DEVOPS-AN OPEN ACCESS JOURNAL
Open Journal

Review

Requirements Management in DevOps Environments: A Multivocal Mapping Study

Rogelio Hernández[1]; Begoña Moros[2]; Joaquín Nicolás[2*]

[1]*Faculty of Engineering, University Autonomous of Guerrero, Chilpancingo, Mexico*

[2]*Department of Informatics and Systems, Faculty of Computer Science, University of Murcia, Campus of Espinardo, 30100 Murcia, Spain*

[*]***Corresponding author***
Joaquín Nicolás
Department of Informatics and Systems, Faculty of Computer Science, University of Murcia, Campus of Espinardo, 30100 Murcia, Spain; E-mail: jnr@um.es

Article information
Received: May 8th, 2024; **Revised:** August 1st, 2024; **Accepted:** August 22nd, 2024; **Published:** September 10th, 2024

Cite this article
Hernández R, Moros B, Nicolás J. Requirements management in DevOps environments: A multivocal mapping study. 2024; 3(2).
doi: https://doi.org/10.70705/ppp.doaj.2024.v03.i02.pp64-82

ABSTRACT

Attention is currently being focused on DevOps, which aims to reduce software development time by means of close collaboration between the development and operations areas. However, little effort has been made to determine the role of requirements management in DevOps. The objective of this study is to help both researchers and practitioners by providing an overview of the best practices regarding requirements engineering in DevOps and identifying which areas still need to be investigated. A multivocal mapping study has, therefore, been carried out in order to study which methodologies, techniques and tools are used to support requirements management in DevOps environments. After applying the review protocol, 37 papers from formal literature and 14 references from grey literature were selected for analysis. The general conclusions obtained after analysing these papers were that, within DevOps, more attention should be paid to: (1) the reuse of require- ments in order to identify systems and software artefacts that can serve as a basis for the specification of new projects; (2) the communication of requirements between the different areas of an organisation and the stakeholders of a project; (3) the traceability of requirements in order to identify the relationship with other requirements, artefacts, tasks and processes; (4) non-functional requirements in order to identify the requirements of the operations area in the early phases of a project; and finally (5) specific requirements tools that should be seamlessly integrated into the DevOps toolchain. All these issues must be considered without ignoring the agile and continuous practices of development, operations and business teams. More effort must also be made to validate new methodologies in industry so as to assess and determine their strengths and weaknesses.

Keywords
Requirements management; Requirements engineering; Agile requirements engineering; DevOps; Multivocal mapping study.

INTRODUCTION

The term DevOps [1–7] emerged in Belgium during the 2009 data centre migration conference [5] with the objec- tive of emphasising agile management techniques in soft- ware production activities. The main goal of DevOps is for development (Dev) and operations (Ops) teams to work in close collaboration. It is a collaborative, multidisciplinary effort within an organisation with which to automate the ongoing planning and continuous delivery of new software versions, while ensuring their correctness and reliability [1–5].

With DevOps, the development department (Dev) is in charge of deploying the new features required in order to meet business requirements, while the operations depart- ment (Ops) is responsible for managing modifications to production and service levels [1, 4]. Figure 1 shows 6 of the most important characteristics of DevOps. As shown in the figure, agility [8–14] is an important characteristic of the methodologies used in DevOps environments because it supports the development capability. Furthermore, DevOps adds the ability to continuously monitor the performance of the software product developed to agile methodologies [4].

Some of the benefits of adopting DevOps include a reduction in the average time of the deployment cycle for soft- ware deliverables, along with a reduction in communication problems among developers, bureaucratic practices and team overload. The implementation of DevOps reduces human error resulting from automation and the explicit knowledge of operation-related tasks required for software development [2].

DevOps requires an organisation to introduce new work processes, tools, personnel, technological changes and inno- vation into its structure and processes [5, 14]. Figure 2 illus- trates the DevOps process, showing how continuity is an essential element.

Requirement's engineering (RE) is a discipline that serves to establish an interface between a system and its environ- ment: it focuses on collecting information (domain models) from the environment, and gathering, negotiating, specifying and validating the requirements needed for the system to be developed [15]. Although the importance of RE for the suc- cess of projects is, to the best of our knowledge, recognised, not much attention has been paid to it in DevOps environ- ments. However, there are proposals for agile requirements engineering and continuous requirements engineering that could fit into the framework of DevOps methodologies [2, 15]. Nevertheless, to the best of our knowledge, there is no should be traced back to requirements, as this is essential for
continuous improvement.

DevOps has evolved into two main new definitions denominated as BizDevOps and SecDevOps. BizDevOps [20] involves all the areas of a company. BizDevOps pre- sents a new approach that provides business departments with the active possibility of creating the final parts of the software. This means that, to a certain extent and within certain limits, the business department staff can become a kind of programmers. This new concept of BizDevOps will serve to expand horizons by connecting continuous RE with business needs. Furthermore, the DevSecOps (or SecDevOps) approach integrates security from the begin- ning and throughout the life cycle of an application in the fast and agile world of DevOps [21]. DevSecOps advocates shifting security to the left (by applying security measures throughout the development process), security by design and continuous security testing, by creating automated security controls in the DevOps workflow [22].

The objective of this research is to show the role that requirements play in DevOps practices. This is done through the use of a multivocal mapping study (MMS) [23]. An MMS differs from a classical systematic mapping study (SMS) [24] as regards the types of sources considered for the study. While an SMS focuses on academic, peer-reviewed resources such as journal or conference papers (hereafter, formal literature), an MMS includes grey literature (GL) such as books, white papers, annual reports or blogs. An MMS is useful for both researchers and practitioners, since it synthesises the state of the art and the state of the practice in a specific domain [23]. Given the fact that the topic under study in this paper appears to be more active in industry when compared to academia, ignoring GL could lead to the omission of the valuable point of view of practitioners.

This is not the first time that a multivocal study regarding DevOps has been conducted, but, to the best of our knowl- edge, it is the first MMS on RE in DevOps. Recently, Duarte- Amaro et al. [25] have a multivocal literature review (MLR) on DevOps practices and capabilities published, which resulted in an ordered taxonomy with the aim of reaching a consensus between researchers and practitioners. Moreover, Myrbakken and Colomo-Palacios [21] conducted an MLR in order to clarify the definition of DevSecOps and its characteristics. These authors also analysed the main benefits and challenges expected after the adoption of DevSecOps. This same topic was addressed by Mao et al. [26], but this time the authors focused on the impact of DevOps on software security, practitioners´ understanding of this issue and the practices used to support it. The other major evolution of DevOps, BizDevOps, has also been analysed by means of an MLR [27] in order to understand the meaning of the term and its features, together with the benefits and challenges of its adoption.

The main goal of this MMS is to review the evidence that
can be found in literature and other sources concerning the:
(1) methodologies; (2) techniques; and (3) tools that have been proposed in order to carry out requirements manage- ment activities in DevOps environments; together with (4) the methods used to validate these proposals, if any. The objective of this secondary study is to (1) categorise the results concerning the earlier goals and (2) motivate a dis- cussion highlighting current topics of requirements man- agement in DevOps that require further attention by both researchers and practitioners.

This paper is structured as follows. Section 2 describes the procedure employed to carry out this MMS. Section 3 shows the proposed categorisation of results, while Sect. 4 discusses some important issues related to those results. Sec- tion 5 then analyses threats to validity, and finally, Sect. 6 includes the conclusions of this secondary study.

2 Research method

This MMS was carried out by following the recommenda- tions regarding SMSs proposed by Petersen et al. [24, 28], supplemented by the recommendations of Garousi et al.
[23] in order to consider GL. This SMS is organised in four stages: (1) feasibility; (2) planning; (3) conducting the review; and (4) presentation of results, as shown in Fig. 3. The protocol used to conduct the MMS follows the standard structure of a protocol for SMSs, taking the guidelines pro- posed by Garousi et al. as variation points.

2.1 Feasibility

First, a feasibility study was carried out in order to determine whether the proposed MMS was of interest. The analysis was carried out by performing the following tasks:

- Determine initial searches to verify whether there are papers that deal with the central subject of this research.
- Identify keywords from the results of the initial searches in order to obtain an idea of how to define the search string.
- Define the initial search string.
- Apply the initial search string to the bibliographic databases in order to discover the number of studies.

Several secondary studies on agility and requirements management were found [8, 9, 12]–[14, 29, 30]. There are also reviews, such as that by Haindl et al. [31] which dis- cusses non-functional requirements (NFRs) in DevOps, but the focus of the present paper is more general and also cov- ers the processes and techniques that apply to requirements management.

Having discovered the interest in the research topic, it is necessary to motivate the need for an MMS rather than an SMS. According to Garousi et al. [23], this decision should be made systematically using a well-defined set of seven criteria. Only if at least one of the criteria is met, the MMS is justified. The more criteria that are met, the greater the need to conduct an MMS: (1) the subject is "complex" and not solvable by considering only formal literature; (2) the lack of volume in formal literature concerning RE in DevOps; (3) the im-

portance of contextual information regarding which requirements management issues work in DevOps; (4) the validation of academic proposals in industry; (5) the goal is to challenge assumptions or falsify results from practice using academic research or vice versa; (6) the usefulness of the synthesis of insights for both academia and practitioners; and (7) there is a large volume of practitioner sources indi- cating high practitioner interest in a topic. We believe that at least four out of the seven criteria are met in this research (2, 3, 4, 6), and conducting an MMS consequently appears to be appropriate for the topic under study.

The papers identified by means of the tentative searches, which explicitly refer to the terms DevOps and requirements, were used as the basis on which to develop and test the ini- tial search string, which is presented in Sect. 2.2.2.

2.2 Planning

In the planning stage, the search and collection protocol was proposed on the basis of the following tasks (Fig. 3): (1) establishment of research questions (RQs); (2) definition of a search strategy for the selection of primary studies; (3) definition of the inclusion and exclusion criteria; (4) data extraction strategy; and finally (5) definition of the quality criteria.

2.2.1 Establishment of the research question

In line with the objectives of this study, the RQs focused on categorising and structuring the studies related to DevOps and requirements. Four RQs were, therefore, identified for this purpose. These would describe the current knowledge on requirements management in DevOps in order to identify gaps that would provide specific areas for further research. The RQs were the following:

2.2.1.1 RQ1 What methodologies and frameworks are used for requirements management in DevOps? The goal of RE methodologies is to make the problem that is being addressed clear and complete, and to ensure that the solu- tion is correct, reasonable and effective [32]. RE activities (elicitation, analysis, documentation, validation and veri- fication) were traditionally carried out during the analysis phase of the software development life cycle. Nevertheless, the increase in iterative and agile methodologies led to a new way of working in order to adapt the practice to a more flexible and dynamic world. Some studies have reported

the challenges of agile RE [9, 29, 30, 33]. Given the fact that DevOps has evolved from agile methodologies, it is interesting to study whether agile RE methodologies have been adopted as is in DevOps or whether they have been improved in order to overcome the reported challenges. In addition to methodologies, frameworks are also included in this RQ, bearing in mind that a framework is a kind of loose, incomplete working structure that provides a process and allows the integration of practices and tools. A methodology is, in contrast, more complete and usually encompasses a set of principles, process models, techniques and tools.

2.2.1.2 RQ2 What techniques are used for requirements management in DevOps? According to Zowghi and Coulin [34], a technique is a way of doing something or a practical method applied to particular tasks. Different techniques are used for each of the core RE activities. Requirements docu- mentation can specifically be addressed by using conceptual models or natural language [35]. While traditional RE relies mainly on specifying textual requirements documents, the use of user stories as a means to specify requirements is widely accepted in agility [29], which is the dominant trend in DevOps [36]. However, Kasauli et al. [33] revealed that user stories, even when combined with test case specifica- tions, appear to be insufficient for requirements specifica- tions. The lack of documentation as an obstacle to the use of agile RE was also pointed out by Curcio et al. [29]. In this RQ, it is, therefore, worth studying the techniques adopted in DevOps in order to deal with requirements management modelling and documentation needs, since DevOps profes- sionals have pointed out that poorly defined functional and technical requirements are a challenge in organisations [37].

2.2.1.3 RQ3 What software tools are used to support requirements management in DevOps? A successful DevOps implementation generally relies on an integrated set of tools commonly known as a toolchain. A DevOps toolchain is a set of tools that automates the tasks of design- ing, building, testing, delivering, deploying, managing and controlling software applications throughout the system development lifecycle. It helps to achieve key DevOps principles, including continuous integration, continuous delivery, automation and collaboration. This makes product delivery faster and more efficient. Since RE is key aspect of the development process, software tools whose objective is to support requirements management techniques should be seamlessly integrated into the DevOps toolchain. The need for an adequate toolchain for agile RE [33] and the lack of specialised, automated tools [29] hamper the adoption of agile RE. The objective of this RQ is, therefore, to ascertain which requirement management tools can be integrated into the DevOps toolchain in order to contribute to automation and continuous integration.

2.2.1.4 RQ4 What methods have been used for the valida- tion of the methodologies, techniques and tools identified in RQs 1–3? According to Wohlin et al. [38], papers that evaluate their proposals by using controlled experiments, case studies or surveys can be considered to have been empirically evaluated. It is important to record whether the methodologies, techniques or tools identified in RQs1-3 have been not only proposed but also validated by means of some sort of empirical method, which will make the conclusions of those proposals sounder. As occurred in the work of Ouhbi et al. [39], the studies in this MMS have been classified into the following categories: (1) case study: an empirical inquiry that investigates a phenomenon within its real-life context. This term may refer to single or mul- tiple case studies; (2) survey: a method with which to col- lect quantitative information concerning experiences with RE and DevOps, e.g. by means of questionnaires; and (3) experiment: an empirical method applied under controlled conditions in order to observe their effects in DevOps con- texts. After analysing the studies selected for the MMS, three more categories which cannot be considered empirical were added: (4) proof of concept, which serves to verify that the solution is feasible from a technical point of view; (5) experience report, to describe an experience that has been conducted in such a way that it cannot be seen as properly empirically evaluated; and finally (6) not specified.

2.2.2 Defining search strategies for primary studies

The PICO (Population, Intervention, Comparison, Out- comes) criteria [40] were used to construct the search string, specifically Population (i.e. the application area) and Inter- vention (that is, the software methodology, tool, technology or procedure that addresses a specific issue). Comparison and Outcomes have not been specified in order to open up the search string, thus searching for as many studies as possible.

Each PICO criterion was linked using the Boolean oper- ator AND, whereas synonyms or alternative spelling in each criterion were linked using the Boolean operator OR. Table 1 shows the parts of the final search string, together with the rationale behind their in- clusion.

Information was sought by employing the scientific data- bases of (1) ACM Digital Library; (2) IEEE Xplore; (3) ScienceDirect; (4) Springer; and (5) Wiley online library to search for peer-reviewed studies (formal literature), together with Google Scholar. Further- more, the Google search engine was used to search for GL, togeth- er with RE Magazine and DevOps.com. The scientific databases consulted ensure the quality of their articles because they follow a rigorous review and publication process. No time restrictions were imposed on the searches in order to obtain all the published studies and not omit any that could be relevant for the pre- sent research.

2.2.3 Definition of inclusion and exclusion criteria

The relevant papers regarding the MMS were identified dur- ing the selection process, and inclusion and exclusion cri- teria were, therefore, defined. Inclusion criteria encompass the features that candidate studies should have, while exclu- sion criteria specify the aspects that lead to the rejection of any of the candidate studies. The criteria for the selection of studies should ensure papers that correspond to primary studies, which add value to the answers to the RQs. The cri- teria for the evaluation of peer-reviewed papers published in formal literature (journals, conferences or workshops), and GL (books, book chapters, web post, reports, white papers), are described below. The exclusion criteria were the following:

EC1 The study is duplicated.
EC2 The study is not published in English.
EC3 The item corresponds to a formal study that is less than 4 pages long or is a tutorial/workshop/course sum- mary only.
EC4 The full text of the study is not available.
EC5 Title, keyword list and abstract of the study do not discuss any practice regarding requirements manage- ment in DevOps, or it is not a primary study.
EC6 The study extends another paper, in which case the most complete version in terms of the RQs was selected. EC7 The item corresponds to an advertisement for a vendor's tool.

The inclusion criteria were:

IC1 The study describes a methodology or framework with which to deal with requirements in DevOps.
IC2 The study describes a technique with which to sup- port requirements management in DevOps (e.g. elici- tation, analysis, specification, documentation, reuse, traceability, validation).
IC3 The study describes a requirements management tool to be integrated into DevOps toolchains.
IC4 The study is the most complete version of a set of papers by the same authors, considering the RQs.

The above criteria were inspired by the Kuhrmann et al.[41] exemplary inclusion and exclusion criteria, but were extended with EC6, EC7 and IC4. EC6 and IC4 were intro- duced in order to select the most complete report of a line of work, while, as proposed by Myrbakken and Colomo-Pala- cios [21], EC7 was included in order to avoid the inclusion of advertising information originating from vendors rather than practitioners' experiences.

2.2.4 Definition of data extraction strategy—classification scheme

According to the classification scheme suggested by San- tos et al. [42], the data extraction strategy is based on the exploration of certain parts of each study (abstract, introduction, methods and results). Only in those cases in which the information is not clear or not well structured is it necessary to carry out a full reading of the paper. The process shown in Fig. 4 was followed in this MMS. This process is based on the keywords of the study, which is a strategy by which to reduce the time needed to develop the classification scheme and to ensure that the scheme considers existing studies.

The data extraction strategy consists of categorising each RQ into more specific criteria and concepts. This, therefore, makes it possible to establish a procedure whereby more specific terms are iteratively created for each RQ so as to facilitate the extraction and categorisation of studies [43]. If there are questions about the clas- sification of a study, the scheme is refined in order to allow the addition, modification or removal of concepts. In this type of scenario, once the item has already been classified, it has to be evaluated once again in order to validate its classification.

The classification procedure is described in Fig. 5. According to the recommendation of Petersen et al. [28], it is necessary to record information from all stages. Men- deley, a software for bibliographic management, was, therefore, used to extract information and read the selected articles, while Google Sheets was used to document the extracted concepts by means of a matrix.

2.2.5 Quality criteria

The purpose of including a quality assessment (QA) process is to ensure that the selected papers constitute a set of sound references. This process could be used to reduce the effort in the data extraction phase by including only those primary studies that exceed a pre-defined threshold (that is, a mini- mum quality level) [44]. In our case, it was used in order to enhance our secondary study by validating the selected studies, and not to eliminate any of them. This was done in order to analyse as much literature as possible.

The quality of the selected papers was assessed by defin- ing a 3-point Likert scale questionnaire. Each criterion could result in one

of three possible answers: Yes = 1, Par- tially = 0.5 or No = 0. The maximum score that a selected study could achieve was, therefore, 7 (i.e. all the QAs were rated as 1), and the minimum was 0. Table 2 shows the ques- tions that were included in the questionnaire in order to eval- uate the quality of the studies. The questionnaire was created by the first author of this paper. It should be noted that Q.A.6 is biased in relation to the publication year of the study and that this criterion should, therefore, have been scored by tak- ing into account both publication year and number of cites. However, this combination was not considered in this study for the sake of simplicity, as the quality criteria analysis did not remove any studies in the final selection.

Following the guidelines proposed by Garousi et al. [23], several QA criteria were considered in formal literature and grey literature. The QA GL checklist for software engineer- ing provided by these authors was used as a basis, and the criteria taken into account in order to select resources from the GL were: (1) "Authority of the producer", that is, the reputation of the organisation that published the informa- tion or the experience of the author (e.g. DevOps Institute, Microsoft or Atlassian); and (2) "Methodology", i.e. the source clearly states a methodology regarding requirements management in DevOps. These QA criteria for GL were added to the inclusion criteria in the selection process.

2.3 Conducting the review

The searches for formal literature were carried out in June 2022 by the first author of this paper as part of his doc- toral thesis. Table 3 shows the number of papers obtained by applying the search string to each database. Filters were applied in order to delimit English language papers and arti- cles. The only adaptation required for the search string con- cerned the ScienceDirect database and consisted of remov- ing the wildcard "*" from the search string, as this notation was not supported.

The selection process is shown in Fig. 6, while Table 4 shows the categorisation of the figures corresponding to each database. A total of 10,402 studies had to be screened by reading their title, keywords and abstract, thus obtain- ing 214 potentially relevant studies. Given the large number of results obtained from Google Scholar, which yielded a total of 8,460 articles distributed on more than 100 pages, an effort bounded criterion [23] was applied as a stopping criterion by taking into account the first 100 search hits, and the search was continued further if the hits on the last page still revealed additional relevant search results.

The first author of this paper applied the exclusion criteria to the 214 potentially relevant studies by re-reading the title, abstract, keywords and eventually the introduction, obtaining a total of 82 candidate studies. This review included the elimination of 35 duplicates obtained from all databases (application of EC1). In line with the guidelines of Kuhr- mann et al. [41], the set of studies was evaluated in the order of the databases shown in Table 4 so as to manage the duplicate studies obtained from the different databases. For example, papers obtained from the ACM Digital Library were maintained, while the same papers were removed from the set of results obtained from the remaining data- bamatched the search string, onlyses, even though traces to all the databases were maintained on the spreadsheet used to collect the papers. Furthermore, the columns regarding non-English studies (EC2) and short papers (EC3) are not included in Table 4 since these criteria did not remove any potentially relevant study. After, 11 non-available studies were discarded (application of EC4), and EC5 was then applied in order to eliminate 86 additional studies based on a new reading of the abstract and a review of the introduc- tions to the papers, thus leading to a set of 82 available can- didate studies.

The list of candidate studies was then reviewed by all the authors of this paper so as to evaluate whether or not each candidate study should be selected (by applying inclusion criterion IC1-3). The studies were judged independently by all three authors, and disagreements were discussed until a consensus was reached. Finally, the three authors agreed on the most complete version of the candidate studies represent- ing a coherent line of work by the same authors (application of EC6 for exclusion and IC4 for inclusion). This eventually led to the selection of 37 studies that were deemed relevant to answer the RQs (see Table 5, which also includes the evaluation of the QA). All the details of the selection process can be found at shorturl.at/bfTX3.

In addition to the searches in formal literature, the second author of this paper also searched for GL in order to seek the practitioners' point of view. The searches focused on two of the specialised websites regarding the domains involved in this secondary study, together with Google. The websites consulted were: (1) Requirements Engineering Magazine,1 the magazine for RE Professionals supported by the IREB (International Requirements Engineering Board); and (2) DevOps.com, which is considered the largest collection of original content related to DevOps on the Net, as claimed on the website. In these two cases, the search string was simplified, as given in Table 6. Please bear in mind that the number of results in Google fluctuates considerably on a day-to–day basis.

Given the huge number of results obtained by Google, an effort bounded criterion was again applied as a stopping criterion. When no new results were obtained on one page, at least five more pages containing the same result were reviewed before stopping. In the case of the DevOps.com search engine, it was necessary to trust in the classifica- tion of results that this search engine provides, and only the results it considered relevant were reviewed. Although 4,860 results matched the search string, only 10 are shown for the "Relevance" criterion of this search engine, and 4 of them were considered to be related to the RQs. Table 6 shows the number of potentially relevant studies that were selected by reading the title and metadata provided in the result sets. The potentially relevant studies were then read in their entirety in order to apply the inclusion and exclusion criteria, which resulted in 14 selected studies. The selection process was agreed on by all the authors of this paper. The details of the selection process can be seen at shorturl.at/guDL4.

2.3.1 Data extraction

The data extraction strategy presented in Sect. 2.2.4 made it possible to analyse the selected papers, while the informa- tion regarding the RQs was recorded by means of a data- sheet (shorturl.at/clKX7). This information was iteratively processed by all the authors of this paper until a consensus on a stable set of categories was reached. Table 7 shows the definition of the categories for each RQ that were used to classify the selected studies and guide the presentation of

the results. The table also indicates the number of studies found per category, along with their references.

Some studies provide insights into more than one RQ, thus showing that the categories are disjoint among neither RQs nor the categories of an RQ. For instance, one study can include a methodology proposal (RQ1) that encompasses new techniques (RQ2) together with tool support (RQ3), while another may present a process with which to build a model (which we denominated as Model Construction, RQ1) and an extension to a technique that can be used to build a model (which we denominated as Model Notation, RQ2). What is more, a study can be mapped onto the Agile and Lean Methodologies and the Model Construction categories (both in RQ1).

Figure 7 shows a word cloud based on the keywords of the selected studies from formal literature. The figure high- lights the large number of topics covered by the umbrella of this secondary study and how this knowledge area is still highly fragmented. At first sight, it appears that keywords such as Metrics, Reuse and Traceability deserve more atten- tion in literature. What is more, no keywords deal with Sus- tainability or Green software. Detailed results are presented in the following section.

3 Results

Following the recommendations of Webster and Watson [94], the next step was to conduct a detailed analysis of the selected papers by means of a descriptive synthesis of con- cepts related to the RQs. This involved answering each of the RQs posed in the research protocol. In order to categorise the results obtained for RQs 1–3, a set of categories was iteratively built on the basis of the authors of this paper's experience.

3.1 RQ1. What methodologies and frameworks are used for requirements management in DevOps?

As shown in Table 8, the Agile and Lean Methodologies category is the most prevalent, with 15 studies. The most frequently mentioned methodology in the field of require- ments management in DevOps is by far Scrum (6 studies) [45, 55, 65, 71, 73, 74], which is not really a requirements management methodology, but rather an agile project man- agement framework. In addition, some studies [73, 74] are limited to using a Kanban blackboard to manage require- ments. For example, in the work presented by Sinkala et al. [73], which concerns a case study related to the acceleration of software delivery in the context of requirements analysis, the authors mention that Kanban boards are used to put user stories in order so as to facilitate the prioritisation of require- ments. Another example is the work of Stray et al. [74] on large-scale dependency management in agility. This study states that teams have the possibility of choosing between Scrum and Kanban or a combination of both, in which agile practices such as daily stand-up meetings, Scrum of Scrum meetings and product demonstrations are carried out.

With regard to the Scrum-based approaches, Ali et al.

[45] show a hybrid systematic reuse process (a DevOps process based on agile methodologies together with a soft- ware development and management process) that allows the extension and incorporation of new processes into DevOps, such as a new development process, a maintenance process for a reuse repository and a project evolution process. The objective of these processes is to increase productivity, pro- ject quality and rapid delivery. In their work, Ali et al. [45] use Scrum events, such as Sprint planning meetings; preliminary functional requirements and software components are identified in this kind of meetings, and software engineers subsequently analyse candidate artefacts for reuse.

Also with regard to the Scrum-based approaches, Olsze- wska et al. [65] adopt Event-B in their proposal for the con- struction of formal models in highly critical complex sys- tems. Event-Bis, a formal method and a modelling language

for step-by-step analysis and modelling at the system level, is based on the action systems formalism. In this proposal, the authors adopt Scrum, since it is a framework that ena- bles agile and iterative development, it is based on frequent deliveries and short development cycles, and it has features that conceptually overlap with Event-B, such as iteration and refinement steps. Another reason for choosing Scrum was the clear definition of deadlines for iterations (sprint organi- sation) and the set of meetings that take place throughout the development process. Apart from Scrum, studies have been found showing different methodologies for requirements management in DevOps, as shown in Table 8. The remain- der of this section provides a brief description of the most characteristic methodology in each category.

In the security category, the proposal by Casola et al. [52] shows the use of the SSDE (security SLA-based security-by- design) methodology, which adopts security SLAs (service- level agreements) in order to model the security properties of a raw application in the cloud. This methodology makes it possible to carry out an assessment of the existing risks and an automated evaluation of the security level actually granted on the basis of the environment and deployment conditions. The SSDE methodology proposes a process that consists of three main phases, namely modelling, security assessment per component and security assessment per application.

In their work, Rios et al. [68] propose MUSA (multicloud secure applications) as a DevOps-oriented approach with which to support all the phases of a lifecycle of multicloud applications that is aware of security and privacy issues. MUSA encompasses activities starting from application privacy and security by design (including the creation of SLA) to deployment on selected cloud services and finally continuous assurance of SLA compliance in operation. MUSA enables multidisciplinary DevOps teams, bringing together application architects, developers, security archi- tects, business managers, service operators and system administrators in order to manage security and privacy risks in all phases of the multicloud application lifecycle.

In the Business category, Wiecher et al. [79] advocate a BizDevOps approach that supports a continuous update of requirements starting from business models. The project environment (Biz) is, therefore, supported as regards defin- ing scenarios that describe expectations regarding the pro- ject outcome and the possible impact. These high-level sce- narios are subsequently utilised in order to derive concrete and executable system scenarios (Dev). These executable system scenarios are used as early prototypes of the system in development that can be validated and aligned with the project environment (Ops).

The work carried out by Özcan-Top et al. [66] presents a hybrid assessment approach for medical device software development com-

panies consisting of the implementation of KATA, a general-purpose framework that includes a set of practices with which to achieve goals when the path to the goal is uncertain. The improvement to KATA provides an experimental approach with a 4-step pattern: (1) under- stand the direction or challenge; (2) understand the current

ALM (application lifecycle management) Adopting application lifecycle manage- ment condition; (3) establish the next target condition; and (4) iterate towards the target condition. In process improve- ment, the experimental approach employed to find the right practices and figure out how to implement them is very important, as it prevents companies from over-investing in a solution that will not work for them. The challenge of invest- ing in the right improvement practices at the right time can be achieved only with the help of an experimental approach. In the Model Construction category, Wahaballa et al. [78] suggest UDOM (Unified DevOps Model), a proposal that can help to standardise DevOps terminologies, concepts, patterns, culture and tools. UDOM includes three sub- models: (1) Application and Data Model (requirements are modelled at this stage); (2) Workflow Execution Model; and

(3) Infrastructure Model. This approach consequently helps to reduce the time required to build product deliverables, mitigate project risks, satisfy customer requirements and improve team productivity.

The Service Lambda approach by Hosono et al. [57] uses BizMap to collect data from (1) use cases and (2) application control diagrams. The data and dependencies of the dia- grams are stored in XML documents in a so-called Service portfolio (containing requirements, design, implementation and operation data repositories). A graphical model genera- tor that is based on SysML notation is used to transform the service portfolio data into Scalable Vector Graphics (SVG), generating the list of elements or levels of NFRs required in order to communicate with customers, analysts, designers and operators.

In their work, Casale et al. [51] describe the RADON (Rational Decomposition and Orchestration) project, which focuses on the development of an advanced DevOps frame- work with which to design, prototype, deploy, test, verify and evolve complex applications built on: (1) serverless FaaS (Function as a Service), defining events, triggers and actions (handling functions); (2) services of various granu- larities, typically microservices, implementing the business logic; and (3) data pipelines, i.e. specialised microservices (or serverless functions), resources and combined communi- cation mechanisms that manage the lifecycle of data (ingest, filtering, transformation, buffering, scheduling, analysis, transfer and storage). A new requirements specification technique is described in the context of RADON, which is described in RQ2 below.

The work of Furfaro et al. [54] shows a proposal for a methodology denominated as ResDevOps, whose objective is to have continuous control over Enterprise Information Systems (EIS) requirements. It uses the GOReM (Goal- Oriented Requirement Methodology) methodology to man- age the complexity of the requirements model. The GOReM phases are iterative, traceable and ready to receive changes in the EIS cycle, in which many partners with different cultures, languages and objectives cooperate in order to achieve innovative results.

With regard to the Requirements Specification category, Kirikova et al. [17] show their proposal called FREEDOM. This is a framework consisting of several functional units, namely F—future representation, R—reality representation, E1—requirements engineering, E2—compliance engineer- ing, D—design and implementation, O—operations and M—management. All these units are related by means of a series of links that correspond to analysis, monitoring, feed- back and change request information. The applicability of the fractal functional architecture of the FREEDOM contin- uous RE framework is illustrated by using three continuous RE methods. The fractal architecture of the continuous RE framework helps to represent scalability in RE. It also sup- ports the flexibility of RE by providing the ability to adapt to different system development contexts, different software engineering approaches and different requirements manage- ment methods.

The GL found recognises that requirements management, although not listed as a phase in the DevOps cycle, is a key aspects as regards the success of any software development approach [18]. A high-level study of the GL reveals the importance that practitioners place on automating quality assurance and testing in the requirements management pro- cess in DevOps. Practitioners stress the relationship between requirements and tests, in particular by means of the Behav- iour-Driven Development (BDD) paradigm, which contains purpose-specific languages such as Gherkin (c.f. Sect. 3.2, RQ2) and tools such as Cucumber (c.f. Sect. 3.3, RQ3). If BDD helps to make the process of giving users what they need smooth and faster, then it is a fit for any DevOps imple- mentation [86]. Two methodological approaches can be highlighted in this context: (1) KCycle: Knowledge-Based and Agile Software Quality Assurance [81]; and (2) DevOps Plan [82], which includes a continuous auditing process. The GL also pays particular attention to business needs [84, 85] and requirements reuse [18, 83]. Finally, both BizDevOps and DevSecOps are referenced in the GL, signifying that they have transcended formal literature.

3.2 RQ2. What techniques are used for requirements management in DevOps?

As shown first in Table 9 and then in Fig. 8, the most fre- quently mentioned techniques for requirements management in DevOps are agile techniques such as epics and user sto- ries, as they are used in 17 of the studies analysed. This result coincides with the results of RQ1, as agile method- ologies are prevalent in DevOps, and the standard Scrum practice is to use user stories to specify both functional and non-functional requirements [45].

In their work, Furfaro et al. [54] and Hosono et al. [57] employ use cases to specify the functionalities with which business scenarios should be provided, while Furfaro et al. generate use cases from diagrams of goals and sub-goals (SoftGoal and Goals Diagrams) together with the specifica- tion of business rules and regulations, such as relationships between stakeholders, business roles, actors and business goals. Use cases are employed to show the hierarchies and constraints of all the elements involved in the development process. These diagrams provide requirements with the pos- sibility of identifying the actors, implications and processes involved in requirements management within a software project.

The DSPL4DevOps (Dynamic Software Product Line for DevOps) proposal of Nakanishi et al. [63] starts with an initiation process in which requirements and changes in the system to be developed are acquired using Software Product Lines (SPLs) to capture and

describe known uncer- tainties. Once the requirements have been acquired in the above process, the uncertainty model is built by involving developers and operators in order to identify the uncertain- ties in requirements, to study possible solutions by which to resolve the uncertainties and to model them as a decision tree. Furthermore, for each possible solution, developers and operators assess the impact on the adaptive dynamic system, the system configuration and the operation process, which are summarised in the uncertainty assessment table.

In the context of the RADON project mentioned in RQ1, Casale et al. [51] propose a requirements formalism that is close to the concepts and abstractions typical of human thinking. This formalism is easy to translate into standard models for orchestration. In order to address these chal- lenges, RADON introduces a Constrained Definition Lan- guage (CDL) with which to formally specify both functional and non-functional requirements, thus increasing automation in the design of FaaS-based applications.

The techniques for requirements management most fre- quently mentioned in GL are those typical of agilism: a hierarchy that basically consists of epics, user stories and tasks [82, 84, 87, 88, 93]. However, numerous references indicate an evolution of user stories: the concept of feature [81, 84, 91], which constitutes a step forward towards the automation of user stories. A feature is a kind of function of the product or system that delivers business value and fulfils one customer's needs; a feature is basically a user story, but is specified in a type of formal, executable language, such as Gherkin or K + [81]. Gherkin is a business-driven devel- opment language focused on specifying expected business behaviour and can be used to specify user stories together with their acceptance criteria. In this respect, Gherkin serves to automate relationships between requirements and testing. K + is, on the other hand, a kind of semiformal notation for user stories that was inspired by Gherkin.

3.3 RQ3. What software tools are used to support requirements management in DevOps?

As shown in Fig. 9, more than 30 different tools have been proposed for requirements management in the DevOps framework. Table 10 shows the tools organised into a set of categories proposed by the authors of this paper, in which each tool can be mapped onto more than one category if appropriate. In most cases, commercial tools are reported in the selected studies, but some custom software develop- ments can also be found.

The tool most widely used in the Agile Requirements and Project Management Tool category in Table 10 is Jira Software, which is basically an issue management tool that has been adapted to software project management. Efforts have been made to extend Jira with certain functionalities, as occurs in the work by Mittal et al. [60]. In this case, plug-ins are incorporated into Jira in order to cover the integration of requirements into source code by employing plug-ins with tools such as Git, GitHub, GitLab and Microsoft's Azure TFS (Team Foundation Server). This category contains tools that allow continuity in software development, such as that used by Kirikova et al. [17], or that employed by Theunissen et al. [76] in their work on continuous RE, which uses the Confluence tool to collaborate as a team and in the require- ments documentation tasks, thus allowing the generation of reports of the deliverables in agile methodologies such as in Scrum in order to generate sprint reports.

With regard to the Requirements Repository and Inter- change Management Tools category in Table 10, there is, for instance, the proposal by Ali et al. [45] for the reuse of previous development artefacts. These authors developed a repository of reusable software artefacts to support rapid delivery by reusing artefacts during the development pro- cess. This repository includes the requirements of previously developed systems. The artefacts are stored as ontologies for later reuse. When a new software project is presented, which is developed following Scrum, the system require- ments are determined and the repository is queried in order to discover whether similar artefacts exist for reuse, which shortens development time.

Several tools can be found in the Specification Models Management Tools category in Table 10. In their work, Hugues et al. [58] mention that the first step in their Twin- Ops proposal is to define system requirements, sub-functions and use case scenarios. Containers are employed to perform these tasks for the delivery of modelling environments by following the DevOps philosophy; the proposed tools are Papyrus SysML 1.6, the OSATE AADL toolchain and MDT (Modelica Development Tooling). This environment is employed to capture the first set of system models: a col- lection of OMG SysML 1.6 diagrams that capture high-level requirements and use cases. Moreover, Taryana et al. [75] employ Visual Paradigm (VP), which uses UML notation, to deal with use case diagrams, class diagrams and sequence diagrams. VP facilitates the use of reverse engineering from source code changes to the design phase.

The use of tools to manage ontologies is documented by Oliver et al. [64] in the Requirements Knowledge Manage- ment Tools category in Table 10, when using Protégé to specify ontologies to a Prolog rules engine. This engine identifies the description logic of most of the ontologies described, thus allowing the data flow models and annota- tions to be coded so as to obtain a domain-specific language. When a requirement becomes difficult to specify, and risks outside the scope of the system start to be addressed, this study discusses how to reduce the requirements in order to better reflect the current development context and the risk allowed. This is possible by making a formal link between ontologies, requirements and risk. This work proposes two stages in the requirements reduction process: (1) weakening the requirements to a point at which the system is imple- mentable and (2) continuing to weaken them in order to allow the greater use and collection of information within the limits of the risks that the organisation is willing to take, typically for future testing and other new business needs.

The Formal Requirements Language Support and Analy- sis category can be illustrated with the Service Lambda and the BeSoS applications (Behaviour-driven and Scenario- based Requirements Modelling for Systems of Systems). Service Lambda [57] includes an IDE which has tools that assist in the requirements definition, architecture design and prototyping of applications, and simulate the execution of large-scale applications on developers' PCs. The so-called Service portfolio incorporates data from these tools and enables automatic data exchange among them, thus avoid- ing problems and redundancies. Furthermore, the BeSoS tool [79] serves to create formal requirements models using scenario-based modelling techniques. The use of the Jenkins automation server subsequently permits the creation of a continuous requirements validation process with which to automate the process and obtain rapid feedback

on newly added requirements, thus creating a strong link between the stakeholders' point of view and systems engineering.

The Traceability Management Tools category in Table 10 includes the Software Artefact Traceability Analyser (SAT- Analyser) tool proposed by Rubasinghe et al. [70] in order to support DevOps practices with traceability, considering test, configuration and deployment artefacts for traceability management within DevOps practices.

The Microsoft Office or similar tools category describes the adaptation of general-purpose informatics tools in order to manage basic aspects of requirements management, such as word processors, spreadsheets and text presentation soft- ware. This category makes sense when recalling that RE tools are sometimes said to be expensive, fairly complex to learn and difficult to integrate with other tools used in DevOps [18].

General tools for requirements management are cited in the GL [18], such as IBM Rational Doors [85], but the GL pays particular attention to tools that enable BDD, such as KCycle [81] and that which is most commonly used, Cucumber [86]. Agile tools such as JIRA [90] and Conflu- ence [90] are also cited in the GL. Moreover, one remark- able tool mentioned in the GL is Microsoft Azure DevOps, which manages requirements by employing the so-called Azure Boards, which encompass business processes (epics) and business process steps (features) [84, 93]. The activi- ties described in BABOK (Guide to the Business Analysis Body of Knowledge) can be managed by means of Azure DevOps (i.e. Approve Requirements, Trace Requirements, Maintain Requirements, Prioritize Requirements and Access Requirements Change).

3.4 RQ4. What methods have been used for the validation of the methodologies, techniques and tools identified in RQs 1–3?

With regard to the methods used for the evaluation of the studies selected from formal literature (RQ4 in Table 7), the most widely used evaluation method is that of case studies, with 16 studies, followed by experience reports, with 9 stud- ies. A number of studies (21) do not provide information on the evaluation method followed, if any. The remainder of this section shows some examples of how the evaluation methods are put into practice.

With regard to the case study evaluation method, Gupta et al. [55] conducted a successful case study in a health- care organisation in order to adopt continuous delivery and DevOps in the software engineering project of a team spread across three countries (USA, Germany and India). The authors reported their lessons learned and recom- mendations from a project manager, quality manager and architect´s point of view. In their findings, Gupta et al. [55] uncovered the challenges involved in adopting continuous delivery and DevOps, and they described the need to focus on automation as a critical element by which to achieve con- tinuous delivery. Leaving aside manual processes, in order to obtain continuous and quality requirements it is necessary to improve processes and operations by implementing project incremental workshops in which all members involved in the development process are included. Furthermore, Stray et al.[74] developed a case study on dependency management in large-scale agile DevOps teams. This study consisted of observing 38 meetings, followed by interviews with mem- bers of five DevOps teams. These authors used a taxonomy of dependencies to explore agile practices that act as coor- dination mechanisms in a large-scale project.

With regard to experience reports, the proposal by Morales et al. [61] can be found in the area of software development in regulated environments. This approach was described through the use of a report on the implementa- tion of DevOps practices in Highly Regulated Environments (HREs). One constraint imposed in an HRE in relation to requirements is that there is no agile process when approv- ing the development of a project. In addition, the absence of a centralised communication repository and the incorpora- tion of new requirements into the Software Development Lifecycle (SDLC) do not allow the direct implementation of DevOps. These authors propose a DevOps assessment process in HRE with which to identify bottlenecks.

No proposals were found in the category of evaluation through proof of concept in formal literature; however, the

GL contained the proposal by Lange et al. [93], which shows an example of the configuration of Azure DevOps in order to perform requirements management.

Finally, a few studies use laboratory experiments, such as the proposal by Haindl et al. [56], whose main objective is to discover the performance of various TAICOS (Tailoring Stakeholder Interests to Task-Oriented Functional Require- ments) language components in order to reliably evalu- ate feature constraints with measurements. Moreover, Ali et al. [45] conducted a controlled experiment with which to measure the effectiveness and efficiency of their proposed hybrid DevOps process with systematic reuse. A pilot pro- ject consisting of a mobile application in the health area was developed throughout the experiment with the objective of monitoring the health of a patient with a cardiovascular disease. The results of this proposal revealed an average gain of 35.2% in terms of function points (which measure the size of the software) by reusing 30.63% of the reusable artefacts available in the repository.

4 Discussion

Software development has become a rapidly accelerating business confronting the challenges of demanding contexts, changing customer requirements and an increasing need for short development cycles and fast time to market. Software development organisations should be able to produce high- quality software while maintaining an innovative edge [59]. DevOps has emerged in order to enable the continuous evo- lution of the system on the basis of the requirements encoun- tered in the operation of the system. DevOps is a paradigm that makes development and operation teams work closely together, feeds the requirements acquired during operation back to development and improves the system in a shorter delivery time [63]. According to Sinkala et al. [73], the incorporation of DevOps into agile environments inherited the challenges related to agile RE.

Our secondary study provides an insight into the state of DevOps in requirements management by basically consider- ing methodologies, techniques and tools, and this, therefore, led to the decision to structure the discussion in terms of RQs 1–3. We believe that RQ4 has been sufficiently dis- cussed in Sect. 3 and does not merit a special subsection in this Sect. 4. The discussion has been arranged in several On the need for… epigraphs that the authors of this paper have found to be of interest. These epigraphs came about itera- tively as a synthesis starting from the experience acquired during the analy-

sis of the results. Furthermore, it should be noticed that the close relationships among methods, tech- niques and tools sometimes signifies that the same study may have implications for more than one RQ, as the study may contribute with, e.g. a methodology plus the extension of an existing technique and tool support.

In our view, this section can help both researchers and practitioners, but it is undeniable that it is focused more on existing gaps in research. Nevertheless, we believe that practitioners could also benefit from the epigraphs in "Dis- cussion" by considering them as a list of topics to which attention should be paid during the implementation of the RE process in DevOps. The GL has been included in the MMS in order to bring issues drawn from the GL sources that may be closer to practitioners than to Academy onto the research agenda.

One important topic that has been ignored in the selected studies is sustainability. This is a broad term that could imply the extension of methodologies, techniques and tools, thus concerning RQs 1–3 transversely. Sustainability is a multidimensional aspect (with environmental, economic, individual, social and technical dimensions) that must, according to the United Nations' 2030 Agenda for Sustainable Development, be improved. In the near future, we expect literature to contain more efforts with which to take into account sustainability from the early stages of develop- ment in mainstream DevOps environments, starting from the Kalskrona Manifesto [95] and the first proposals in the field of RE and sustainability [96].

4.1 RQ1. What methodologies and frameworks are used for requirements management in DevOps?

4.1.1 On the need for requirements management methods regarding specific vertical and horizontal domains

As can be observed in the results related to RQ1, there are methodologies that make it possible to follow DevOps in horizontal domains (i.e. domains that concern software transversely, such as security, safety or privacy). With regard to security, the proposal which stands out is clearly DevSecOps [21], but more studies in this domain can also be cited, particularly the work by Ayerdi et al. [48], which shows a taxonomy with which to elicit the design operation of continuous security requirements. In this work, require- ments can be elicited in two steps. Firstly, generic require- ments are derived, which can be applied to any cyber- physical system. Secondly, these generic requirements are exemplified for specific systems, by building: (1) a release plan, i.e. language requirements with which to specify the configuration and the process of building the software; (2) an implementation plan, i.e. the requirements for a language with which to specify all the steps related to the implementa- tion; and finally (3) a validation plan.

Our results show more concerns regarding security issues. For example, Atighetchi et al. [47] mention that as organisations move towards a more complete adoption of agile software development and DevOps for mission critical systems, the need to continuously and rapidly assess cyber security requirements will become increasingly important. Security assessment tools currently focus on performing Cyber Vulnerability Analysis (CVA) on software designs and artefacts rather than on requirements, skipping critical steps required in order to examine requirements and support design decisions early in the development lifecycle or as a change in requirements.

Also with regard to security, Casola et al. [52] recom- mend adding an actor who is an expert in security when it is necessary to make decisions in order to address spe- cific security issues (e.g. to select a security technology, to address specific security requirements, etc.), to select a security technology or to address specific security requirements. But attention should be paid to modern development methodologies, as they require new methods and techniques that will: (1) allow security to be taken into account in each development iteration, possibly based on a security-by- design approach; (2) not negatively affect the speed and flexibility of the development process; and (3) consist of short, mostly automated tasks that can be executed even by personnel with a basic knowledge of security. Furthermore, in the GL, Curiel [83] states that the risk that personal data could be intercepted or compromised in each step of the application architecture requires the application of security requirements that match the expected risk. This study pro- poses a set of guidelines denominated as the Application Security Verification Standard (ASVS Guideline), which consists of a list of requirements and tests that can be used to define what a secure application is, thus enabling the reuse of requirements in the security realm.

In addition to horizontal domains, some concerns regarding vertical domains (i.e. domains that encompass a specific kind of applications, such as banking, automotive or medi- cal apps) have also been found. Gaps have been detected in organisations developing software for medical devices, as they must comply with regulatory requirements and international standards if they wish to commercialise their devices. The MDevSPICE (Medical Device Software Pro- cess Assessment Framework) proposal by Özcan-Top et al.[66] integrates software requirements from various inter- national medical device standards and guidance documents with best practices for software and system development into a single reference source. But this proposal does not allow a complete adoption of agile methodologies such as Scrum, because these authors state that medical devices need the approval of requirements from the beginning, and changing requirements during the sprints is not, therefore, allowed. The rigorous processes in the medical domain make it difficult to make a modification if, for example, a require- ment changes, but it cannot be aligned with the standard chosen by the developer company. A solution for the medical software requirements obtained must be analysed prior to development. This analysis generally includes identifying and prioritising conflicting, missing, incomplete, ambiguous, inconsistent, incongruent or unverifiable requirements. In order to achieve this, the definition of a so-called ready practice is recommended, which is an agreement that details the set of conditions that must be met in order to consider that a requirement is ready to be included in an iteration for delivery [66].

4.1.2 On the need for requirements management methods encompassing business needs and processes

Some proposals extend DevOps by exploring adjoining areas that can be taken into account, such as business pro- cesses (BizDevOps). There is a set of proposals regarding BizDevOps that relates DevOps to business needs and pro- cesses. BizDevOps allows people in business departments to express and review requirements in a practical way and therefore reduces the knowledge transfer required from the business to IT and provides the fastest possible feedback cycles (i.e. supporting the "Biz" in BizDevOps). BizDevOps enables

IT departments to control the entire application development process in order to ensure a high quality of software artefacts (the "Dev" in BizDevOps). BizDevOps provides an integrated and automated toolchain integration with which to allow increased automation and thus the pace of development (the "Ops" in BizDevOps) [53].

BizDevOps inherits requirements specification issues from DevOps, which leads to development and operations teams taking requirements into development; for example, when requirements are created solely by business people, they tend to be written in a way that developers cannot understand, thus creating the need for further clarification. In addition, developers may encounter technical constraints that make it infeasible to fully meet a requirement, thus cre- ating the need to report the problem to the business area and wait for a decision. It should be noted that business people often tend to be constantly busy and unavailable, creating long lead times for development teams, and even leading to situations in which the development team is forced to make product decisions with no business basis so as to meet deadlines [62].

In their work, Moreira et al. [62] mention that in order to solve specification problems it is convenient to create the following roles within the BizDevOps process: (1) the Busi- ness tracker, who comprises all activities related to the crea- tion and continuous maintenance of a list of requirements, such as user stories, product backlogs and feature lists. The business tracker must ensure that project development is always guided by the requirements list, and must coordinate the stakeholders in order to define the requirements along with their importance for and risk to the project; and (2) the

Business ambassador, who comprises all activities related to the translation of user needs into business requirements and the translation of these requirements into features to be developed. Other activities include negotiating timelines with the customer. The business ambassador is expected to be able to explain technical constraints and resource alloca- tion in such a way that customers can understand the require- ments and each deadline can be met. Business ambassadors should inform developers about the priority of each require- ment, thus allowing them to invest their efforts in maximising value delivery.

Finally, Wiecher et al. [79] include a BizDevOps approach with which to support the continuous updating of requirements. The project environment (Biz) is, therefore, supported by defining scenarios that describe the expecta- tions regarding the project outcome and the possible impact. These high-level scenarios are subsequently used in order to derive concrete, executable system scenarios (Dev). These executable system scenarios are then used as first prototypes of the system under development that can be validated and aligned with the project environment (Ops).

4.1.3 On the need to improve requirements communication among roles

The seamless communication of issues concerning require- ments in DevOps is a critical issue. It is for this reason that Sinkala et al. [73] make suggestions regarding requirements activities in DevOps, while stressing the importance of com- munication. When processing requirements, the proposed solutions for requirements analysis should be validated, and these should be cross-communicated between roles. The solution can be in the form of a set of attributes such as testability and integrability, which should be verified in the requirements hierarchy. This is seen in hierarchy-based strat- egies, in which more concrete resources and techniques are brought to the requirements analysis and breakdown process. However, since it is driven by the fast pace of delivery and the high frequency of testing, requirements analysis must not only identify dependencies between requirements, but also communicate them. This can, therefore, be facilitated by involving multiple experts in this process. One important impact is that of having multiple requirements authors in order to reduce the size of the requirements structure and handovers, and to allow more developers to participate in this process. Nevertheless, concerns have been expressed regarding the level of system knowledge, as this appears to be a challenge for large-scale systems.

The solutions for requirements analysis and communication among roles may imply requirements metainfomation in the form of a set of attributes, such as comparability and comprehensiveness. However, as it is driven by the fast pace of delivery and the high frequency of testing, requirements analysis must not only identify dependencies between requirements, but also communicate them. This should be performed by involving multiple experts in this process [73]. The role of requirements analysts with technical back- grounds should be included [62] in the development team. These people must be able to understand and communicate well with both developers and customers, and be able to write requirements by adding technical details. Require- ments analysts will, therefore, work as a bridge between developers and the business side, gathering requirements from customers and translating them into user stories or use cases that developers can understand, and communicating the development team's questions for the business side to answer.

According to Oliver et al. [64], clear communication is fundamental in any development environment in which the inclusion of specific legal requirements increases the com- plexity of communication, in addition to the usual combina- tion of customer and engineering requirements. One of the main issues confronting privacy is the use of terms such as personally identifiable information and personal data [49], which have legal definitions, despite the fact that it is com- plex to apply these broad terms to the variety of situations that occur in information systems.

A lack of documentation is one effect of the lack of com- munication between development and operational teams [50], and eventually with the business area [62]. A closer communication with the business area can improve the pro- ductivity of the development teams by reducing the waiting time for requirements clarification. What is more, having additional business knowledge in the development teams allows more flexibility through a better re-prioritisation process. In addition, Forbrig et al. [53] identify that communi- cation among stakeholders is the most important aspect in development. Social, cultural and communication skills are necessary for all stakeholders. Knowledge representation is very important; it would be helpful if domain experts could express their ideas in a semiformal notation that would allow transformations and generations.

4.1.4 On the need to support requirements reuse in continuous processes

Software reuse, and requirements reuse in particular, has attracted the attention of the research community for more than 25 years. Software reuse is a way in which to increase quality and productiv-

ity, and there is a consensus that the benefits of reuse are greater when the abstraction level is increased and not only code, but also designs and specifi- cations, are reused. Requirements reuse has been a classic research topic in the RE community. We wonder whether this could be the reason why it seems that GL rather than formal literature shows more interest in how to put requirements reuse into practice in DevOps, although the GL provides only a few practical tips regarding this issue.

The only work dealing with requirements reuse in DevOps that can be found in formal literature is the research by Ali et al. [45], which presents a complete agile-based DevOps process with a systematic reuse-based software development and management process. The process is inte- grated into the Scrum framework by including a set of activi- ties around a requirements repository, in which an ontology enhances artefact retrieval. This process can be integrated into DevOps through the consideration of operations and customer feedback.

The GL studied contains several contributions that stress the importance of reusability as a means to improve both quality and productivity, but they do not propose methods, techniques or tools with which to include requirements reuse in DevOps. The DevOps Plan [82] proposes the use of meta- data requirements to ease the detection and reuse of software artefacts. As mentioned above in this section, Curiel [83] proposed a reusable requirements and test list, together with guidelines that can be used to define what a secure applica- tion is (the so-called ASVS, Application Security Verification Standard).

4.2 RQ2. What techniques are used in requirements management in DevOps?

4.2.1 On the need for requirements techniques regarding traceability

The traceability management of software artefacts is a key challenge in DevOps environments owing to the continuous changes of artefacts, which can often make these artefacts inconsistent [70]. Traceability management includes both:

(1) backward traceability in order to update requirements as a consequence of an incident in the production environment; and (2) forward traceability in order to propagate changes in requirements to subsequent software development artefacts. DevOps should provide a fast feedback cycle and communi- cate the changes as soon as possible.

In their work, Taryana et al. [75] argue that traceability is required in order to permit Continuous Development (CD) to run simultaneously from analysis to test or vice versa, while Rubasinghe et al. [70] provide a proposal for a trace- ability management tool dealing with continuous integration in DevOps practices. Traceability is also recognised as being relevant for the DevOps Plan methodology [82]. Moreover, when considering software artefacts related to testing, con- figuration and deployment, the SAT-Analyser tool is able to manage traceability links between some of the early phases of the software development lifecycle. The novel contribu- tion of this work is that the proposed traceability process model consists of artefact change detection, change impact analysis and change propagation. In addition, this tool pro- vides multiuser accessibility and is integrated with a leading DevOps tool stack in order to enable collaboration.

Knowledge dependency requirements are also related to traceability and are a critical input in software develop- ment. According to Stray et al. [74], if domain knowledge or details of the requirements are not known and must be located or identified, then the progress of the project may be hindered. Requirements prioritisation in agile projects and access to customers are vital: knowledge dependency occurs when team members are waiting for information about a requirement, a task, technical information, a previous deci- sion, or because they do not know what other team members are doing.

4.2.2 On the need for requirements techniques regarding ambiguity and testing

One of the benefits of avoiding ambiguity in requirements is that of allowing proper testing. In this respect, the use of semi-structured user stories [81] or languages such as Gher- kin [86] have been proposed as solutions by which to reduce ambiguity and generate test cases automatically.

According to Sinkala et al. [73], poor requirements man- agement can cause errors in system integration testing and can, therefore, delay software delivery. In this respect, Ravi- chandran et al. [67] suggest that active flowcharting helps to eliminate ambiguity in requirements and reduce defects in the design phase. These flowcharts will generate the smallest set of automated tests required for maximum coverage. They also help test teams know which features should receive the most rigorous testing on the basis of analysis and metrics collection capabilities. Moreover, microservice architec- tures have been emphasised in order to promote autonomous teams working on both requirements and testing [73].

4.2.3 On the need for requirements techniques regarding the proper treatment of non-functional requirements

Developers should define and consider NFRs (sometimes known as quality requirements or operational requirements [90]) properly in the early stages, otherwise it becomes very complex and costly to address them later on. However, Ols- son et al. [97] did not find any approaches that integrate quality requirements within DevOps, although there are some studies addressing quality requirements and agile methods.

Our results show approaches with which to deal with NFRs that pursue continuity in the development process. In this respect, in their proposed CI (Continuous Integration) framework for NFRs testing, Yu et al. [80] provide an over- view of the correlations between CI and NFRs, along with information on which software tools can be used in each component of the CI framework. The tools in this CI frame- work are useful for teams or organisations when they have to choose tools with which to build a CI environment, and the NFRs in the framework make it possible to see which NFRs are affected in specific CI components by using appropriate tools.

Moreover, Wahaballa et al. [78] present the FURPS model, which is proposed in order to unify and classify functional requirements and software quality attributes (i.e. NFRs). FURPS stands for functionality, usability, reliability, performance and support. With regard to quality require- ments, Behutiye et al. [50] argue that having a good defini- tion in quality requirements helps during decision making; these authors report that several roles can be found in soft- ware organisations, such as (1) the Requirements specifica- tion engineer,

who is in charge of discovering requirements that can impact on the product, i.e. how many users will use the application; and (2) the Software architect, who in some cases is not familiar with quality requirements, which can lead to bad practices during which requirements are invented or certain assumptions are made that impact on the performance of the product. A checklist of tasks and having a canonical way of looking at the NFRs help with the documentation of quality requirements.

One of the means proposed to deal with NFRs is that of specification languages. In this respect, the Casale et al.[51] proposal called RADON allows developers to analyse the specification model using a hierarchy of logic program- ming and simulation techniques to determine whether the decomposition or aggregation of certain services produces an improvement to the fulfilment of the specified require- ments. This mechanism will allow progressive decisions to be enacted on the model, resulting in a solution with optimal decomposition and the satisfaction of security and privacy policies, in addition to the usual NFR performance and cost requirements. Monitoring feedback will also be available in a dashboard for users in order to diagnose the runtime behaviour of each application component and semi-automat- ically identify that should be prioritised in the design. Fur- thermore, Haindl et al. [56] present the so-called Language with which to evaluate feature-dependent non-functional requirements, whose objective is to automatically specify and evaluate feature-dependent NFRs through the use of quantitative constraints. The language not only makes it possible to express the criteria for the fulfilment of these constraints, but also allows the acquisition of the measures required and their expected evolution over time. It provides a comprehensive set of time series operations and time fil- ters of different ranges and supports metrics and ordinal threshold values. This allows the continuous monitoring of detailed NFR compliance at the level of individual functions in DevOps, thus providing an effective means to counteract quality degradations at the function rather than at the prod- uct level.

4.3 RQ3. What software tools are used to support requirements management in DevOps?

4.3.1 On the need for requirements management tools

A well-established tool infrastructure supports the deliv- ery of software releases on a continuous basis. Tools can improve the delivery cycle, ranging from facilitating require- ments management, simplifying day-to-day development activities, quality assurance or speeding up deployment to handling rapid user feedback. Organisations that employ such tools in practice could gain additional development momentum that would allow them to shorten release cycles and accelerate the delivery of new software to evolving mar- kets. The toolchain should be as complete as possible in order to enable day-to-day implementations. However, it is also necessary to note that even a good toolchain does not guarantee rapid deployment [59].

Although tools are paramount in DevOps, Mäkinen et al. [59] show that tools are uncommon in requirements elicitation, artefact repository, quality attribute testing and acceptance testing. Requirements elicitation, for example, is mainly done without tools, and many organisations reported that acceptance testing is done manually. JIRA is used to track development and project management tasks, although it does not natively provide functions with which to track testing activities such as a test plan, a test execution cycle and test cases. It is necessary to have an add-on that pro- vides not only the functions mentioned above, but also a link to test automation tools such as Cucumber, SpecFlow, JBehave and CI/CD tools such as Jenkins and Bamboo [60]. Practitioners also report a lack of use of requirements management tools in DevOps. For example, Sardinha [84] states that managing requirements without proper tools will not provide developers with any business value with which to make informed decisions about priorities or the impacts of changes on processes. In industry, it is not unusual for requirements to be simply compiled in a list and managed on various spreadsheets.

In order to evaluate the tools, it is necessary to answer the following questions: are these tools available to developers and do they work effectively? Are their tool sets multiple for the same purpose? Can the current tool set be reduced in order to exclude those that are not favoured by develop- ers? Does it provide well-qualified long-term staff who are assigned to specific job functions, such as IT infrastructure, software implementation, programmers for specific lan- guages, unit testers and system integration? A properly documented and easy-to-configure toolchain enables a company to start development quickly and makes it possible to rapidly deploy new software to customers [59]. Finally, but of no less importance, traceability is another important aspect to be taken into account when choosing a DevOps tool [82].

4.3.2 On the need for requirements managements tools supporting continuous practices

One mandatory aspect of continuous implementation is a seamless pipeline from code to delivery. In order to achieve continuous deployment, a software company must set up an automated toolchain and establish tool-related work- flows [59]. DevOps advocates continuity, which should also encompass requirements. For example, the S-BPM technique is employed in the work by Forbrig et al. [53] to cover some aspects of continuous software engineering: human-centred design, continuous requirements engineering and continuous business process modelling. Furthermore, it might be useful for business domain experts to have the chance to specify their ideas in different graphical or textual notations at several abstraction levels.

Our results contain efforts such as that of Furfaro et al. [54], who propose ResDevOps, which has the ability to have continuous control over the whole evolution of EIS (Enterprise Information System) requirements over time. One potential weakness of this proposal could be the addi- tional effort required to build the first GOReM model of the overall EIS, although the methodology is easy to apply and is effective. In addition, a complete EIS model must be built incrementally. Another case of continuity in requirements is the proposal of Kirikova et al. [17], which comprises the FREEDOM continuous RE framework with five functions:

(1) requirements acquisition; (2) requirements analysis; (3) requirements representation; (4) requirements management; and (5) requirements validation. The fractal architecture of the continuous RE framework helps to represent scalability in RE. It also supports the flexibility of RE by providing the ability to adapt to different system development contexts (i.e. diverse agile methodologies), different

software engineering approaches and different requirements management meth- ods. To continue with continuity in requirements, Taryana et al. [75] focus on direct and reverse engineering, which are part of the Dev stage performed owing to an improvement to the developer's own source code or to a new customer. Once a DevOps cycle is attained, development work can be performed concurrently with operations. The requirements to be addressed in an iteration are usually captured in a task planning tool; it is not advisable to duplicate requirements, i.e. to specify them elsewhere. The requirements captured are often functional requirements, or specific development- related tasks derived from functional requirements.

In today's software development, prioritising work and knowing what needs to be done relies on continuously maintaining a backlog, which is essentially a cumulative collection of work to be done [59]. A proper backlog man- agement tool is an essential part of software development.

Finally, it is worth once again highlighting the rel- evance of obtaining real-time feedback from monitoring tools in continuous improvement [19]. Requirements can be changed at a lower cost only by means of appropriate metrics regarding how those requirements have been turned out in production.

4.3.3 On the need for requirements management tool managing documentation in continuous processes

The prevalence of DevOps further accelerates the pace of the development and delivery of features with shorter cycle times. However, most DevOps practices focus much more on feature (code) development than on documentation, result- ing in a rhythmic mismatch between feature and document deliveries. As a result, documentation becomes a serious bottleneck to the rapid delivery of value to customers [69]. Lack of documentation can have an impact on the success of software [50].

According to Behutiye et al. [50], agile software develop- ment promotes minimal documentation and often prioritises functional requirements over NFRs. Minimal emphasis on documentation can be beneficial as regards reducing soft- ware time to market, but it can also be a concern, espe- cially in the case of NFRs, as they are difficult to specify and document and are crucial to the success of the soft- ware. Practitioners identify the documentation of NFRs as being important in order to clarify the meaning and scope of NFRs. This is a key aspect, as requirements often change and NFR specifications can evolve over time.

In a similar vein, Sharma et al. [72] indicate that agile methodologies do not encourage heavy documentation for requirements gathering. Documents such as the System Requirements Specification (SRS) have, therefore, almost disappeared in these contexts. Agile methodologies have greatly simplified the task of requirements gathering by introducing the ability to handle frequent changes in busi- ness requirements with the help of short delivery cycles. Agile frequently suggests capturing the requirements through user stories, which have a drawback as they are high level and do not usually capture all the details of the require- ments. In order to make user stories more accurate, accept- ance criteria are, therefore, also documented in an attempt to ensure that the agile team clearly knows what conditions must be met for the working software to be accepted.

In their work, Rong et al. [69] propose DevDocOps for continuous automated documentation. DevDocOps extends the scope of DevOps, improves value delivery by support- ing continuous documentation and bridges the gap between feature delivery and document delivery with automation. DevDocOps should improve accuracy, consistency and timeliness in the documents generated. Several roles in DevDocOps create supporting documents throughout the development process. For example, software developers are more familiar with the features they have developed. They are able to provide the supporting documents with the nec- essary updated information without further distortions in knowledge transfer. The original technical writers, who are well versed in terminology and regulations, can then concen- trate on tasks such as polishing, reviewing and checking the content and formatting the supporting materials, which further improves the final quality of the supporting documents. Within a delivery chain, a set of templates is created to col- lect and transform the required information from its source to the target documents for delivery. Different templates were designed in order to guide and constrain developers to provide the necessary information for documentation.

Continuous documentation should also encompass busi- ness processes. According to Kum-Seun [85], requirements in BizDevOps are collectively and collaboratively owned and managed by both the business and IT. Requirements should, therefore, be documented by means of a platform that enables teams to notice what is being committed, describing how their work supports business goals. For instance, the use of a wiki platform like that of Atlassian Confluence could be helpful to promote collaboration and to maintain documentation relevant for everyone [90].

5 Threats to validity

The actions taken to minimise the impact of the threats to validity are described in this section. A list of threats to validity that encompasses descriptive validity, theoretical validity, generalisability, interpretative validity and repeat- ability was proposed by Petersen et al. [24]. Nevertheless, in this study we have preferred to follow the more detailed framework by Ampatzoglou et al. [98], in which literature is reviewed in order to identify threats to validity and a set of common mitigation actions.

5.1 Study inclusion/exclusion bias

This threat to validity refers to issues that might occur when applying the inclusion and exclusion criteria. Only the first author of this paper conducted the whole selection process and important studies might, therefore, have been missed. In order to reduce this threat and improve the confidence in the set of selected papers, a second round of peer review was conducted after a first pre-selection of 82 candidate papers. This threat has, therefore, been mitigated by means of the participation of the three authors of the paper in the selection process. Each author evaluated each RQ on the 82 candidate studies independently: if one author thought that the study dealt with any of the RQs, then this study was proposed to be included by that author. In the case of discrepancies, that study was discussed by all the authors until an agreement was reached. The set of inclusion and exclusion criteria had been documented beforehand in the review protocol. What is more, this set of criteria was inspired by the guidelines of Kuhrmann et al. [41] to be in line with the best practices in the field.

5.2 Construction of the search string

In order to mitigate potential problems concerning the build- ing of the search string, an iterative search string construc- tion process was followed, which was inspired by the PICO criteria. This process was performed by the first author of this paper and supervised by the second and third authors. Furthermore, in order not to leave out any potentially inter- esting paper, the search string was defined in rather a broad manner, signifying that the inclusion and exclusion criteria were relied upon in order to filter out relevant studies.

5.3 Data extraction bias

The identification of incorrect data and the identification of incorrect relationships may lead to the formulation of incorrect conclusions. The data was, therefore, extracted from the selected studies by the first author of this paper, but the second author also performed data extraction in an independent manner. Disagreements were then discussed by all the authors until a consensus was reached.

5.4 Selection of digital libraries

This issue refers to problems that may arise when using very specific, too broad, or not credible search engines. This threat was first mitigated through the inclusion of the best- known digital libraries in the RE field. The search of formal literature was performed by using IEEE Digital Library, ACM Digital Library, Science Direct, SpringerLink, Wiley online library and Google Scholar. Most of the research on RE can, according to the statistics regarding literature search engines, be drawn from these electronic libraries [99]. Sec- ondly, no temporal limits were specified in the inclusion and exclusion criteria for both the formal literature and the GL. Thirdly, including GL in the review is one of the mitigation actions that address this threat of the Selection of digital libraries. With regard to GL, Garousi et al. [23] claim that the quality of GL is more diverse and often more laborious to assess. It was for this reason that not only Google but also two credible sources, which were the Requirements Engi- neering Magazine and DevOps.com, were used as sources of GL. Finally, two criteria on the QA checklist of GL for software engineering provided by Garousi et al. were taken into account in order to select resources from the GL: (1) authority of the producer and (2) methodology.

5.5 Robustness of initial classification

This concerns the classification schema proposed. With regard to this threat, we understand that two issues are debat- able: (1) the categories proposed in order to classify the results obtained for RQs 1–3 (Sect. 3); and (2) the epigraphs employed in order to structure the Discussion (Sect. 4). We are conscious that no standard taxonomies have been used to support these categorisations, but extensive discussion among the authors was carried out until a consensus was reached, based on their experience in RE.

5.6 Researchers' bias

This potential threat refers to the possible bias that the authors of the secondary study may have had while they interpreted or categorised the results. In order to reduce this bias, all the authors again agreed on the discussion of each RQ and the conclusions of the secondary study.

5.7 Repeatability

This threat concerns the possibility of not being able to repeat the results. This process was made as repeatable as possible by putting into practice several of the actions pro- posed by Ampatzoglou et al. [98], namely (1) developing a review protocol; (2) ensuring conformance to well-estab- lished guidelines, such as those of Petersen et al. [24, 28] and Garousi et al. [23]; (3) documenting the search process; (4) involving more than one researcher in the process; (5) documenting the inclusion and exclusion criteria; and (6) documenting the review process.

5.8 Publication bias

This threat is related to the possibility of the majority of the primary studies being identified in a specific publica- tion venue. Moreover, the results of the secondary study might be biased as the result of inaccurate primary stud- ies. In this respect, it is recognised that publishing negative results may be more difficult and that publication bias may, therefore, arise. In order to mitigate this threat, the authors of this study decided to include GL and to perform an MMS. Furthermore, the time coverage of the publications is broad, and the number of publication venues in which the selected studies were found was high.

5.9 Generalisability

Lack of generalisability refers to the possibility of not being able to generalise the results of the study. In order to pro- mote generalisability, both academic and industrial papers have been considered in this MMS, as suggested by Ampat- zoglou et al. [98], together with a broad time and publication coverage, as stated above.

6 Conclusions

The constant deployment of software, which may even occur several times a day, requires software development teams to have clear requirements for the functions they are working with. Without a steady flow of requirements at the begin- ning of the implementation pipeline, the development team will not be able to maintain a steady pace. Regardless of the pace, keeping track of all the data associated with the requirements is an essential part of software development. DevOps has been widely adopted in the software industry owing to its ability to deliver continuous quality deliverables and its flexibility as regards communication between devel- opment and operations teams. According to Moreira and Schneider [100], DevOps and agile currently constitute one of the major challenges for RE, as more and more companies rely on approaches based on direct communication rather than on documentation, and deal with requirements that are not as visible and explicit in a specification as in traditional approaches. In the MMS carried out in this paper, repre- sentative studies have been selected in order to explore the methodologies, techniques and tools that require attention in requirements manage-

ment in DevOps.

The results show that there is no specific methodology for the requirements management process in DevOps. This secondary study opens up the possibility of design- ing methodologies, techniques and tools that will involve people, processes and artefacts in order to improve the RE process in a continuous software process. A specific RE approach for DevOps should support the continuous and sustainable delivery of requirements, requirements reuse, the proper management of quality requirements, and the provision of clear and up-to-date requirements documenta- tion in a continuous RE process integrated into the DevOps toolchain. This new or customised methodology should support requirements traceability by contemplating require- ments elicitation, code development aligned to requirements and project testing by taking into account functional and non-functional requirements. Requirements traceability would also be necessary in order to improve communica- tion concerning requirements and to involve not only the development but also operations and business teams. This would, therefore, make it possible to improve the elicitation of requirements that both business and operations areas can provide to software development.

Acknowledgements This research is part of the OASSIS-UMU (PID2021-122554OB-C32) project, supported by the Spanish Minis- try of Science and Innovation and the European Regional Development Fund (ERDF).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability All data generated or analysed during this study are included in this published article, through the following links: shorturl. at/bfTX3, shorturl.at/guDL4 and shorturl.at/clKX7.

REFERENCES

1. Leite L, Rocha C, Kon F, Milojicic D, Meirelles P (2019) A survey of DevOps concepts and challenges. ACM Comput Surv. https://doi.org/10.1145/3359981

2. Lwakatare LE et al (2019) DevOps in practice: a multiple case study of five companies. Inf Softw Technol 114:217–230. https:// doi.org/10.1016/j.infsof.2019.06.010

3. Jabbari R, Bin Ali N, Petersen K, Tanveer B (2016) What is DevOps? A systematic mapping study on definitions and practices. ACM Int Conf Proc Ser. https://doi.org/10.1145/2962695.2962707

4. Nicolau de França BB, Jeronimo H, Travassos GH (2016) Char- acterizing DevOps by hearing multiple voices. ACM Int Conf Proc Ser. https://doi.org/10.1145/2973839.2973845

5. ZulfahmiToh M, Sahibuddin S, Mahrin MN (2019) Adoption issues in DevOps from the perspective of continuous delivery pipeline. ACM Int Conf Proc Ser 1479:173–177. https://doi.org/ 10.1145/3316615.3316619

6. Sturm R, Pollard C, Craig J (2017) DevOps and continuous deliv- ery. Application performance management (APM) in the digital enterprise. Elsevier, Amsterdam, pp 121–135

7. Luz WP, Pinto G, Bonifácio R (2019) Adopting DevOps in the real world: a theory, a model, and a case study. J Syst Softw. https://doi.org/10.1016/j.jss.2019.07.083

8. Schön EM, Thomaschewski J, Escalona MJ (2017) Agile require- ments engineering: a systematic literature review. Comput Stand Interfaces 49:79–91. https://doi.org/10.1016/j.csi.2016.08.011

9. Heikkila VT, Damian D, Lassenius C, Paasivaara M (2015) A mapping study on requirements engineering in agile software development. In: Proceedings - 41st Euromicro conference on software engineering and advanced applications, SEAA 2015, pp 199–207. https://doi.org/10.1109/SEAA.2015.70

10. Kupiainen E, Mäntylä MV, Itkonen J (2015) Using metrics in agile and lean software development - a systematic literature review of industrial studies. Inf Softw Technol 62(1):143–163. https://doi.org/10.1016/j.infsof.2015.02.005

11. Ramesh B, Cao L, Baskerville R (2010) Agile requirements engi- neering practices and challenges: an empirical study. Inf Syst J 20(5):449–480. https://doi.org/10.1111/j.1365-2575.2007.00259.x

12. Behutiye W et al (2020) Management of quality requirements in agile and rapid software development: a systematic mapping study. Inf Softw Technol 123:106225. https://doi.org/10.1016/j. infsof.2019.106225

13. Coutinho JCS, Andrade WL, and Machado PDL (2019) Require- ments engineering and software testing in agile methodologies: a systematic mapping. In: ACM international conference proceed- ing series, pp 322–331. https://doi.org/10.1145/3350768.33525 84

14. Heck P, Zaidman A (2018) A systematic literature review on quality criteria for agile requirements specifications. Softw Qual J 26(1):127. https://doi.org/10.1007/s11219-016-9336-4

15. Golra FR, Beugnard A, Dagnat F, Guerin S, Guychard C (2016) Continuous requirements engineering using model federation. In: Proceedings - 2016 IEEE 24th international requirements engineering conference, RE 2016, pp 347–352. https://doi.org/ 10.1109/RE.2016.42

16. Forbrig P (2017) Does continuous requirements engineering need continuous software engineering? In: CEUR workshop proceed- ing, vol 1796

17. Kirikova M (2017) Continuous requirements engineering. In: International conference on computer systems and technologies CompSysTech'17 continuous, pp 1–10. https://doi.org/10.1145/ 3134302.3134304

18. Carkenord B (2017) DevOps may get BAs requirements manage- ment tools! 2017. https://rmcls.com/devops-may-get-bas-requi rements-management-tools/

19. Blueprint (2015) The relationship between DevOps and require- ments. https://www.blueprintsys.com/blog/the-relationship- between-devops-and-requirements

20. Gruhn V, Schåfer C (2015) BizDevOps: because DevOps is not the end of the story. Commun Comput Inf Sci 532:388–398. https://doi.org/10.1007/978-3-319-22689-7

21. Myrbakken H and Colomo-Palacios R (2017) DevSecOps: a mul- tivocal literature review. In: International conference on soft- ware process improvement and capability determination (SPICE 2017). https://doi.org/10.1007/978-3-319-67383-7_2

22. Gupta V, Kapur PK, Kumar D (2017) Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling. Inf Softw Technol 92:75–91. https://doi.org/10.1016/j.infsof.2017.07.010

23. Garousi V, Felderer M, Mäntylä MV (2019) Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf Softw Technol 106:101–121. https://doi.org/10.1016/j.infsof.2018.09.006

24. Petersen K, Vakkalanka S, Kuzniarz L (2015) Guidelines for conducting systematic mapping studies in software engineering: an update. Inf Softw Technol 64:1–18. https://doi.org/10.1016/j. infsof.2015.03.007

25. Duarte-Amaro RM, Pereira R, Mira da Silva M (2022) Capabili- ties and practices in DevOps: a multivocal literature review. IEEE Trans Softw Eng. https://doi.org/10.1109/TSE.2022.3166626

26. Mao R et al (2020) Preliminary findings about DevSecOps from grey literature. In: 2020 IEEE 20th international conference on software quality, reliability, and security, QRS 2020, pp 450–457. https://doi.org/10.1109/QRS51102.2020.00064

27. Lohrasbinasab I, Acharya PB, and Colomo-Palacios R (2020) BizDevOps: a multivocal literature review. In: 20th international conference on computational science and its applications – ICCSA 2020

28. Petersen K, Feldt R, Mujtaba S, and Mattsson M (2008) Sys- tematic mapping studies in software engineering. In: 12 th Inter- national conference on evaluation and assessment in software engineering, pp 484–489. https://doi.org/10.1016/j.pedneo.2016. 08.011

29. Curcio K, Navarro T, Malucelli A, Reinehr S (2018) Require- ments engineering: a systematic mapping study in agile software development. J Syst Softw 139:32–50. https://doi.org/10.1016/j. jss.2018.01.036

30. Inayat I, Salim SS, Marczak S, Daneva M, Shamshirband S (2015) A systematic literature review on agile requirements engineering practices and challenges. Comput Hum Behav 51:915–929. https://doi.org/10.1016/j.chb.2014.10.046

31. Haindl P and Plösch R (2020) Focus areas, themes, and objec- tives of non-functional requirements in DevOps: a systematic mapping study. In: 46th Euromicro conference on software engi- neering and advanced applications (SEAA)

32. Jin Z (2018) Requirements engineering methodologies. Environ Model Based Requir Eng Softw Intensive Syst. https://doi.org/ 10.1016/B978-0-12-801954-2.00002-9

33. Kasauli R, Knauss E, Horkoff J, Liebel G, de Oliveira Neto FG (2021) Requirements engineering challenges and practices in large-scale agile system development. J Syst Softw 172:110851. https://doi.org/10.1016/j.jss.2020.110851

34. Zowghi D, Coulin C (2005) Requirements elicitation: a survey of techniques, approaches, and tools. Engineering and managing software requirements. Springer, Berlin Heidelberg, pp 19–46

35. Klaus P, Chris R (2015) Requirements engineering fundamentals. A study guide for the certified professional for requirements engi- neering exam foundation level/IREB compliant. Rocky Nook. https://www.bbau.ac.in/dept/dit/TM/requirementsengi.pdf

36. Hemon A, Lyonnet B, Rowe F, Fitzgerald B (2020) From agile to DevOps: smart skills and collaborations. Inf Syst Front 22:927– 945. https://doi.org/10.1007/s10796-019-09905-1

37. Azad N, Hyrynsalmi S (2022) DevOps challenges organizations: through professional lens. Software business LNBIP, vol 463. Springer, Cham, pp 260–277

38. Wohlin C, Höst M, Henningsson K (2003) Empirical research methods in software engineering. Lect Notes Comput Sci 2765:7–23. https://doi.org/10.1007/978-3-540-45143-3_2

39. Ouhbi S, Idri A, Luis Fernández-Alemán J, Toval A (2015) Requirements engineering education: a systematic mapping study. Requir Eng 20:119–138. https://doi.org/10.1007/s00766-013-0192-5

40. Kitchenham B (2007) Guidelines for performing systematic lit- erature reviews in software engineering. In: Technical report, Ver. 2.3 EBSE

41. Kuhrmann M, Méndez Fernández D, Daneva M (2017) On the pragmatic design of literature studies in software engineering: an experience-based guideline. Empir Softw Eng 22:2852–2891. https://doi.org/10.1007/s10664-016-9492-y

42. Santos JL, Govaerts S, Verbert K, and Duval E (2012) Goal- oriented visualizations of activity tracking. p 143, https://doi. org/10.1145/2330601.2330639

43. Wendler R (2012) The maturity of maturity model research: a systematic mapping study. Inf Softw Technol 54(12):1317–1339. https://doi.org/10.1016/j.infsof.2012.07.007

44. Kitchenham BP, Barbara A, Budgen D (2015) Evidence-based software engineering and systematic reviews. Chapman and Hall/ CRC, New York

45. Ali N, Daneth H, Hong JE (2020) A hybrid DevOps process supporting software reuse: a pilot project. J Softw Evol Process 32(7):1–23. https://doi.org/10.1002/smr.2248

46. Altunel H, Say B (2022) Software product system model: a customer-value oriented, adaptable, DevOps-based product model. SN Comput Sci 3(1):1–11. https://doi.org/10.1007/s42979-021-00899-9

47. Atighetchi M, Simidchieva B, and Olejnik K (2019) Security requirements analysis - a vision for an automated toolchain. In: Proceedings - companion of the 19th IEEE international confer- ence on software quality, reliability and security, QRS-C 2019, pp 97–104. https://doi.org/10.1109/QRS-C.2019.00031

48. Ayerdi J et al. (2020) Towards a taxonomy for eliciting design- operation continuum requirements of cyber-physical systems. In: Proceedings of the IEEE international confer-

ence on require- ments engineering, pp 280–290. https://doi.org/10.1109/RE485 21.2020.00038

49. Bass L, Jeffery R, Wada H, Weber I, and Zhu L (2013) Eliciting operations requirements for applications. In: 2013 1st interna- tional workshop on release engineering, RELENG 2013 - pro- ceedings, pp 5–8. https://doi.org/10.1109/RELENG.2013.66076 88

50. Behutiye W, Rodríguez P, Oivo M, Aaramaa S, Partanen J, Abhervé A (2022) Towards optimal quality requirement documentation in agile software development: a multiple case study. J Syst Softw 183:111112. https://doi.org/10.1016/j.jss.2021.111112

51. Casale G et al (2020) RADON: rational decomposition and orchestration for serverless computing. Softw Intensive Cyber Phys Syst 35(1–2):77–87. https:// doi. org/ 10. 1007/s00450-019-00413-w

52. Casola V, De Benedictis A, Rak M, Villano U (2020) A novel Security-by-design methodology: modeling and assessing security by SLAs with a quantitative approach. J Syst Softw 163:110537. https://doi.org/10.1016/j.jss.2020.110537

53. Forbrig P (2018) BizDevOps and the role of S-BPM. S-BPM ONE 2018:1–8. https://doi.org/10.1145/3178248.3178250

54. Furfaro A, Gallo T, Garro A, Saccà D, and Tundis A (2016) ResDevOps: a software engineering framework for achieving long-lasting complex systems. In: Proceedings - 2016 IEEE 24th international requirements engineering conference, RE 2016, pp 246–255. https://doi.org/10.1109/RE.2016.15

55. Gupta RK, Venkatachalapathy M and Jeberla FK (2019) Chal- lenges in adopting continuous delivery and DevOps in a globally distributed product team: a case study of a healthcare organiza- tion. In: Proceedings - 2019 ACM/IEEE 14th international con- ference on global software engineering, ICGSE 2019, pp 30–34. https://doi.org/10.1109/ICGSE.2019.00020.

56. Haindl P, Plosch R, Komer C (2020) An operational constraint language to evaluate feature-dependent non-functional require- ments. In: Proceedings - 46th Euromicro conference on software engineering and advanced applications, SEAA 2020, pp 34–42. https://doi.org/10.1109/SEAA51224.2020.00017

57. Hosono S (2012) A DevOps framework to shorten delivery time for cloud applications. Int J Comput Sci Eng 7(4):329–344. https://doi.org/10.1504/IJCSE.2012.049753

58. Hugues J, Hristosov A, Hudak JJ, and Yankel J (2020) Twin- Ops - DevOps meets model-based engineering and digital twins for the engineering of CPS. In: Proceedings - 23rd ACM/IEEE international conference on model driven engineering languages and systems, MODELS-C 2020 - companion proceedings, p 668. https://doi.org/10.1145/3417990.3421446

59. Mäkinen S et al (2016) Improving the delivery cycle: a multiple- case study of the toolchains in Finnish software intensive enter- prises. Inf Softw Technol 80:175–194. https://doi.org/10.1016/j. infsof.2016.09.001

60. Mittal P, Mehta P (2020) Optimization of software development process by plugin integration with jira – a project management tool in devops. SSRN Electron J. https://doi.org/10.2139/ssrn.3564436

61. Morales JA, Yasar H, and Volkman A (2018) Implementing DevOps practices in highly regulated environments. In: Interna- tional workshop on secure software engineering in devops and agile development (XP '18 Companion), vol Part F1477. https:// doi.org/10.1145/3234152.3234188

62. Moreira CG and Nicolau De França BB (2019) Towards a healthier collaboration at the business-development interface. In: XXII Ibero-american conference on software engineering, CIbSE 2019, pp 86–99

63. Nakanishi T, Furusho H, Hisazumi K, and Fukuda A (2016) Dynamic SPL and derivative development with uncertainty management for DevOps. In: Proceedings - 2016 5th IIAI inter- national congress on advanced applied informatics, IIAI-AAI 2016, pp 244–249. https://doi.org/10.1109/IIAI-AAI.2016.240

64. Oliver I (2016) Experiences in the Development and usage of a privacy requirements framework. In: Proceedings - 2016 IEEE 24th International requirements engineering conference, RE 2016, pp 293–302. https://doi.org/10.1109/RE.2016.59

65. Olszewska M and Waldén M (2015) DevOps meets formal mod- elling in high-criticality complex systems. In: 1st International workshop on quality-aware DevOps, QUDOS 2015 - Proceed- ings, pp 7–12. https://doi.org/10.1145/2804371.2804373

66. Özcan-Top Ö, McCaffery F (2018) A hybrid assessment approach for medical device software development companies. J Softw Evol Process 30(7):1–15. https://doi.org/10.1002/smr.1929

67. Ravichandran A, Taylor K, Waterhouse P (2016) DevOps for digital leaders: reignite business with a modern DevOps-enabled software factory. CA Press, Berkeley

68. Rios E et al (2019) Service level agreement-based GDPR compli- ance and security assurance in (multi)Cloud-based systems. IET Softw 13(3):213–222. https://doi.org/10.1049/iet-sen.2018.5293

69. Rong G, Jin Z, Zhang H, Zhang Y, Ye W, and Shao D (2019) DevDocOps: towards automated documentation for DevOps. In: Proceedings - 2019 IEEE/ACM 41st international conference on software engineering: software engineering in practice, ICSE- SEIP 2019, pp 243–252. https://doi.org/10.1109/ICSE-SEIP.2019.00034

70. Rubasinghe I, Meedeniya D, Perera I (2021) SAT-analyser trace- ability management tool support for DevOps. J Inf Process Syst 17(5):972–988. https://doi.org/10.3745/JIPS.04.0225

71. Samarawickrama SS, Perera I (2017) Continuous Scrum: a framework to enhance Scrum with DevOps. In: 17th interna- tional conference on advances in ICT for emerging regions, ICTer 2017 – proceedings, pp 19–25. https://doi.org/10.1109/ICTER.2017.8257808

72. Sharma S, Kumar D (2021) Exploring story cards for evaluating requirement understanding in agile software development. J Inf Technol Manag 14:9–22

73. Sinkala K, Debbiche F, and Wrang M (2019) Accelerating soft- ware delivery in the context of requirements analysis and break- down for DevOps: a multiple-case study

74. Stray V, Moe NB, Aasheim A (2019) Dependency man-

agement in large-scale agile: a case study of DevOps teams. In: Proceed- ings of the 52nd Hawaii international conference on system sci- ences, pp 7007–7016. https://doi.org/10.24251/hicss.2019.840

75. Taryana A, Fadli A, Murdyantoro E, Rahmah S (2020) DevOps approach embraces forward and reverse engineering. Int J Appl Inf Technol. https://doi.org/10.25124/ijait.v4i02.2865

76. Theunissen T, Van Heesch U (2017) Specification in continuous software development. EuroPLoP. https://doi.org/10.1145/31477 04.3147709

77. Tüzün E, Tekinerdogan B, Macit Y, İnce K (2019) Adopting integrated application lifecycle management within a large-scale software company: an action research approach. J Syst Softw 149:63–82. https://doi.org/10.1016/j.jss.2018.11.021

78. Wahaballa A, Wahballa O, Abdellatief M, Xiong H, and Qin Z (2015) Toward unified DevOps model. In: Proceedings of the IEEE international conference on software engineering and service sciences, ICSESS, pp 211–214. https://doi.org/10.1109/ICSESS.2015.7339039

79. Wiecher C, Tendyra P, and Wolff C (2022) Scenario-based requirements engineering for complex smart city projects. In: 2022 IEEE European technology & engineering management summit

80. Yu L, Alégroth E, Chatzipetrou P, Gorschek T (2019) Utilising CI environment for efficient and effective testing of NFRs. Inf Softw Technol 117(May):2020. https://doi.org/10.1016/j.infsof. 2019.106199

81. Tort A (2016) KCycle: knowledge-based & agile software quality assurance. An approach for iterative and requirements- based quality assurance in DevOps. Requirements Engineering Magazine

82. de Best B (2022) DevOps plan – requirements from funnel to Scrum board. https://www.itpedia.nl/2017/07/13/devops-plan- requirements-from-funnel-to-scrum-board/

83. Curiel J, 2020 DevSecOps and GDPR: how to go from require- ments to deployment. https://techbeacon.com/security/devse cops-gdpr-how-go-requirements-deployment

84. Sardinha D (2021) Requirements life cycle management with Azure DevOps. https://www.modernanalyst.com/Resources/ Articles/tabid/115/ID/5624/Requirements-Life-Cycle-Manag ement-with-Azure-DevOps.aspx

85. Kum-Seun A (2020) BizDevOps starts with great requirements. https://www.softwarereviews.com/categories/application-lifec ycle-management/research/bizdevops-starts-with-great-requi rements

86. Macvittie D (2018) Watch your language: behavior driven development. https://devops.com/watch-your-language-behav ior-driven-development/

87. Decanio R (2017) Customer demands outpacing business require- ments. https://devops.com/customer-demands-outpacing-busin ess-needs/

88. Alibaba-Cloud-Community (2022) Requirement hierarchy - alibaba DevOps practice guide part 4. https://www.alibaba-clo ud.com/blog/requirement-hierarchy---alibaba-devops-practice- guide-part-4_598557

89. Santos N, Ferreira N, and Machado RJ (2021) Inputs to require- ments engineering in agile projects. How applying lean startup design, thinking and others, impact the task of modeling require- ments, Requirements Engineering Magazine

90. Devopsgroup (2013) The top ten DevOps operational require- ments. https://www.devopsgroup.com/blog/the-top-ten-devops- operational-requirements/

91. Ward C (2016) When DevOps and requirements gathering tech- niques collide. Tech target. Software development technologies. https://www.techtarget.com/searchsoftwarequality/tip/When- DevOps-and-requirements-gathering-techniques-collide

92. Shimel A (2015) Visualizing and defining requirements comes to DevOps. https://devops.com/visualizing-and-defining-requi rements-comes-to-devops/

93. Lange M, Potthoff S, Stephani TS (2019) Requirement man- agement with Microsoft Azure DevOps. https://www.campa na-schott.com/de/de/unternehmen/media-events/detail/requi rement-management-with-microsoft-azure-devops

94. Webster J, Watson R (2002) Analyzing the past to prepare for the future: writing a literature review". MIS Q 26(2):xiii–xxiii. https://doi.org/10.1016/j.freeradbiomed.2005.02.032

95. Becker C et al (2015) The Karlskrona manifesto for sustainability design. https://www.sustainabilitydesign.org/karlskrona-manif esto/

96. Duboc L et al (2020) Requirements engineering for sustainabil- ity: an awareness framework for designing software systems for a better tomorrow. Requir Eng 25(4):469–492. https://doi.org/10. 1007/s00766-020-00336-y

97. Olsson T, Sentilles S, Papatheocharous E (2022) A system- atic literature review of empirical research on quality require- ments. Requir Eng 27:249–271. https:// doi. org/ 10. 1007/ s00766-022-00373-9

98. Ampatzoglou A, Bibi S, Avgeriou P, Verbeek M, Chatzigeor- giou A (2019) Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. Inf Softw Technol 106:201–230. https://doi.org/10.1016/j.infsof.2018.10.006

99. Zhang H, Babar MA, Tell P (2011) Identifying relevant studies in software engineering. Inf Softw Technol 53(6):625–637. https:// doi.org/10.1016/J.INFSOF.2010.12.010

100. Moreira A and Schneider K, (2022) Editorial 1. In: Require- ments engineering conference, pp 403–404. https://doi.org/10. 1007/s00766-022-00392-6.