experts in particular field.
Step4. Transform the SSIM to initial reachability matrix by replacing the entries of SSIM into '1s' and '0s'.
Step5. Check the transitivity of the contextual relation that is used to make basic assumptions made in ISM to reach final stage of reachability matrix. The transitivity is defined as "if a variable A = B and B = C, then also A = C".
Step6. The level partitioning of final reachability matrix into different levels.
Step7. Develop a conical matrix by clustering the factors at same level (across columns and rows) in final reachability matrix. To calculate the driving power of variables add up all number of 1's in rows and to obtain dependence power add up all number of 1's in columns. Rank the driving and dependence power in rows and columns by giving highest ranks depending upon the number of 1's respectively.
Step8. From conical form of reachability matrix, the preliminary diagraph including transitive links is obtained. The final form of diagraph is developed by removing indirect links.
Step9. The ISM model developed in step 8 is reviewed to check the conceptual inconsistency and necessary modifications are made.
Step10. MICMAC analysis to categorize the variables into various clusters considering their driving and dependence power.

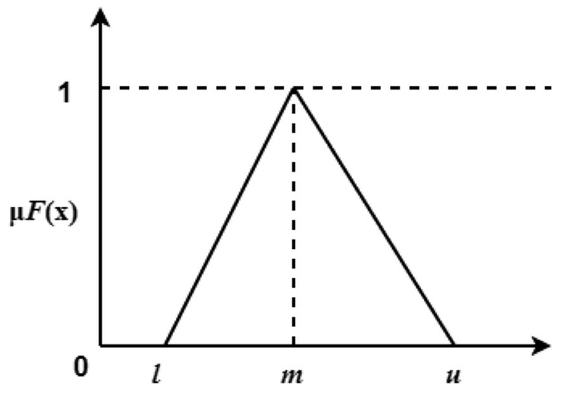

Fig. 5. Triangular Fuzzy Number (TFN) graphical form.

3.3. Fuzzy Set theory

The fuzzy set theory is used to remove the vagueness or uncertainty caused by human thoughts in MCDM (multi-criteria decision-making) problems. This theory was introduced by Zadeh et al. [43]. The definition of fuzzy set theory is:

"A triangular fuzzy number (TFN) is denoted by (l, m, u). The values of l, m and u parameters denotes the smallest value, the most promising value and the largest possible value that describes the fuzzy event. In fuzzy set, the membership function $\mu F(\mathrm{x})$ maps object between 0 and 1" as illustrated in Fig. 5.

Table 3
Triangular fuzzy scale by Bozbura et al. [8].

Linguistic variables	Triangular fuzzy scale	Triangular fuzzy reciprocal scale
Just Equal (JE)	1, 1, 1	1, 1, 1
Equally Importance (EI)	1/2, 1, 3/2	2/3, 1, 2
Weakly Importance (WI)	1, 3/2, 2	1/2, 2/3, 1
Strongly More Importance (SMI)	3/2, 2, 5/2	2/5, 1/2, 2/3
Very Strongly More Importance (VSMI)	2, 5/2, 3	1/3, 2/5, 1/2
Absolutely More importance (AMI)	5/2, 3, 7/2	2/7, 1/3, 2/5

The value (1, 1, 1) represents "Just equal" showing that two factors contributes equally to the objective.

3.3.1. Fuzzy TOPSIS

The fuzzy TOPSIS is used to measure group decision-making problems on a subject matter in a fuzzy environment. Generally, MCDM involves various techniques such as max–min, min–max, ELECTRE, PROMETHEE, TOPSIS, fuzzy TOPSIS, compromise programming, analytical hierarchy process (AHP) and fuzzy AHP that can be used for prioritization and comparing multiple alternatives [44]. Among these techniques, to resolve fuzzy group decision-making problems that contain vague and incomplete data and linguistic variables, fuzzy TOPSIS has gained growing attention [44,45]. On the other hand, the other techniques are not suitable to make quick decision on real life problems [45], as we are facing in this study i.e. selecting best testing capabilities for continuous software development environment (DevOps). This motivated us to select fuzzy TOPSIS approach for the ranking of testing capabilities. Chen [46] proposed the fuzzy TOPSIS approach to solve MCDM problems under uncertainty. The linguistic variables are used by decision-makers to access the weight of criteria. These linguistic variables are converted into triangular fuzzy numbers (TFNs) to control vagueness of linguistic assessment [46]. This technique is the most classical approach to resolve problems of group decisions by calculating the positive and negative ideal solutions. The value of solution if have shortest distance from positive ideal solution; and large distance from negative ideal solution is the best alternative. The effectiveness of fuzzy TOPSIS approach motivated us to use this approach, to prioritize the capabilities of DevOps testing in fuzzy environment based on CAMS criteria [27], and the significance of capabilities at particular levels of developed ISM model. This technique has been adopted in various studies for example; Rostamzadeh and Sofian [47] used fuzzy TOPSIS and fuzzy AHP approach to improve the performance values of organization's production system. Rafi et al. [8] used fuzzy TOPSIS method to prioritize the DevOps data quality assessment challenges based on DevOps principles model i.e., "culture, automation, measurement and sharing". Kannan et al. [26] applied fuzzy TOPSIS and ISM approach to select reserve logistics providers. Some general definitions of fuzzy sets, fuzzy numbers and linguistic variables are reviewed from Cheng and Lin [48] and Kim et al. [49].

An algorithm of required fuzzy decision-making approach (fuzzy-TOPSIS) for dealing with prioritization of capabilities of DevOps testing is given : below:

Step1. Form a committee of decision-makers to rate the alternatives and their defined criteria.

Step2. Select appropriate linguistic variables to evaluate rating of each alternative and weighted criteria by using triangular fuzzy scale developed by Bozbura et al. [8] in Table 3. Assuming that the group of k decision-makers $(M_1, M_2, \ldots M_k)$ containing r alternatives $(A_1, A_2, \ldots A_r)$ and n criteria $(C_1, C_2, \ldots C_n)$.

$$M = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_r \end{array} \begin{array}{cccc} C_{11} & C_{12} & \ldots & C_n \\ R_{11} & R_{12} & \ldots & R_{1n} \\ R_{21} & R_{22} & \ldots & R_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ R_{r1} & R_{r2} & \ldots & R_{rn} \end{array} \tag{1}$$

where rating of all alternatives is represented by R_{rn}.

Step3. Transform the linguistic term into fuzzy numbers and compute the aggregate fuzzy rating for each criteria.

Step4. Construct the normalized decision-matrix $\tilde{N}$ by using the formula in Eqs. (2) and (3)

$$\tilde{N} = [n_{ab}]_{m*n}$$

Where $a = 1, 2, 3 \ldots, m$ and $b = 1, 2, 3 \ldots n$

$$\tilde{n}_{ab} = \left(\frac{l_{ab}}{u_b^*}, \frac{m_{ab}}{u_b^*}, \frac{u_{ab}}{u_b^*}\right) \text{ and } u_b^* = \max u_{ab} \text{ "benefit criteria"} \tag{2}$$

$$\tilde{n}_{ab} = \left(\frac{l_b^-}{u_{ab}}, \frac{i_b^-}{m_{ab}}, \frac{l_b^-}{l_{ab}}\right) \text{ and } l_b^- = \min l_{ab} \text{ "cost criteria"} \tag{3}$$

Step5. Compute weighted normalized matrix $\widetilde{W}$. The weighted-normalized decision-matrix is calculated by using Eq. (4).

$$\widetilde{W} = [\widetilde{w}_{ab}]\, r * n \quad a = 1, 2, 3 \ldots, r \text{ and } b = 1, 2, 3 \ldots n \tag{4}$$

where, $\widetilde{w}_{ab} = \tilde{n}_{ab} {}^* v_b$, v_b is the weight assigned to each criteria.

Step6. Determine the fuzzy positive ideal solution (FPIS) and fuzzy negative ideal solution (FNIS) from weighted-normalized decision-matrix.

$$V+ = \{w_1^*, \ldots, w_n^*\}, \text{ where } w_b^* = \{\max(w_{ab})\},$$
$$a = 1, 2, 3 \ldots, r \text{ and } b = 1, 2, 3 \ldots, n \tag{5}$$
$$V^- = \{w_1^-, \ldots, w_n^-\}, \text{ where } w_b^- = \{\min(w_{ab})\},$$
$$a = 1, 2, 3 \ldots, r \text{ and } b = 1, 2, 3 \ldots, n \tag{6}$$

Step7. Compute the Euclidean distance from fuzzy positive and negative ideal solution for each alternative by using Eqs. (7) and (8).

$$\widetilde{D}^* x_i = \sqrt{\sum_j^n (w_{ab} - w_b^*)^2} \quad j = 1, 2, 3 \ldots n \tag{7}$$

$$\widetilde{D}^- x_i = \sqrt{\sum_j^n (w_{ab} - w_b^-)^2} \quad j = 1, 2, 3 \ldots n \tag{8}$$

Step8. Determine the closeness coefficient of each alternative using Eq. (9).

$$\mathrm{CCi} = \frac{D_i^- X_i}{D_{iX_i}^- + D_{iX_i}^+} \quad \text{where } i = 1, 2, 3 \ldots m \tag{9}$$

Step9. Rank the alternatives status depending upon their values of CCi. The alternative near to positive ideal solution is the best alternative.

4. Results and discussion

4.1. Findings of MLR

The findings of MLR study are presented in this section.

By applying MLR approach, we have identified a list of 20 testing capabilities for DevOps process that will help software

Table 4
List of identified testing capabilities.

C#	Testing capability	Frequency analysis	Percentage $n = 39$
C1	Monitoring as testing	22	56
C2	Environment management	20	51
C3	Destructive testing	17	43
C4	Testing logging	10	25
C5	Traceability management in testing	4	10
C6	Unit testing	15	38
C7	A/B testing	8	21
C8	Bug bash	7	18
C9	Containers as testing	12	30
C10	Beta testing	3	7
C11	Dark launching	6	15
C12	System testing	8	21
C13	Staged rollout	4	10
C14	Canary release	10	25
C15	Smoke testing	3	7
C16	Acceptance testing	6	15
C17	Infrastructure testing	5	13
C18	Feature toggles	4	10
C19	Deployment pipelines	3	7
C20	Testing backend integrations	14	36

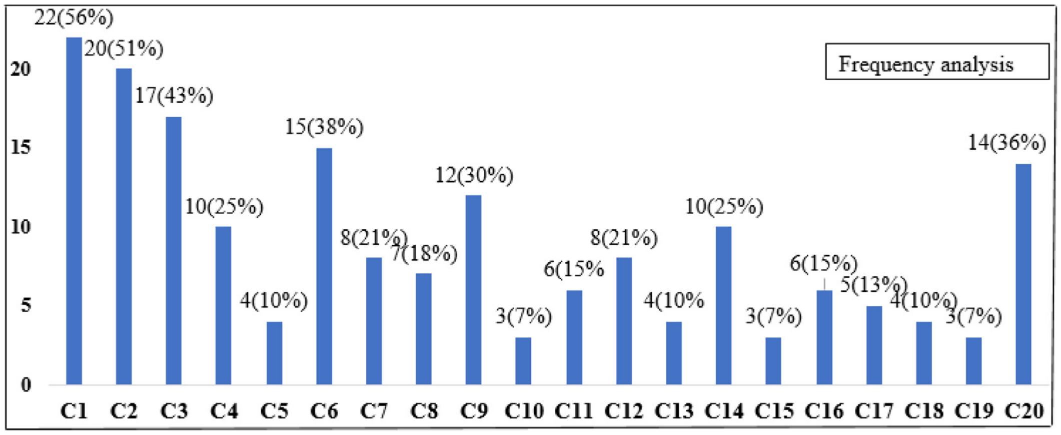

Fig. 6. Graphical distribution of testing capabilities.

experts to establish new testing strategies in their organization for better testing performance. The final list of shortlisted capabilities (from both formal and grey literature i.e., $n = 39$) are presented in Table 4 with frequency analysis to satisfy the research question. The code (C1–C20) was given to each testing capability to perform further analysis. The graphical distribution of capabilities is shown in Fig. 6.

According to the outcomes of MLR, C1 "monitoring as testing" was reported as the most significant capability for the effective testing in DevOps. Clokie [16] reported that millions of people are connected around the world (offline and online) to perform various task in their organizations. Only by monitoring, we can observe unusual behaviors in a continuous processing environment, which we cannot detect through measuring root cause of change. Moreover, Lwakatare et al. [50] suggested monitoring as a main dimension to make DevOps testing successful.

C2 "environment management" was considered as the second most important testing capability for making DevOps process effective. Each unit in production line has different set of users interacting with it for specific purpose. Instead of having dedicated test environments for all teams in DevOps, different teams can create and destroy their own (virtual) environments as required. The testing can be performed on newly created environment and after test execution, the problem discovered can be resolved easily [51]. Introduction of more environments in continuous processing units, may cause problems to manage the testing activities [16,52,53]. Therefore, role of environment management must be clearly defined to each team member of DevOps while performing testing activities.

C6 "unit testing" was reported as the significant capability of testing in DevOps process. Soni [54] and Danglot [55] highlighted that the DevOps testing experts must perform unit testing during software development process to reduce error rates. To avoid uncertainly during deployment phase of software in continuous environment (DevOps), it is essential to validate the individual unit of source code before processing [50,56].

C3 "destructive testing" was cited as the significant capability for DevOps testing process. To recover the service after failure, destructive testing is performed which measures the aspects of failure rates from all perspectives. The people who perform such testing are security testers, performance testers and developers [57,58]. According to Clokie [16] destructive testing is performed to measure the robustness of a software system and to identify the point of failure. Therefore, Clokie argued that the destructive testing could have positive impact on DevOps testing while planning testing strategies.

C20 "Testing backend integrations" was also the highest cited capability to improve DevOps testing performance. According to Perera et al. [59] to improve software quality in DevOps it is important to check server sides (databases) to organize data in table and records during continuous processing as data is coming from various sources. However, the proper management of backend integration will help experts to improve testing in DevOps.

Table 5
Direction of relationship among testing capabilities (SSIM matrix).

ij	C20	C19	C18	C17	C16	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1
C1	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	O	*
C2	V	V	O	A	V	V	V	V	V	V	V	V	A	O	X	V	V	A	*	*
C3	V	A	V	V	V	X	V	V	V	O	V	V	V	V	A	V	V	*	*	*
C4	V	V	V	A	V	A	O	V	V	V	A	O	A	V	V	V	*	*	*	*
C5	V	V	V	V	V	A	O	X	O	A	V	V	V	V	V	*	*	*	*	*
C6	V	A	A	O	A	V	V	V	A	A	A	O	A	A	*	*	*	*	*	*
C7	V	V	V	V	A	A	O	A	A	V	V	V	V	*	*	*	*	*	*	*
C8	V	V	V	V	V	V	V	V	V	V	V	V	*	*	*	*	*	*	*	*
C9	A	A	A	A	O	A	A	O	A	A	A	*	*	*	*	*	*	*	*	*
C10	V	V	V	V	V	V	O	V	A	X	*	*	*	*	*	*	*	*	*	*
C11	V	X	V	V	V	V	V	O	A	*	*	*	*	*	*	*	*	*	*	*
C12	O	V	V	V	V	V	V	V	*	*	*	*	*	*	*	*	*	*	*	*
C13	X	O	V	A	V	O	A	*	*	*	*	*	*	*	*	*	*	*	*	*
C14	V	A	V	A	O	A	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C15	V	A	V	V	V	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C16	V	O	A	A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C17	V	A	V	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C18	V	A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C19	O	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C20	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

C9 “containers as testing” was reported as the important capability to perform testing in automation units of DevOps. Maheshwari et al. [60] and Révész and Pataki [61] specified the importance of containers technology for testing during deployment process of software to make production simulated accurately. This testing technique of “isolation” in DevOps will help other components of software to work smoothly [48].

C4 “testing logging” was the other most significant capability for DevOps testing. According to Clokie [16] “A bug in development can be a good opportunity to test log files”. Although, such analysis of testing log files might not happens much beyond incidents, but it can reveal the hidden issues and user behaviors in the system. Using these test log files we can improve testing performance in DevOps software firms.

C12 “system testing” was cited as the significant capability for testing. Lwakatare et al. [50] suggested that system testing is essential to improve DevOps testing process, as this testing capability is used to evaluated the whole system against specified requirements. The experts must perform end to end testing to check the functionalities of the system completely.

According to Clokie [16] C8 “bug bash” is the important capability for testing in DevOps in which all teams (development, operational, security, testing, design, management etc.) put aside their daily task and participated in a dedicated testing session.

Furthermore, Révész and Pataki [61] stated that C14 “canary release” is the best capability for DevOps testing. Sometimes quick tests are required to verify everything is according to the required task, before performing other time-consuming tests. To perform this task new software is released to small group of people for verification and after their approval, software component is deployed properly [16].

4.2. ISM findings

ISM is applied to identify the contextual relationship between the testing capabilities for DevOps identified from literature (MLR). Using ISM we have developed a conceptual framework to improve DevOps testing process. The framework consist of levels, hierarchies and links among identified capabilities. This framework is simplistic and provides valuable insights in the context to improve DevOps process based on the significance of testing capabilities for DevOps. This technique has been applied in various other fields i.e. supply chain, knowledge management, medical, quality management, production and planning, education etc. [22,40,41].

4.2.1. Structure self-interaction matrix (SSIM)

For analyzing the relationship between variables (testing capabilities) for DevOps, we computed SSIM through opinion of experts. We invited decision makers (DM) from industry and academia who are well aware of DevOps testing process. The demographic detail is presented in Table 10. The sample size (i.e., 5 DM) may cause vagueness and limit our research to some extent. There are various studies, which have used small sample size for example, Kannan et al. [26] used five decision makers for the selection of reverse logistic providers using ISM and fuzzy TOPSIS approach. Liangliang et al. [42] used panel of nine experts to study vulnerability factors of urban rail transit system using AHP and ISM approach. Shahmeem [62] used five decision-makers to priorities the factors in a firm using agile manifesto. Wong and Li [63] evaluate their findings based on nine experts in the field of intelligent building systems. Rafi et al. [8] used five DevOps experts to rank the challenges hindered in DevOps adoption using Fuzzy TOPSIS approach.

We used four symbols to show the direction of relationship between variables (i and j) Table 5.
V: The variable i will help to achieve variable j
A: The variable j will be achieved by variable i
X: The variables i and j will help to achieve each other
O: The variables i and j are not related to each other
The Table 5 shows the SSIM matrix after gathering opinions from experts. The following examples explain the use of symbols V, A, X and O in SSIM.

1. The C1 “monitoring as testing” will help to improve testing C20 “backend integration” so; there relationship is shown by V (row of C1 and column of C20).
2. As per experts’ opinion, the C1 “monitoring as testing” and C2 “environment management for testing” are unrelated to each other so there relationship is represented by O in the cell at integration row of C1 and column of C2.
3. The C2 “environment management” is possible because of C17 “infrastructure testing”, hence A is assigned at the intersection cell of both variables (row of C2 and column of C17).
4. Similarly, C2 “environment management and C6 “testing logging” helps to achieve each other therefore X is assigned at the intersection cell of both variables (row of C2 and column of C6).

4.2.2. Reachability matrix

This matrix is formed by substituting the values of V,A,X and O in SSIM into binary form (0, 1). The protocols to assign a value as 0 and 1 are as follow:

Table 6
Reachability matrix.

i, *j*	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
C1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	0	1	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1
C3	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	1
C4	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	0	1	1	1
C5	0	0	0	0	1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	1
C6	0	1	1	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1
C7	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1
C8	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C9	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
C10	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1
C11	0	0	0	0	1	1	1	0	1	1	1	0	0	1	1	1	1	1	1	1
C12	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0
C13	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	1	0	1	0	1
C14	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	1	0	1
C15	0	0	1	1	1	0	1	0	1	0	0	1	0	1	1	1	1	1	0	1
C16	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	0	0	0	1
C17	0	1	0	1	0	0	0	0	1	0	0	1	1	1	0	1	1	1	0	1
C18	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	1	0	1
C19	0	0	1	0	0	1	0	0	1	0	1	1	0	1	1	0	1	0	1	0
C20	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1

Table 7
Final reachability matrix.

i, *j*	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	Dri	Rank
C1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	10
C2	0	1	*1	1	1	1	*1	*1	1	1	1	1	1	1	1	1	*1	*1	1	1	19	10
C3	0	1	1	1	1	*1	1	1	1	1	*1	1	1	1	1	1	1	1	*1	1	19	10
C4	0	*1	*1	1	1	1	1	*1	*1	*1	1	1	1	*1	*1	1	*1	1	1	1	19	10
C5	0	*1	*1	*1	1	1	1	1	1	1	*1	*1	1	*1	*1	1	1	1	1	1	19	10
C6	0	1	1	*1	*1	1	*1	0	*1	*1	0	*1	1	1	1	*1	*1	*1	0	1	16	8
C7	0	*1	*1	*1	*1	1	1	1	1	1	1	*1	*1	*1	*1	*1	1	1	1	1	19	10
C8	0	1	*1	1	*1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	10
C9	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	2	1
C10	0	*1	*1	1	*1	1	1	0	1	1	1	*1	1	*1	1	1	1	1	1	1	18	9
C11	0	*1	*1	*1	1	1	1	0	1	1	1	*1	*1	1	1	1	1	1	1	1	18	9
C12	0	*1	*1	*1	*1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	*1	18	9
C13	0	0	0	0	1	*1	1	0	*1	*1	0	1	1	*1	0	1	0	1	0	1	11	5
C14	0	0	0	0	0	*1	0	0	1	*1	0	1	1	1	0	*1	0	1	0	1	9	4
C15	0	*1	1	1	1	*1	1	0	1	*1	0	1	*1	1	1	1	1	1	0	1	16	8
C16	0	0	0	0	0	1	1	0	0	*1	0	1	0	*1	0	1	0	0	0	1	7	2
C17	0	1	0	1	0	*1	0	0	1	*1	0	1	1	1	0	1	1	1	0	1	12	6
C18	0	0	0	0	0	1	0	0	1	*1	0	1	0	*1	0	1	0	1	0	1	8	3
C19	0	0	1	0	0	1	0	0	1	0	1	1	0	1	1	0	1	0	1	0	9	4
C20	0	0	0	*1	0	*1	*1	0	*1	1	*1	*1	*1	1	*1	*1	*1	*1	*1	1	15	7
Dep	1	12	13	14	13	19	16	7	19	18	12	19	16	19	14	18	15	17	12	18		
Rank	1	3	4	5	4	10	7	2	10	9	3	10	7	10	5	9	6	8	3	9		

1. If the value of (i, j) is V in SSIM then the value in reachability matrix will become 1 and the (j, i) entry will become 0.
2. If the value of (i, j) is A in SSIM then the value in reachability matrix will become 0 and the (j, i) value will become 1.
3. If the value of (i, j) is O in SSIM then both values (i, j) and (j, i) in reachability matrix will become 0.
4. If the value of (i, j) is X in SSIM then both values (i, j) and (j, i) in reachability matrix will become 1.

Following these protocols, the reachability matrix is shown in Table 6.

The final reachability matrix is computed by checking the transitivity as explained in Section 3. The *1 value in Table 7 are included to incorporate transitivity to fill the gaps, if any, while developing SSIM.

The driving power, dependence power and ranks of all variables are also calculated in Table 7. The driving power is the total number of variables, which it helps in achieving that variable (including variable itself). The dependence power is the total number of variables, which may help in achieving that particular variable. These values will be used in Matrix Cross-Reference Multiplication Applied to a Classification (MICMAC) analysis [26] where the variables are categorized into four groups i.e., autonomous, dependent, independent and linkage variables.

4.2.3. *Level partitioning of reachability matrix*

We derived reachability set and antecedent sets from final reachability matrix. The reachability set depends upon variable itself and the other variables it may impact, whereas the antecedent set consists of itself and other variables, which impact it. All variables with their antecedent set, intersection set and reachability set are shown in Table 8. The intersection set is derived from antecedent and reachability set. The top level of ISM is achieved by the variables that have same reachability set and intersection set. These are the top most variables leading all other variables and will be removed from consideration in next iterations. The same procedure will be applied to all other variables until we found the level of each variable. In this case, the entire process of leveling the variables is reached in six iterations as shown in Table 8.

4.2.4. *Formation of conical matrix*

After level partitioning conical matrix is formed (Table 9) by clustering the capabilities at same level across the columns and rows of final reachability matrix. The mutual relationship of

Table 8
Level partitioning of final reachability matrix.

C#	Level partitions			
	Iteration one			
C#	Reachability set	Antecedent set	Intersection set	level
C1	1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1	1	
C2	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 17	2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 17	
C3	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 19	2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 19	
C4	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 17, 20	2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 17, 20	
C5	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15	2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15	
C6	2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17, 18, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	2, 3, 4, 5, 6, 7, 10, 12, 13, 14, 15, 16, 17, 18, 20	
C7	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 20	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 20	
C8	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 7, 8	2, 3, 4, 5, 6, 7, 8	
C9	7, 9	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20	7, 9	I
C10	2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20	2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20	
C11	2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 19, 20	2, 3, 4, 5, 6, 7, 10, 11, 12, 19, 20	
C12	2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	
C13	5, 6, 7, 9, 10, 12, 13, 14, 16, 18, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 17, 20	5, 6, 7, 10, 12, 13, 14, 20	
C14	6, 9, 10, 12, 13, 14, 16, 18, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	6, 10, 12, 13, 14, 16, 18, 20	
C15	2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17, 18, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 19, 20	2, 3, 4, 5, 6, 10, 11, 12, 15, 20	
C16	6, 7, 10, 12, 14, 16, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20	6, 7, 10, 12, 14, 16, 20	I
C17	2, 4, 6, 9, 10, 12, 13, 14, 16, 17, 18, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 17, 19, 20	2, 4, 6, 10, 12, 17, 20	
C18	6, 9, 10, 12, 14, 16, 18, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 17, 18, 20	6, 10, 12, 14, 18, 20	
C19	3, 6, 9, 11, 12, 14, 15, 17, 19	1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 19, 20	3, 11, 12, 19	
C20	4, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20	4, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	
Iteration two				
C1	1, 3, 4, 5, 8, 11, 13, 15, 17, 18, 19	1	1	
C2	2, 3, 4, 5, 8, 11, 13, 15, 17, 18, 19	2, 3, 4, 5, 8, 11, 15, 17	2, 3, 4, 5, 8, 11, 15, 17	
C3	2, 3, 4, 5, 8, 11, 13, 15, 17, 18, 19	1, 2, 3, 4, 5, 8, 11, 15, 19	2, 3, 4, 5, 8, 11, 15, 19	
C4	2, 3, 4, 5, 8, 11, 13, 15, 17, 18, 19	2, 3, 4, 5, 8, 11, 15, 17	2, 3, 4, 5, 8, 11, 15, 17	
C5	2, 3, 4, 5, 8, 11, 13, 15, 17, 18, 19	1, 2, 3, 4, 5, 8, 11, 13, 15	2, 3, 4, 5, 8, 11, 13, 15	
C8	2, 3, 4, 5, 8, 11, 13, 15, 17, 18, 19	1, 2, 3, 4, 5, 8	2, 3, 4, 5, 8	
C11	2, 3, 4, 5, 13, 15, 17, 18, 19	1, 2, 3, 4, 5, 8, 11, 19	2, 3, 4, 5, 19	
C13	5, 13, 18	1, 2, 3, 4, 5, 8, 11, 13, 15, 17	5, 13	
C15	2, 3, 4, 5, 13, 15, 17, 18	1, 2, 3, 4, 5, 8, 11, 15, 19	2, 3, 4, 5, 15	
C17	2, 4, 13, 17, 18	1, 2, 3, 4, 5, 8, 11, 15, 17, 19	2, 4, 17	
C18	18	1, 2, 3, 4, 5, 8, 11, 13, 15, 17, 18	18	II
C19	4, 11, 13, 15, 17, 18, 19	1, 2, 3, 4, 5, 8, 13, 15, 17, 18	4, 13, 15, 17, 18	

(*continued on next page*)

Table 8 (*continued*).

	Level partitions			
	Iteration one			
C#	Reachability set	Antecedent set	Intersection set	level
Iteration three				
C1	1, 3, 4, 5, 8, 11, 13, 15, 17, 19	1	1	
C2	2, 3, 4, 5, 8, 11, 13, 15, 17, 19	2, 3, 4, 5, 8, 11, 15, 17	2, 3, 4, 5, 8, 11, 15, 17	
C3	2, 3, 4, 5, 8, 11, 13, 15, 17, 19	1, 2, 3, 4, 5, 8, 11, 15, 19	2, 3, 4, 5, 8, 11, 15, 19	
C4	2, 3, 4, 5, 8, 11, 13, 15, 17, 19	2, 3, 4, 5, 8, 11, 15, 17	2, 3, 4, 5, 8, 11, 15, 17	
C5	2, 3, 4, 5, 8, 11, 13, 15, 17, 19	1, 2, 3, 4, 5, 8, 11, 13, 15	2, 3, 4, 5, 8, 11, 13, 15	
C8	2, 3, 4, 5, 8, 11, 13, 15, 17, 19	1, 2, 3, 4, 5, 8	2, 3, 4, 5, 8	
C11	2, 3, 4, 5, 13, 15, 17, 19	1, 2, 3, 4, 5, 8, 11, 19	2, 3, 4, 5, 19	
C13	5, 13	1, 2, 3, 4, 5, 8, 11, 13, 15, 17	5, 13	III
C15	2, 3, 4, 5, 13, 15, 17	1, 2, 3, 4, 5, 8, 11, 15, 19	2, 3, 4, 5, 15	
C17	2, 4, 13, 17	1, 2, 3, 4, 5, 8, 11, 15, 17, 19	2, 4, 17	
C19	4, 11, 13, 15, 17, 19	1, 2, 3, 4, 5, 8, 13, 15, 17	4, 13, 15, 17	
Iteration four				
C1	1, 3, 4, 8, 11, 15, 17, 19	1	1	
C2	2, 3, 4, 8, 11, 15, 17, 19	2, 3, 4, 8, 11, 15, 17	2, 3, 4, 8, 11, 15, 17	
C3	2, 3, 4, 8, 11, 15, 17, 19	1, 2, 3, 4, 8, 11, 15, 19	2, 3, 4, 8, 11, 15, 19	
C4	2, 3, 4, 8, 11, 15, 17, 19	2, 3, 4, 8, 11, 15, 17	2, 3, 4, 8, 11, 15, 17	
C8	2, 3, 4, 8, 11, 15, 17, 19	1, 2, 3, 4, 8	2, 3, 4, 8	
C11	2, 3, 4, 15, 17, 19	1, 2, 3, 4, 8, 11, 19	2, 3, 4, 19	
C15	2, 3, 4, 15, 17	1, 2, 3, 4, 8, 11, 15, 19	2, 3, 4, 15	
C17	2, 4, 17	1, 3, 5, 8, 11, 15, 17, 19	2, 4, 17	IV
C19	4, 11, 15, 17, 19	1, 2, 3, 4, 8, 15, 17	4, 15, 17	
Iteration five				
C1	1, 3, 8, 11, 15, 19	1	1	
C3	3, 8, 11, 15, 19	1, 3, 8, 11, 15, 19	3, 8, 11, 15, 19	V
C8	3, 8, 11, 15, 19	1, 3, 8	3, 4,	
C11	3, 15, 19	1, 3, 8, 11, 19	3, 19	
C15	3, 15	1, 3, 8, 11, 15, 19	3, 15	
C19	11, 15, 19	1, 3, 8, 15	15	
Iteration six				
C1	1	1	1	VI

Table 9
Conical matrix.

i.j	C6	C7	C9	C10	C12	C14	C16	C20	C18	C5	C13	C2	C4	C17	C3	C8	C11	C15	C19	C1	Dri	Rank
C6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	16	8
C7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	19	10
C9	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1
C10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	18	9
C12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	18	9
C14	1	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	9	4
C16	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	7	2
C20	1	1	1	1	1	1	1	1	1	0	1	0	1	1	0	0	1	1	1	0	15	7
C18	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	8	3
C5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	19	10
C13	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	11	5
C2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	19	10
C4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	19	10
C17	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	12	6
C3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	19	10
C8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	19	10
C11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	18	9
C15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	16	8
C19	1	0	1	0	1	1	0	0	0	0	0	0	0	1	1	0	1	1	1	0	9	4
C1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	19	10
Dep	19	16	19	18	19	19	18	18	17	13	16	12	14	15	13	7	12	14	12	1		
Rank	10	7	10	9	10	10	9	9	8	4	7	2	5	6	4	2	3	5	3	1		

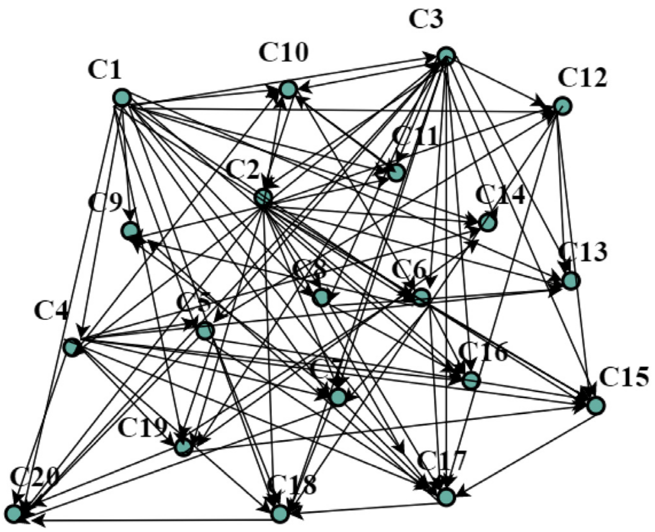

Fig. 7. Mutual relationship of testing capabilities.

capabilities is shown in Fig. 7. The driving and dependence power is represented as "Dri" and "Dep" in Table 9.

4.2.5. *Interpretation from ISM based framework*

The framework is derived from conical form of reachability-matrix. The primary diagraph obtained to show mutual relationship between capabilities (Fig. 7) including transitivity links after omitting the indirect links from reachability matrix, the diagraph is converted into ISM model as shown in Fig. 8. The Fig. 8 illustrates that C1 "monitoring as testing" (level-6) is the most significant capability while dealing with DevOps testing. All the other capabilities in above level (i.e. level 5–level 1) were derived from the capability, C1 "monitoring as testing". The C3 "destructive testing", C8 "bug bash", C11 "dark launching", C15 "smoke testing" and C19 "deployment pipelines" appearing at level 5 are also significant capabilities for DevOps testing with different driving and dependence power (for each capability) as explained briefly in MICMAC analysis. C6 "unit testing, C7 "A/B testing", C9 "containers as testing", C10 "beta testing", C12 "system testing", C14 "canary release", C16 "acceptance testing" and C20 "testing backend integrations" are the top level testing capabilities of DevOps on which effectiveness of DevOps testing performance depends. To show the relationship between i and j variables we used pointing arrows in ISM model.

4.2.6. *MICMAC analysis*

The aim of conducting MICMAC analysis is to determine the dependence power and driving power of variables (capabilities) [64]. The clusters in MICMAC analysis are classified into four sub-clusters.

1. The first cluster of autonomous variables have weak dependence and driving power. Mostly such variables are disconnected from the system because they have few links, which connect them to the system.
2. Second cluster of dependent variables consists of variables that have weak driving power and strong dependence power.
3. The third cluster of linkage variables, have strong driving and dependence power. Any action on these variables will effect the other variables linked to it.
4. Fourth cluster of independent variables have weak dependence power but strong driving power.

The variables with strong driving power are known as key variables, mostly fall in cluster of linkage and independence variables. The driving and dependence power of each variable is discussed in Table 9 and Fig. 9. It is clear from Fig. 9 that the capability; C1 "monitoring as testing" has driving power 19 and dependence power 1 which indicates that C1 is a key variable and is placed in cluster IV according to corresponding values of driving and dependence power. The variable C18 "feature toggles", is placed in cluster II depending upon the driving power (8) and dependence power (17). Similarly, the variable C9 "containers as testing" is placed in cluster II (driving power 2 and dependence power 19) which shows that C9 has strong dependence power and weak driving power. The same process is used to place all the other variables in graph (Fig. 9) to analyze their dependence and driving power.

4.3. *CAMS model*

CAMS is the fundamental model for the adoption of DevOps in software firms. It is a short form for (Culture, Automation, Measurement and Sharing) [27].

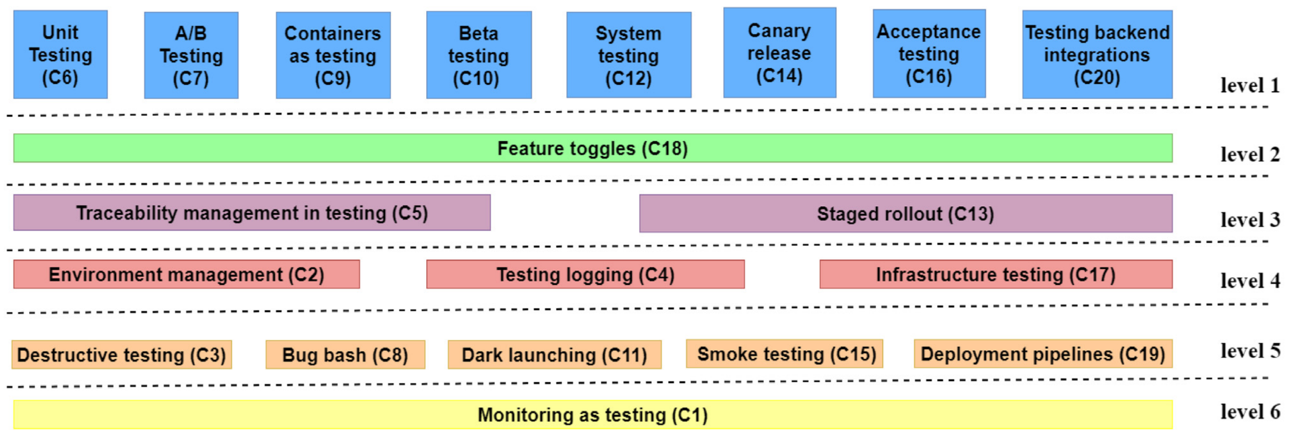

Fig. 8. Holistic view of testing capabilities (ISM model).

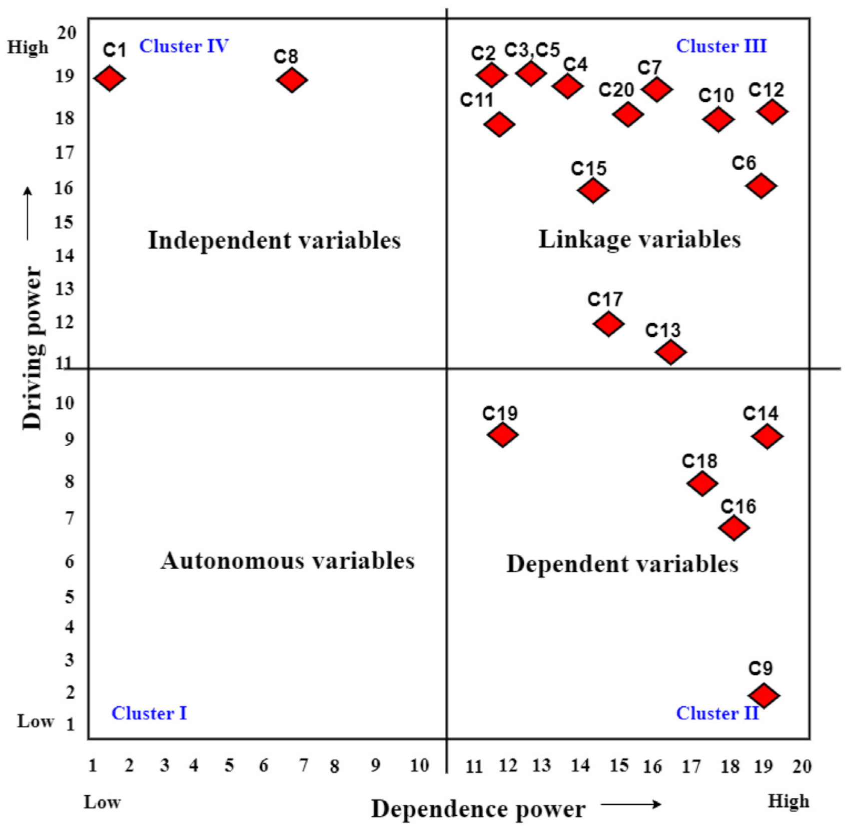

Fig. 9. MICMAC analysis.

Culture is a cornerstone of DevOps, by cooperating on tasks with all teams (in different department) and creating an environment of helping each other to complete task on time.
Automation is a continuous push code from development into production trying to make DevOps more reliable by saving cost, time, and effort.
Measurement: is about monitoring, collecting, analyzing, and assisting the team to work together on a common goal.
Sharing: is about sharing the tools, discovery, and ideas with others for making DevOps successful. Sharing is the heart of DevOps while dealing with collaboration.

4.4. Application results of Fuzzy TOPSIS

The ISM technique is applied to evaluate the inter-relationship among the various testing capabilities for DevOps. It also determined the levels and directions of capabilities depending upon their dependence and driving power. However, the findings of ISM technique are limited to identify the major drivers among these capabilities. Therefore, to weight the significance of each capability in fuzzy environment (for order preference by similarity to ideal solution TOPSIS) we have applied the Fuzzy TOPSIS approach. It is assumed that the derivation of relative weight of

Table 10
Demographic detail of DMs.

DM	Designation	City	Organization services	DevOps experience	Total experience
DM1	Testing analyst	China	Cloud manufacturing	8	15
DM2	Operational manager	Germany	Software development	6	11
DM3	System operator	US	Technology	5	8
DM4	Security assistant	Korea	IT services	5	7
DM5	Project manager	Saudi Arabia	Software development	3	5

Table 11
Combined decision matrix.

Weights	0.5	1.4	2.5	1.2	2.5	3.5	0.5	1.7	3	0.5	1.3	3
	Culture			Automation			Measurement			Sharing		
	Combined decision matrix											
C1	1.5	2.4	3.5	1.5	2.5	3.5	1.5	2.5	3	1.5	1	3.5
C2	0.5	2.1	3	1.5	2.5	3.5	0.5	1.3	1.5	0.5	1.6	2.5
C3	1	1.2	2.5	0.5	2.3	3.5	0.5	1.4	3	1	1.3	2.5
C4	1.5	2.1	3	1.5	2.5	3	1.3	3	3.5	1	2.4	3.5
C5	0.5	1.2	2	1.5	2.3	3	0.5	2	3	1.5	1	1.5
C6	0.5	1.2	2.5	1.5	2.4	3.5	0.5	1.7	3	0.5	1.2	2.5
C7	1.5	2.3	3	1.5	1.7	3	1	2	3	0.5	1.7	3
C8	1.5	1.6	2.5	1.5	2.3	3.5	0.5	1.3	2	0.1	1	1.5
C9	0.5	1.2	2.5	0.5	2.1	3.5	0.5	1.2	2.5	0.5	1.3	2
C10	1	1.2	2.5	0.5	2.1	3	0.5	1.3	2	0.1	1	1.5
C11	1	2	3	0.5	1.5	3	1	1.7	3	1	1.3	2
C12	0.5	1.4	2.5	1.5	2.1	3.5	0.5	1	1.5	0.5	1	1.5
C13	1	1.2	2	1.5	2.3	3	0.5	1	1.5	0.5	1	2
C14	0.5	1.4	2.5	1	2.4	3.5	0.5	1.3	2	0.1	1	1.5
C15	1	1.2	2	0.5	1.4	2.5	0.5	1.2	2	1	1.5	2.5
C16	1	1.2	2	0.5	1.4	2.5	0.5	1.3	2	0.1	1.2	2
C17	0.1	1.8	2	1.5	2.3	3	0.5	1.4	3.5	1.5	1.4	2
C18	0.5	1	1.5	0.5	1	1.5	0.5	1.2	2	0.5	1	1.5
C19	1	1.2	2	0.5	1.4	2.5	0.5	1.2	2	1	1	1.5
C20	0.5	1.4	2	1.5	2.3	3	0.5	1.4	2	1	1.2	3

each testing capability for DevOps will provide guidelines to the software experts in setting the priorities and will recommend key focus areas.

The Fuzzy TOPSIS is a multi-criteria group decision-making approach therefore, we formed a committee of five decision makers the demographic detail of decision-makers (DM) is given in Table 10. The decision makers were the same who were involved for ISM approach; we send them all details to the experts to collect their opinions about testing capabilities for DevOps process.

The questionnaire was used to gather responses from practitioners. The assessment of questionnaire is done by sending the questionnaire to three external experts (one academic from King Saud University, Saudi Arabia and two from industry IT.Soft company, Italy). They suggested some changes in questionnaire such as, "add comments section, the table of triangular fuzzy scale should be added in questionnaire etc". We updated our manuscript according to their suggestion and revised questionnaire after approval from experts is given in Appendix C.

The steps performed after gathering data from DMs are as follows:

Step1&2: The DMs used linguistic terms to assess the importance of testing capability in DevOps platform. These linguistic values were converted into numbers by using triangular fuzzy number scale as shown in Table 5.

Step 3: We computed the combined decision matrix to evaluate the ratings of capabilities with respect to criteria using Eq. (1). There are in total 20 identified testing capabilities for DevOps and four criteria of CAMS model. The four criteria are "culture, automation, measurement and sharing" discussed in Section 4.3. The aggregate fuzzy rating of combined decision matrix is given in Table 11.

Step 4 We computed the normalized decision matrix based on cost and beneficial criteria by using Eqs. (2) and (3). The criteria "automation" is selected as cost criteria depending upon its attributes i.e., "cost, time, revenue, effort etc. The computed results are presented in Table 12.

Step 5: The normalized-weighted decision-matrix is determined by multiplying weights of each criteria with capabilities (alternatives). These criteria weights were also collected from five DMs during response gathering process. Using Eq. (4) the normalized-weighted decision-matrix is given in Table 13.

Step 6: In this step, we computed the FNIS and FPIS for each capability. The capability is better if value is near to FPIS. The results are calculated after using Eqs. (5) and (6) shown in Tables 14 and 15.

Step 7: The Euclidean distance from fuzzy positive and negative ideal solution for each capability is calculated by using Eqs. (7) and (8) as shown in Tables 14 and 15.

Step 8: The closeness coefficient is calculated by using Eq. (9). The results are shown in Table 16.

Step 9: The ranks are computed after calculating the CCi value for each capability (Table 16). The higher the CCi value, the higher will be the rank of that capability.

The graphical distribution of testing capabilities with respect to their significance in DevOps testing process are presented in Fig. 10. According to the ranking of testing capabilities (Fig. 10) C2 "monitoring as testing" with CCi value (0.81) is marked as the highest ranked capability while managing testing activities in DevOps. C6 "unit testing" with CCi value (0.72) is the second highest ranked testing capability, as unit testing reduces the error rates and remove uncertainty during deployment process [50]. Another highest ranked testing capability is C3 "destructive testing" with CCi value (0.71). According to Stahl et al. [57] C3 "destructive testing" helps in recovery after failure and its value cannot be denied. Hence, all the above marked testing capabilities are essential in DevOps testing process and would help software experts in improving testing process of DevOps.

Table 12
Normalized fuzzy matrix.

Weights	0.5	1.4	2.5	1.2	2.5	3.5	0.5	1.7	3	0.5	1.3	3
	Culture			Automation			Measurement			Sharing		
	Normalized fuzzy matrix											
C1	0.4	0.7	1.0	0.4	0.7	1.0	0.2	0.2	0.3	0.4	0.3	1.0
C2	0.1	0.6	0.9	0.4	0.7	1.0	0.3	0.4	1.0	0.1	0.5	0.7
C3	0.3	0.3	0.7	0.1	0.7	1.0	0.2	0.4	1.0	0.3	0.4	0.7
C4	0.4	0.6	0.9	0.4	0.7	0.9	0.1	0.2	0.4	0.3	0.7	1.0
C5	0.1	0.3	0.6	0.4	0.7	0.9	0.2	0.3	1.0	0.4	0.3	0.4
C6	0.1	0.3	0.7	0.4	0.7	1.0	0.2	0.3	1.0	0.1	0.3	0.7
C7	0.4	0.7	0.9	0.4	0.5	0.9	0.2	0.3	0.5	0.1	0.5	0.9
C8	0.4	0.5	0.7	0.4	0.7	1.0	0.3	0.4	1.0	0.0	0.3	0.4
C9	0.1	0.3	0.7	0.1	0.6	1.0	0.2	0.4	1.0	0.1	0.4	0.6
C10	0.3	0.3	0.7	0.1	0.6	0.9	0.3	0.4	1.0	0.0	0.3	0.4
C11	0.3	0.6	0.9	0.1	0.4	0.9	0.2	0.3	0.5	0.3	0.4	0.6
C12	0.1	0.4	0.7	0.4	0.6	1.0	0.3	0.5	1.0	0.1	0.3	0.4
C13	0.3	0.3	0.6	0.4	0.7	0.9	0.3	0.5	1.0	0.1	0.3	0.6
C14	0.1	0.4	0.7	0.3	0.7	1.0	0.3	0.4	1.0	0.0	0.3	0.4
C15	0.3	0.3	0.6	0.1	0.4	0.7	0.3	0.4	1.0	0.3	0.4	0.7
C16	0.3	0.3	0.6	0.1	0.4	0.7	0.3	0.4	1.0	0.0	0.3	0.6
C17	0.0	0.5	0.6	0.4	0.7	0.9	0.1	0.4	1.0	0.4	0.4	0.6
C18	0.1	0.3	0.4	0.1	0.3	0.4	0.3	0.4	1.0	0.1	0.3	0.4
C19	0.3	0.3	0.6	0.1	0.4	0.7	0.3	0.4	1.0	0.3	0.3	0.4
C20	0.1	0.4	0.6	0.4	0.7	0.9	0.3	0.4	1.0	0.3	0.3	0.9

Table 13
Weighted normalized matrix.

Weights	0.5	1.4	2.5	1.2	2.5	3.5	0.5	1.7	3	0.5	1.3	3
	Culture			Automation			Measurement			Sharing		
	Weighted normalized fuzzy matrix											
C1	0.2	1.0	2.5	0.5	1.8	3.5	0.1	0.3	1.0	0.2	0.4	3.0
C2	0.1	0.8	2.1	0.5	1.8	3.5	0.2	0.7	3.0	0.1	0.6	2.1
C3	0.1	0.5	1.8	0.2	1.6	3.5	0.1	0.6	3.0	0.1	0.5	2.1
C4	0.2	0.8	2.1	0.5	1.8	3.0	0.1	0.3	1.2	0.1	0.9	3.0
C5	0.1	0.5	1.4	0.5	1.6	3.0	0.1	0.4	3.0	0.2	0.4	1.3
C6	0.1	0.5	1.8	0.5	1.7	3.5	0.1	0.5	3.0	0.1	0.4	2.1
C7	0.2	0.9	2.1	0.5	1.2	3.0	0.1	0.4	1.5	0.1	0.6	2.6
C8	0.2	0.6	1.8	0.5	1.6	3.5	0.1	0.7	3.0	0.0	0.4	1.3
C9	0.1	0.5	1.8	0.2	1.5	3.5	0.1	0.7	3.0	0.1	0.5	1.7
C10	0.1	0.5	1.8	0.2	1.5	3.0	0.1	0.7	3.0	0.0	0.4	1.3
C11	0.1	0.8	2.1	0.2	1.1	3.0	0.1	0.5	1.5	0.1	0.5	1.7
C12	0.1	0.6	1.8	0.5	1.5	3.5	0.2	0.9	3.0	0.1	0.4	1.3
C13	0.1	0.5	1.4	0.5	1.6	3.0	0.2	0.9	3.0	0.1	0.4	1.7
C14	0.1	0.6	1.8	0.3	1.7	3.5	0.1	0.7	3.0	0.0	0.4	1.3
C15	0.1	0.5	1.4	0.2	1.0	2.5	0.1	0.7	3.0	0.1	0.6	2.1
C16	0.1	0.5	1.4	0.2	1.0	2.5	0.1	0.7	3.0	0.0	0.4	1.7
C17	0.0	0.7	1.4	0.5	1.6	3.0	0.1	0.6	3.0	0.2	0.5	1.7
C18	0.1	0.4	1.1	0.2	0.7	1.5	0.1	0.7	3.0	0.1	0.4	1.3
C19	0.1	0.5	1.4	0.2	1.0	2.5	0.1	0.7	3.0	0.1	0.4	1.3
C20	0.1	0.6	1.4	0.5	1.6	3.0	0.1	0.6	3.0	0.1	0.4	2.6
A+	0.2	1.0	2.5	0.5	1.8	3.5	0.2	0.9	3.0	0.2	0.9	3.0
A−	0.0	0.4	1.1	0.2	0.7	1.5	0.1	0.3	1.0	0.0	0.4	1.3

5. Summary and discussion

The aim of this study is to identify the testing capabilities for better testing performance in DevOps. The findings of this study will provide body of knowledge to both industrial experts and researchers that can help them to successfully execute and manage testing strategies in DevOps paradigm. To reach the objective of this study, firstly, we have applied the multivocal literature review to explore the testing capabilities reported by grey (white papers, blogs, expert reports etc.) and formal literature (mainstream research articles and books). By performing multivocal literature review, a total of 20 DevOps testing capabilities are identified that are related to the core four principles of DevOps i.e., CAMS (culture, automation, measurement and sharing). The identified capabilities present the certain key areas of testing that need to improve for effective performance of testing in DevOps process. Furthermore, the questionnaire survey study was conducted with the aim to get the insight of industry experts regarding the findings of multivocal literature study i.e., DevOps testing capabilities. The experts participated in questionnaire survey study marked as all the identified testing capabilities are important for DevOps paradigm. In next steps, we applied the ISM approach to analyze the interrelationship between the DevOps testing capabilities. At final step, we used the fuzzy TOPSIS approach to prioritize the investigated testing capabilities with respect to their significance for DevOps principles (CAMS). The results of this study present the dependence and driving relationship between the testing capabilities, and their priorities with respect to CAMS, which will assist the software experts to establish new strategies of testing-based on the priorities of testing-capabilities in DevOps. The summary of findings is discussed below:

5.1. Identification of testing capabilities

To make testing in DevOps more transparent and effective, we identified the testing capabilities from literature using MLR

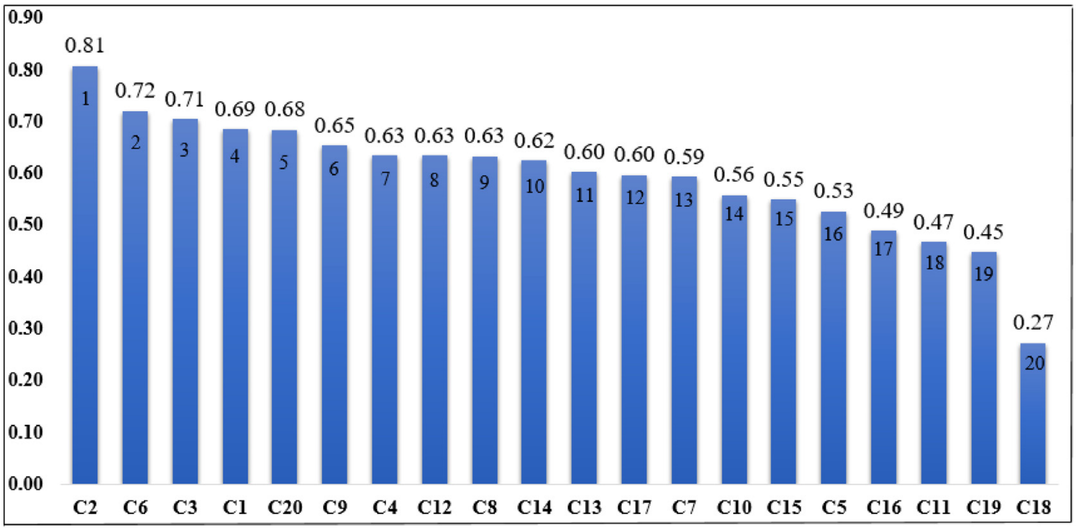

Fig. 10. Graphical representation of testing capabilities.

Table 14
Fuzzy positive ideal solution.

C#	Culture	Automation	Measurement	Sharing	Di+
C1	0.00	0.00	1.19	0.30	1.49
C2	0.23	0.00	0.11	0.53	0.88
C3	0.50	0.21	0.15	0.55	1.41
C4	0.22	0.29	1.12	0.04	1.66
C5	0.68	0.30	0.25	1.03	2.27
C6	0.50	0.04	0.21	0.56	1.32
C7	0.21	0.44	0.90	0.30	1.85
C8	0.45	0.08	0.12	1.04	1.69
C9	0.50	0.26	0.09	0.78	1.63
C10	0.50	0.39	0.12	1.04	2.04
C11	0.23	0.54	0.89	0.78	2.44
C12	0.48	0.16	0.00	1.04	1.68
C13	0.68	0.30	0.00	0.80	1.78
C14	0.48	0.11	0.12	1.04	1.74
C15	0.68	0.76	0.09	0.53	2.06
C16	0.68	0.76	0.12	0.79	2.35
C17	0.64	0.30	0.15	0.77	1.87
C18	0.89	1.32	0.09	1.04	3.34
C19	0.68	0.76	0.09	1.04	2.56
C20	0.67	0.30	0.14	0.36	1.47

Table 15
Fuzzy negative ideal solution.

	FNIS				
C#	Culture	Automation	Measurement	Sharing	Di-
C1	0.89	1.32	0.03	1.00	3.25
C2	0.67	1.32	1.18	0.51	3.68
C3	0.42	1.27	1.17	0.50	3.37
C4	0.68	1.08	0.09	1.04	2.89
C5	0.21	1.04	1.16	0.12	2.52
C6	0.42	1.31	1.16	0.50	3.38
C7	0.70	0.93	0.30	0.76	2.69
C8	0.45	1.29	1.17	0.00	2.91
C9	0.42	1.24	1.18	0.26	3.10
C10	0.42	0.98	1.17	0.00	2.57
C11	0.66	0.89	0.31	0.27	2.14
C12	0.42	1.26	1.20	0.03	2.91
C13	0.22	1.04	1.20	0.25	2.71
C14	0.42	1.29	1.17	0.00	2.89
C15	0.22	0.60	1.18	0.51	2.52
C16	0.22	0.60	1.17	0.25	2.25
C17	0.28	1.04	1.17	0.29	2.77
C18	0.03	0.00	1.18	0.03	1.25
C19	0.22	0.60	1.18	0.07	2.08
C20	0.23	1.04	1.17	0.75	3.18

Table 16
CCi values and ranks of testing capabilities.

C#	Capabilities	CCi	Ranks
C1	Monitoring as testing	0.685	4
C2	Environment management	0.808	1
C3	Destructive testing	0.705	3
C4	Testing logging	0.634	7
C5	Traceability management in testing	0.527	16
C6	Unit testing	0.720	2
C7	A/B testing	0.593	13
C8	Bug bash	0.633	9
C9	Containers as testing	0.654	6
C10	Beta testing	0.558	14
C11	Dark launching	0.467	18
C12	System testing	0.634	8
C13	Staged rollout	0.603	11
C14	Canary release	0.624	10
C15	Smoke testing	0.550	15
C16	Acceptance testing	0.489	17
C17	Infrastructure testing	0.597	12
C18	Feature toggles	0.272	20
C19	Deployment pipelines	0.448	19
C20	Testing backend integrations	0.684	5

(including formal and grey literature). Applying the step-by-step protocols of MLR, 15 pieces of grey literature and 24 articles from formal literature were selected. Considering the total selected 39 studies, the data extraction process was performed and a total of 20 testing capabilities were identified. The detail MLR findings is explained in Section 4.1and the list of identified DevOps testing capabilities is presented in Table 4.

5.2. Interaction between the testing capabilities

To measure the relationship between the testing capabilities, we have applied ISM approach to present holistic view of capabilities. This relationship will help the experts to measure the relative importance and the inter-relationships between the capabilities while developing DevOps testing strategies. The results show that considering the driving power, C1 "monitoring as testing" is the most important testing capability for the better performance measures in DevOps (Fig. 8, Section 4.2.5). It is less dependent on other testing capabilities and has strong driving power. Fig. 8 reveals that C1 "monitoring as testing" capability forms the base of DevOps testing process for better performance measures. C6 "unit testing", C7 "A/B testing", C9 "containers as testing", C10 "beta testing", C12 "system testing", C14 "canary release", C16 "acceptance testing" and C20 "testing backend integrations" are the top-level testing capabilities for the improvement of DevOps implication and performance. Thus, ISM model proposed in this study will provide a holistic view of the testing capabilities, that shows the interrelationship between the capabilities; and the understanding capabilities interrelationship will assist the practitioners to develop the better testing strategies for DevOps process.

5.3. Prioritization of testing capabilities

In decision-making, the use of linguistic terms plays an important role when it is difficult to express performance values in crisp numeric form. To fix this concern, we have used fuzzy TOPSIS approach as a generalized technique to solve multi-criteria decision-making (MCDM) problems under fuzzy environment. The values of closeness coefficient (CCi) presented in Table 16, Section 4.4, was used to determine the significance ranking of all testing capabilities (Fig. 10, Section 4.4). This ranking will assist the practitioners to update and revise their testing strategies in software firms based on the significant of testing capabilities in DevOps, to make performance loop successful. This study provides more objective information related to the adoption of better testing capabilities while developing DevOps testing strategies.

According to Fig. 10, C2 "the environment management" with CCi value (0.81) is the top ranked capability and it needs special focus of practitioners for the improvement of testing performance in DevOps. The environment management plays an important role for the maximum utilization of available resources. By taking steps to manage the DevOps testing environment, will reduce waste and helps in reuse and recycling of various resources [65]. C6 "unit testing" with CCi value (0.72) is the second most important testing capability for DevOps, as it helps to test small units in a program to avoid uncertainty during processing. The purpose of performing unit testing is to validate each component of software during development phase [55]. Another important capability in DevOps testing is C3 "destructive testing" having CCi value (0.71), it will help DevOps experts to check out test cases (which are carried to the specimen failure). The reason of conducting destructive testing is to reduce failure rates and cost [16]. The other high ranked capabilities to improve the DevOps process are: C1 "monitoring as testing" (0.69), C20 "testing backend integration" (0.68) and C9 "containers as testing" (0.65). This ranking of testing capabilities will help DevOps experts to upgrade their testing strategies in software firms for better performance, based on the significance of capabilities with respect to DevOps process.

6. Study implications

The study point-out an important aspect of DevOps paradigm that is DevOps testing process. The results and analysis of this study provides a significant knowledge for the DevOps experts of software industry, by exploring the testing capabilities for DevOps process. The identified capabilities plays an important role in the improvement of DevOps testing process. In this study, the ISM model of testing capabilities was proposed to give holistic view of all capabilities and to show inter-relationship between them. This conceptual model will assist the industrial experts with better understanding to revise their testing strategies based on the significance of testing capabilities in DevOps process. Besides, study findings provides a state-of-the-art review in the areas of DevOps testing process, which is a valuable contribution for academic research. Moreover, the prioritization of testing capabilities provides comprehensive analysis to the software experts, by highlighting the relative importance of testing capabilities with respect to DevOps. In summary, this study provides a detail overview of testing capabilities with the perspective to enhance testing performance while adopting DevOps culture.

7. Study limitation

Towards the generalization of study findings, some threats need to be addressed. For example, the researchers bias in the data extraction process. To address this threat, we have performed the interrater reliability test, and the results shows that the extracted data unbiased and consistent.

Another potential threat towards the validity of study findings is the search string was executed on limited digital repositories and this may lead to miss some related literature. Based on the studies conducted in the context of software engineering, this omission is not systematic [29,30].

One possible threat to validity is about sample size of decision makers (n = 5) which may cause vagueness. There are various studies, which have used small sample size [49,62,63] therefore, based on the existing studies small sample size is acceptable to perform analysis.

Moreover, the prioritization of testing capabilities based on MCDM approach is considered hasty and effect the results. To address this threat, we have used fuzzy TOPSIS approach which is the most suitable approach to resolve uncertainty in fuzzy group-decision problems.

8. Conclusion and future work

The rapid increase in the adoption of DevOps paradigm motivated us to explore and prioritize the testing capabilities for DevOps reported in the literature. Though, a total of 20 capabilities were identified using the multivocal literature review study. The testing capabilities were structured in the form of ISM model to analyze the inter-dependencies among the capabilities in driving towards effective testing strategies for DevOps process. In this effort the major strategic, operational and performance outcomes of testing capabilities were derived. Moreover, the prioritization of testing capabilities using fuzzy TOPSIS approach locates the key capabilities, which can play an important role in improving the DevOps testing process. It includes environment management, monitoring as testing, unit testing, destructive testing and testing backend integrations. The interaction with experts and finding from ISM and fuzzy TOPSIS clearly proved that there is an urgent need to involve testing capabilities in DevOps process (to increase testing performance). The results and analysis of this study will assists the industry practitioners to consider the key testing capabilities, for proper execution of DevOps process.

Therefore, in future, to increase the utility of model we will perform case study analysis by implementing the proposed model in practical situations. We will also perform further analysis on DevOps testing to explore more insights for effective testing performance in DevOps process.

CRediT authorship contribution statement

Saima Rafi: conceptualization, Data collection. **Muhammad Azeem Akbar:** conceptualization, Data collection, Methodology, supervision. **Wu Yu:** Methodology, supervision. **Ahmed Alsanad:** Data collection, Resources, Formal analysis. **Abdu Gumaei:** Data collection, Resources, Formal analysis. **Muhammad Umer Sarwar:** Review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors are grateful to the Deanship of Scientific Research, King Saud University, Saudi Arabia for funding through Vice Deanship of Scientific Research Chairs.

Appendix A

Formal literature (https://tinyurl.com/y5kguuqh).

Appendix B

Grey literature (https://tinyurl.com/yyrk8ysr).

Appendix C

Sample of questionnaire survey Fuzzy TOPSIS (https://tinyurl.com/y3o2nvtw).

Appendix D

Source file for computational reproducibility: https://figshare.com/s/b02b0c03a6112afc2615.

References

[1] R. Feijter, R. Vliet, E. Jagroep, S. Overbeek, S. Brinkkemper, Towards the Adoption of DevOps in Software Product Organizations: A Maturity Model Approach, Technical Report Series, 2017 (UU-CS-2017-009).

[2] N. Forsgren, D. Smith, J. Humble, J. Frazelle, 2019 Accelerate state of DevOps report, 2019.

[3] S. Rafi, W. Yu, M.A. Akbar, RMDevOps: A road map for improvement in DevOps activities in context of software organizations, in: Proceedings of the Evaluation and Assessment in Software Engineering, 2020 pp. 413–418.

[4] M. Virmani, Understanding DevOps & bridging the gap from continuous integration to continuous delivery, in: Fifth International Conference on the Innovative Computing Technology (INTECH 2015), IEEE, 2015, pp. 78–82.

[5] F. Erich, DevOps is simply interaction between development and operations, in: International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, Springer, Cham, 2018, pp. 89–99.

[6] 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.

[7] J. Smeds, K. Nybom, I. Porres, DevOps: A definition and perceived adoption impediments, in: International Conference on Agile Software Development, Springer, Cham, 2015, pp. 166–177.

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

[9] P. Woodall, M. Oberhofer, A. Borek, A classification of data quality assessment and improvement methods, Int. J. Inf. Qual. 16 3 (4) (2014) 298–321.

[10] A.A.U. Rahman, L. Williams, Software security in DevOps: Synthesizing practitioners' perceptions and practices, in: 2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery, CSED, IEEE, 2016, pp. 70–76.

[11] M.G. Jaatun, Software security activities that support incident management in secure DevOps, in: Proceedings of the 13th International Conference on Availability, Reliability and Security, 2018, pp. 1–6.

[12] S. Rafi, W. Yu, M.A. Akbar, Towards a hypothetical framework to secure DevOps adoption: Grounded theory approach, in: Proceedings of the Evaluation and Assessment in Software Engineering, 2020, pp. 457–462.

[13] A. Capizzi, S. Distefano, M. Mazzara, From DevOps to DevDataOps: Data management in DevOps processes, 2019, arXiv preprint arXiv:1910.03066.

[14] M. Mazkatil, A. Koziolek, Continuous integration and performance model, in: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. ser. ICPE '18. NewYork, NY, USA:ACM, 2018, pp. 153–158. [Online]. Available: http://doi.acm.org/10.1145/3185768.3186285.

[15] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Pearson Education, 2010.

[16] K. Clokie, A practical guide to testing in DevOps, 2017.

[17] P. Zimmerer, Strategy for continuous testing in iDevOps, in: Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, 2018, pp. 532–533.

[18] A. Bertolino, G.D. Angelis, A. Guerriero, B. Miranda, R. Pietrantuono, S. Russo, DevOpRET: Continuous reliability testing in DevOps, J. Softw. Evol. Process (2020) e2298.

[19] F. Faber, Testing in DevOps, in: The Future of Software Quality Assurance, library.oapen.org, Springer, Cham, 2020, pp. 27–38.

[20] R. Pietrantuono, S. Russo, On adaptive sampling-based testing for software reliability assessment, in: 2016 IEEE 27th International Symposium on Software Reliability Engineering, ISSRE, IEEE, 2016, pp. 1–11.

[21] D.S. Cruzes, K. Melsnes, S. Marczak, Testing in a DevOps era: Perceptions of testers in Norwegian organisations, in: International Conference on Computational Science and Its Applications, Springer, Cham, 2019, pp. 442–455.

[22] G. Kannan, A.N. Haq, P. Sasikumar, S. Arunachalam, Analysis and selection of green suppliers using interpretative structural modelling and analytic hierarchy process, Int. J. Manag. Decis. Mak. 9 (2) (2008) 163–182.

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

[24] P. Parthiban, H.A. Zubar, C.P. Garge, A multi criteria decision making approach for suppliers selection, Procedia Eng. 38 (2012) 2312–2328.

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

[26] 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.

[27] S. Guthrie, DevOps Principles- The CAMS Model. https://medium.com/@seanguthrie/devops-principles-the-cams-model-9687591ca37a.

[28] 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, http://dx.doi.org/10.1016/j.infsof.2018.09.006.

[29] M.A. Akbar, S. Mahmood, A. Alsanad, A.A.A. Alsanad, A. Gumaei, S.F. Qadri, A multivocal study to improve the implementation of global requirements change management process: A client-vendor prospective, J. Softw. Evol. Process (2020) e2252.

[30] 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.

[31] 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.

[32] L. Chen, M.Ali Babar, H. Zhang, Towards an Evidence-Based Understanding of Electronic Data Sources, 2010.

[33] 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.

[34] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, 2007.

[35] K.A. Hallgren, Computing inter-rater reliability for observational data: An overview and tutorial, Tutor. Quant. Methods Psychol. 8 (1) (2012) 23.

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

[37] J.W. Warfield, Developing interconnected matrices in structural modeling, IEEE Trans. Syst. Man Cybern. 4 (1) (1974) 51–81.

[38] A.P. Sage, Interpretive Structural Modeling: Methodology for Large-Scale Systems, McGraw-Hill, NewYork, 1977, pp. 91–164.

[39] G. Kannan, A.N. Haq, Analysis of interactions of criteria and sub-criteria for the selection of supplier in the built-in-order supply chain environment, Int. J. Prod. Res. 45 (17) (2007) 3831–3852.

[40] H.S. Tooranloo, M.A. Shahamabad, Designing the model of factors affecting in the implementation of social and environmental accounting with the ISM approach, Int. J. Ethics Syst. (2020).

[41] T. Sahoo, D.K. Banwet, K. Momaya, Developing a conceptual framework for strategic technology management using ISM and MICMAC methodology: A case of automotive industry in India, Glob. Bus. Rev. 12 (1) (2011) 117–143.

[42] L. Song, Q. Li, G.F. List, Y. Deng, P. Lu, Using an AHP-ISM based method to study the vulnerability factors of urban rail transit system, Sustainability 9 (6) (2017) 1065.

[43] L.A. Zadeh, G.J. Klir, B. Yuan, Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers. vol. 6, World Scientific, 1996.
[44] C. Erdin, H.E. Akbaş, A comparative analysis of fuzzy topsis and geographic information systems (GIS) for the location selection of shopping malls: A case study from turkey, Sustainability 11 (14) (2019) 3837.
[45] K. Yoon, C.L. Hwang, Manufacturing plant location analysis by multiple attribute decision making. Part I. Single-plant strategy, Int. J. Prod. Res. 23 (1985) 345–359.
[46] C.T. Chen, Extensions of the TOPSIS for group decision-making under fuzzy environment, Fuzzy Sets and Systems 114 (1) (2000) 1–9.
[47] R. Rostamzadeh, S. Sofian, Prioritizing effective 7Ms to improve production systems performance using fuzzy AHP and fuzzy TOPSIS (case study), Expert Syst. Appl. 38 (5) (2011) 5166–5177.
[48] C.-H. Cheng, Y. Lin, Evaluating the best main battle tank using fuzzy decision theory with linguistic criteria evaluation, European J. Oper. Res. 142 (2002) 174–186.
[49] S. Kim, K. Lee, J.K. Cho, C.O. Kim, Agent-based diffusion model for an automobile market with fuzzy TOPSIS-based product adoption process, Expert Syst. Appl. 38 (6) (2011) 7270–7276.
[50] L.E. Lwakatare, P. Kuvaja, M. Oivo, Dimensions of DevOps, in: International Conference on Agile Software Development, Springer, Cham, 2015, pp. 212–217.
[51] J. Verona, Practical DevOps, Packt Publishing Ltd., 2016.
[52] A.A. Ur Rahman, L. Williams, Security practices in DevOps, in: Proceedings of the Symposium and Bootcamp on the Science of Security, 2016, pp. 109–111.
[53] M.A. Akbar, S. Mahmood, M. Shafiq, A. Alsanad, A.A.A. Alsanad, A. Gumaei, Identification and prioritization of DevOps success factors using fuzzy-AHP approach, Soft Comput. (2020) 1–25.
[54] 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: 2015 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM, IEEE, 2015, pp. 85–89.
[55] B. Danglot, Automatic Unit Test Amplification For DevOps (Doctoral dissertation), 2019.
[56] A. Ravichandran, K. Taylor, P. Waterhouse, Devops finetuning, in: DevOps for Digital Leaders, Apress, Berkeley, CA, 2016, pp. 151–169.
[57] D. Stahl, T. Martensson, J. Bosch, Continuous practices and DevOps: Beyond the buzz, what does it all mean? in: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA, IEEE, 2017, pp. 440–448.
[58] S.N. Mullaguru, Changing scenario of testing paradigms using DevOps–A comparative study with classical models, Glob. J. Sci. Technol. (2015).
[59] P. Perera, R. Silva, I. Perera, Improve software quality through practicing DevOps, in: 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions, ICTer, IEEE, 2017, pp. 1–6.
[60] S. Maheshwari, S. Deochake, R. De, A. Grover, Comparative study of virtual machines and containers for devops developers, 2018, arXiv preprint arXiv:1808.08192.
[61] Á. Révész, N. Pataki, Continuous A/B testing in containers, in: Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis, 2019, pp. 11–14.
[62] M. Shameem, R.R. Kumar, C. Kumar, B. Chandra, A.A. Khan, Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process, J. Softw. Evol. Process 30 (11) (2018) e1979.
[63] J.K. Wong, H. Li, Application of the analytic hierarchy process (AHP) in multi-criteria analysis of the selection of intelligent building systems, Build. Environ. 43 (1) (2008) 108–125.
[64] A. Mandal, S.G. Deshmukh, Vendor selection using interpretive structural modeling (ISM), Int. J. Oper. Prod. Manage. 14 (6) (1994) 52–59.
[65] J.R. Jadhav, S.S. Mantha, S.B. Rane, Development of framework for sustainable lean implementation: An ISM approach, J. Ind. Eng. Int. 10 (3) (2014) 72.