ISSN 2303-4521
Periodicals of Engineering and Natural Sciences *Original Research*
Vol. 11, No. 2, April 2023, pp.207-220

DevOps Ontology - An ontology to support the understanding of DevOps in the academy and the software industry

Jonathan A. Guerrero[1], César Pardo[1], Carlos E. Orozco[1]
[1] University of Cauca, GTI Research Group. Calle 5 No. 4-70, Popayán, Colombia

ABSTRACT

Currently, the degree of knowledge about what DevOps really means and what it entails is still limited. This can result in an informal and even incorrect implementation in many cases. Although several proposals related to DevOps adoption can be found, confusion is not uncommon and terminology conflict between the proposals is still evident. This article proposes DevOps Ontology, a semi-formal ontology that proposes a generic, consistent, and clear language to enable the dissemination of information related to implementing DevOps in software development. The ontology presented in this article facilitates the understanding of DevOps by identifying the relationships between software process elements and the agile principles/values that may be related to them. The DevOps Ontology has been defined considering the following aspects: the REFSENO formalism that uses the representation in UML was used and the language OWL language using Prótegé and HermiT Reasoner to evaluate the consistency of its structure. Likewise, it was satisfactorily evaluated in three application cases: a theoretical validation; instantiation of the continuous integration and deployment practices proposed by the company GitLab. Furthermore, a mobile app was created to retrieve information from the DevOps Ontology using the SPARQL protocol and RDF language. The app also evaluated the Ontology's proficiency in responding to knowledge-based questions using SPARQL. The results showed that DevOps Ontology is consistent, complete, and concise, i.e.: to say: the consistency could be observed in the ability to be able to infer knowledge from the ontology, ensuring that the ontology is complete by checking for any incompleteness and verifying that all necessary definitions and inferences are well-established. Additionally, the ontology was assessed for conciseness to ensure that it doesn't contain redundant or unnecessary definitions. Furthermore, it has the potential for improvement by incorporating new concepts and relationships as needed. The newly suggested ontology creates a set of terms that provide a systematic and structured approach to organizing the existing knowledge in the field. This helps to minimize the confusion, inconsistency, and heterogeneity of the terminologies and concepts in the area of interest.

Keywords: Development and Operations (DevOps); Resource Description Framework (RDF); Protégé; SPARQL; OWL

Corresponding Author:

César Pardo
PhD. MSc. Researcher. Full Professor
University of Cauca
Carrera 2, Calle 15n. Office 444, Colombia
E-mail: cpardo@unicauca.edu.co

1. Introduction

The software development industry has been in constant flux since its inception, driven largely by the need to improve the quality of software products and services through constant improvement of the processes and practices used to obtain them [1]–[3]. Day by day the needs of clients, the guidelines established by models and the industry standards in software development present higher and higher levels of demand and complexity. Several solutions to these demands and complexities have emerged. The best known and that are used currently can be classified into agile and traditional approaches. Among the traditional are CMMI [4], RUP [5], Microsoft

Solution Framework [6], ISO 9001 [7], and ISO 29110 [8], while agile approaches include Scrum [9], Lean Software [10], TDD [11], and XP [12]. Others that might be mentioned here are the category of hybrid approaches that combine traditional and agile approach practices, such as Scrum & XP [13], Scrumban [14], and Scrum & CMMI [15]. However, regardless of how frameworks are categorized, these involve only elements related to the development (Dev), of a software product, leaving aside aspects related to operations (Ops), which involve the provision of infrastructure, continuous monitoring, collaboration, experimentation, deployment, and maintenance of products [16]–[18]. Elsewhere, from the area of operations or infrastructure, frameworks such as ITIL [19] and COBIT [20], and general standards such as ISO/IEC 20000 [21] have been developed, this has led to significant advancements in the process elements such as the steps, activities and practices related to the management of information technology services and the management of processes to ensure the continued existence and success of software products. However, there is lack of clarity on how these advances from the operations or infrastructure area can be integrated naturally with dynamics and objectives around the software development area.

Due to advances in the digitization and automation of practices in software development - configuration management, integration, deployment, and continuous monitoring - there is currently a need in the industry for joint work between the areas of software development and infrastructure, since only then will it be possible to increase productivity and achieve the high degree of competitiveness the industry requires [22]–[24]. Nevertheless, the current frameworks on either side are usually complementary. For example, the software development area addresses practices that allow it to adapt quickly to changes, rapid delivery of new functionalities in productive environments, experimentation in how to face complex requirements, rotation of roles/responsibilities in the team during the development of a project and autonomy to take on challenges. The infrastructure area meanwhile seeks to standardize and stabilize its requirements, processes, tools, and roles [25], [26]. Due to this type of situation and other similar situations, a wall of confusion has been generated those divides professionals in each of these two areas, since historically each area has sought to improve its internal processes without considering other areas involved [27].

Agile approaches have recently introduced one of the most recently mentioned concepts in the software industry, it is the concept of: DevOps. Dating back to 2009 [28]–[30], DevOps can be defined as a set of practices that tries to bridge the gap between develop and operations covering all the elements that help in the implementation of high-quality software [31], [32]. DevOps emphasizes the utilization of human expertise, technology, and streamlined processes to foster a culture of innovation and ongoing collaboration throughout the software development lifecycle. DevOps practices are mainly focused on continuous deployment, monitoring, testing and maintenance [16], [18], [33]–[35], all in an accelerated, reliable way and without suffering any deterioration in quality [36]. These practices can be reinforced by different tools that can automate their execution, such as code repositories for integration, management tools, and software testing frameworks [37]–[39].

According to [40], DevOps automation is reinforced by specific design patterns that enhance the seamless delivery of software applications on cloud-based platforms. Additionally, DevOps is known to enable the prompt delivery of builds, features, and bug fixes, resembling an industrial assembly line [35]. However, adopt DevOps is not trivial [30], to enable a clear and correct adoption, it is important that the first step is to organize and clearly understand the concepts, practices, tools, definitions, and the relationships between them, as well as understanding the benefits and challenges for their adoption to be effective in the development of systems [41], [42]. In this light, it is necessary to reduce the ambiguity and heterogeneity of the knowledge developed around DevOps by the scientific and academic community. To do so, this article proposes DevOps Ontology, an ontology that aims to reduce the identified gap. DevOps Ontology generically describes the elements that must exist in any DevOps adoption process. As such, the ontology can be used regardless of the practices and the approach being used. DevOps Ontology thus organizes the knowledge around the implementation and adoption of DevOps in the software industry using a formal mechanism, thereby, clarifying the meaning of concepts and terminology within the domain to facilitate knowledge sharing and collaboration among all parties involved.

Moreover, DevOps Ontology has the potential of facilitate the evaluation/assessment of DevOps compliance in software organizations by identifying the connections between the components related to software development process, agile principles, and software measurement. Accordingly, DevOps Ontology has been enhanced and adjusted to account for the unique attributes of software processes within each organization, with the aim of facilitating the integration of DevOps practices. The integration of ideas and concepts from different ontological solutions was achieved integrating the elements of different solutions such as the Ontology of Process-reference Models (PrMO) [43].

In addition to this introduction, the work is organized as follows. Section 2 analyzes the current state of the art, while Section 3 introduces DevOps Ontology, a collection of interrelated concepts intended to provide a reference for developing models, standards, and practices for implementing DevOps in the software development industry. In Section 4, different application cases are described, along with how they address a set of competency questions. Finally, Section 5 concludes the work and outlines areas for future research.

2. Related work

After conducting a comprehensive and structured systematic mapping of the literature [37], a detailed analysis was made of the studies and initiatives that address the terminology, definitions, concepts and relationships between concepts related to DevOps through (i) exploratory studies that seek to identify the terminology and concepts present in the literature in trying to adopt DevOps; (ii) the definition of taxonomies that establish the elements and relationships necessary to build a conceptual framework in order to validate the knowledge about DevOps; and (iii) ontologies developed for specific aspects and practices such as continuous integration, build, deployment and assessment of DevOps. The results obtained after reviewing the literature are presented below:

Exploratory studies in the use of terminology related to DevOps: The DevOps term arise in 2009 [44]; over the years, the scientific community has dedicated its efforts to getting to know the definition, objectives, scope, and elements associated with DevOps. To achieve this, exploratory studies were carried out that sought to understand "what" is DevOps both in the literature and in the business environment, for example in [45] a study was conducted that presents the most common fallacies and mistakes that companies in the software development industry make in seeking to adopt DevOps. Moreover, systematic reviews of the literature were carried out to identify the concepts, practices, tools, and necessary elements for the definition of a body of knowledge in DevOps [41], [46]. In [47] and [37] exploratory studies have similarly been executed to observe and analyze the trends and perspectives around DevOps as a concept. In addition, [48] presents a process for the adoption of DevOps. Finally, in [27] and [49] exploratory studies are carried out to identify the concepts and challenges that companies must confront when adopting DevOps.

Taxonomies to support DevOps: Related studies that propose the use of taxonomies and their application were identified through empirical case studies or by means of validation of external evaluators in software development companies: (i) in [50] and [51] taxonomies were proposed that describe the relationships that must be consider to face the functional challenges present in the adoption of DevOps; (ii) in [52] a taxonomy was presented to identify the necessary structure that allows facing the challenges related to SecDevOps; (iii) the taxonomy proposed in [53] investigates the consequences resulting from applying DevOps during the software delivery process; and in (iv) the definition of an empirical taxonomy was carried out based on the most relevant elements for the adoption of DevOps that were identified by 11 DevOps experts [54].

Ontologies to support DevOps: The exploratory study identified solutions proposed to support specific DevOps elements using ontologies. For example, (i) in [55] a software configuration management process is proposed; in (ii) the OSDAS [56] ontology was proposed that carries out the unification of multiple agile approaches including DevOps; and in (iii) present a generic ontology to assess the maturity level of DevOps [57]. However, the term DevOps is mentioned generically or at a very high level, evidencing that although there are solutions to support practices and specific aspects related to DevOps, it was not possible to observe a solution that contributes a conceptual framework that involves the values, principles, practices, roles, processes in an integral way.

In this sense, multiple proposals were identified as having been developed to support specific domains and processes during the software development process, following agile and/or traditional approaches through the characterization of terms, concepts, and relationships between clear definitions. Parallel efforts have been made to resolve differences and conflicts found in the terminology in discussing DevOps, by means of exploratory studies, systematic mapping, systematic reviews of the literature and the definition of formal processes to support DevOps. Meanwhile, studies have been carried out to define ontologies and taxonomies focused on specific aspects related to the practices, values and principles proposed by DevOps individually, based on activities such as continuous integration, build, deployment, and software configuration management. In addition, aspects related to the identification of challenges associated with DevOps and their assessment using ontologies have been studied. As a result, it is possible to observe that significant advances have been made that seek to resolve the terminological and conceptual differences present when seeking to establish a unified DevOps framework. However, it can also be seen that there is no consensus and homogeneity in the concepts used, the relationships raised, and the definitions adopted by researchers and the industry, this makes it challenging to comprehend the essential elements needed for a successful DevOps adoption process in software development companies. For this reason, initiatives have been proposed that seek to mitigate/solve by defining processes that seek to standardize knowledge around this issue. For example, the IEEE 2675-2021 standard [58] defines DevOps based on different facets such as mission as first goal, customer as focus, left shift approach, and systemic thinking. Nevertheless, the IEEE 2675-2021 standard lacks a comprehensive reference model that can standardize the terminologies and concepts related to DevOps.

3. Methodology

Several methodological solutions exist to aid in the creation of an ontology, and among them is knowledge management that relies on ontologies [59]; (ii) Methontology [60]; (iii) other approaches that can assist in defining an ontology include a translation method for developing portable and flexible ontology solutions [61]; (iv) ontology based on first order logic [62]; (v) ontologies characterization based in the software lifecycle context using REFSENO [63], among others. To create DevOps Ontology, the authors chose to use the REFSENO methodology. This approach is designed to create ontologies in the field of software engineering and provides constructors to define concepts, attributes, and relationships. It is also more user-friendly than other methods, reducing the complex and unintuitive representations found in other solutions that rely on predicate logic. REFSENO is an adaptation of Methontology, which is used to build ontologies at the knowledge level and from scratch, and it provides techniques to analyze the consistency of the ontology and its instances at the implementation level. It allows, too, the representation of knowledge at two levels of abstraction: (i) the knowledge level, which describes the "what to represent" - and which is independent of implementation [63] and (ii) the symbols level, which defines the "how to represent". REFSENO proposes three levels of knowledge: (i) the epistemological level, which describes epistemic primitives such as concepts, attributes, and relationships; (ii) the conceptual level, that makes it possible to describe the standard vocabulary; and (iii) the linguistic level, that proposes the mechanisms to define concrete instances of the constructs defined in previous levels [64]. To define DevOps Ontology, the REFSENO approach was used, which involves three stages: construction, evolution, and validation. Additionally, descriptive logic (DL) was followed, as it is more expressive than propositional logic [65]. The construction stage involved defining the epistemological level, which described the concepts, attributes, and relationships, the conceptual level, which provided a standard vocabulary, and the linguistic level, which defined mechanisms to represent concrete instances of the constructs. The evolution stage involved updating and refining the ontology over time based on changes in the DevOps domain, while the validation stage involved ensuring the consistency and correctness of the ontology and its instances at the implementation level.

Integration: ontological meaning. In the domain of ontologies, the term "integration" group several different meanings, as pointed out in [66], it can refer to: (i) identifying the correspondence between multiple ontological solutions (pairing); (ii) the outcome of identifying common elements between ontologies (alignment); (iii) mapping, which involves linking the elements of one ontology to at least one entity of another one, in a directed or oriented manner; (iv) merge, which involve create a new ontology by combining two existing ontologies that may overlap; and (v) include one ontology within another ontology, along with the statements that establish their relationship [67].

4. Results

The systematic literature review about the implementation of DevOps in software development companies [37], revealed two primary goals for creating the ontology: (i) identifying the terminology associated with DevOps, and (ii) standardizing that terminology. DevOps Ontology has been designed to fulfill these objectives by providing clear definitions and relationships between DevOps concepts, making it a valuable resource for developing assessment models and processes. As per the discussion introduced by [68], the classification of ontologies in software engineering can be done by grouping them based on their domain or software artifacts. DevOps Ontology belongs to the domain category as it structures the knowledge pertaining to essential DevOps concepts and their relationships within the software development industry. This section provides a general overview of DevOps Ontology, covering its purpose, glossary of concepts and relationships, graphical representation, and integration with other ontologies.

4.1. Purpose of the ontology

The elements presented in DevOps Ontology can be implemented in the following contexts: (i) *Academic*, because it can be used by those interested in learning or researching about DevOps adoption strategies in the software development industry and (ii) *Industrial*, since software development companies will have an instrument to instantiate their adoption processes, allowing them to determine terms and relationships present in the context of DevOps. In the same way, it works as the basis for the design of an instrument to allow the assessment and/or evaluation of the DevOps adoption process through automation with other types of solutions such as machine learning. DevOps Ontology provides the elements that allow answering the competency questions (CQ) described in Table 1.

Table 1. Competency Questions. **Source:** self-made

Id	Question
CQ1	What is the degree of relationship between practices and principles?
CQ2	What are the key components that constitute a practice?
CQ3	What is the relation between the generated products and the DevOps practices, as indicated by each dimension?
CQ4	How are the products generated related to the DevOps principles?
CQ5	How are the tasks associated with the DevOps roles?
CQ6	How are the products related to the DevOps roles?

4.2. Integration of DevOps ontology with other ontologies

As DevOps Ontology serves as a foundation for developing instruments to evaluate the DevOps adoption process in software development companies, it is crucial to incorporate a clear terminology to support the software measurement process. To accomplish this, the Ontology of Process-reference Models (PrMO) [43] was introduced. PrMO establishes the fundamental elements in different approaches, particularly in traditional and agile approaches (Scrum, CMMI, SWEBOK, ITIL, ISO 9001, ISO 27001, ISO 29110, and ISO 31000). Furthermore, DevOps Ontology was integrated with several concepts from the Software Measurement Ontology (SMO) subontology introduced in [69]. SMO clarifies and establishes the essential elements in defining software measures and terminology related to software measurement actions. Figure 1 presents a visual representation of the terms and relations associated to DevOps Ontology using the Unified Modeling Language (UML) representation, which can be accessed at: https://bit.ly/2YQM8jW.

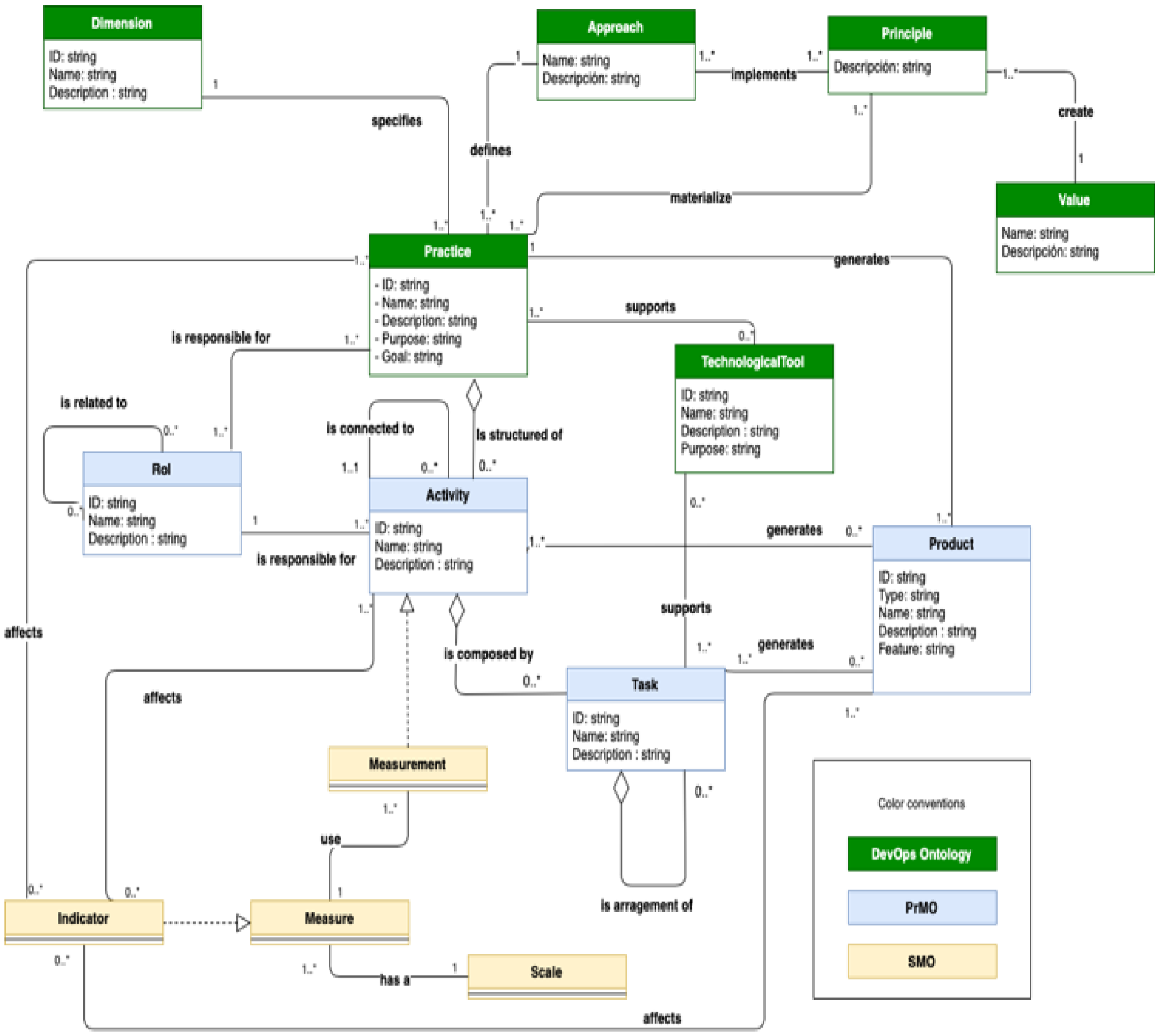

Figure 1. DevOps Ontology Representation in UML. Source: self-made

4.3. DevOps ontology concepts

Table 2 displays all the definitions and concepts incorporated into the ontology. The table is organized into four columns: the first column lists the concept utilized in the ontology, the second column indicates the super concept that it belongs to, the third column explains the term's definition within the ontology, and the last column specifies the reference where the term was obtained or adjusted. For more detailed information on the relationships between the concepts in Table 2, please refer to https://bit.ly/3fL4fOC. The last column of Table 2 can be categorized into three aspects: (i) *"taken from [source]"* for terms proposed by other authors and have a high importance degree in the ontology, without any modification or adaptation to the term; (ii) *"defined from [source]"* for terms that were used to define a new concept without altering the existing one; and (iii) *"quoted from [source]"* for terms that were cited from a source without modification.

Table 2. Concepts in DevOps ontology. Source: self-made

Term	Super concept	Definition	Source
Activity	Concept	It consists of a collection of tasks or operations that are used to create and sustain products aligned with the process goals. The activity encompasses the processes, policies, standards and goals for developing and modifying a set of work products.	Taken from [43]

Term	Super concept	Definition	Source
Approach	Concept	It establishes an overview of how the development process of a software product should be approached. In the case of software development, you can find traditional approaches, agile approaches, and hybrid approaches.	Autors
Dimension	Concept	Business areas in which practices and processes are grouped.	Defined from [70]
Indicator	Measure	A derived measure is one that is based on other measures that employ an analysis model as the measurement technique.	Taken from [69]
Measure	Concept	The measurement approach that has been defined and a measurement scale.	Taken from [69]
Measurement	Concept	A set of procedures that strive to ascertain the measurement outcome for a particular attribute of an entity using different measurement techniques.	Taken from [69]
Value	Concept	Qualities and / or fundamentals that must be present in the members of a work team. collaboration, automation, measurement, and monitoring.	Autors
Practice	Concept	The practices embody the core ideas and values that characterize DevOps in day-to-day operations and development activities.	Taken from [25]
Principle	Concept	Base of ideals, foundations or rules from which ideologies arise.	Defined from [25], [70]
Product	Concept	The collection of items that are created, delivered, and sustained in a project is referred to as a product. A product can be considered as an input or output. Typically, products are concrete objects that are utilized, created, or modified by tasks.	Taken from [43]
Role	Concept	A collection or cluster of obligations, tasks, and competencies that are necessary to execute a particular activity is known as a role.	Taken from [43]
Scale	Concept	A set of values with clearly defined properties.	Taken from [69]
Task	Concept	A process component that outlines the responsibilities carried out by roles is known as a task. In general, a task is related to an input or output product.	Taken from [43]
Technological Tool	Resource	Tools are a crucial component of the DevOps approach as they are closely associated with various DevOps practices like automation, monitoring, and built-in configuration, and aid in implementing these practices.	Taken from [26]

4.4. Discussion of some concepts and their relationships

This section will provide a more detailed analysis and definition of certain terms used in DevOps Ontology, including but not limited to principles, practices, and dimensions. The term value corresponds to the foundations on which the essence of DevOps originates. For example, the most common elements found in the literature suggest that collaboration [70] should exist as a value around different software development areas, reinforced by the exchange of information, the expansion of skills and the change of responsibilities between work teams; secondly, we find automation [70], a crucial element to allow different types of tasks to be provided and deployed repetitively and quickly; in third place, measurement [70], which is important throughout the process since by incorporating metrics it is easier to increase efficiency in product development; and finally monitoring [70], which helps determine the appropriate allocation of resources in order to detect, report and correct problems that may occur during or after any kind of system update. A practice allows the principles to be materialized by carrying out specific actions, the practices of integration, construction, deployment, configuration, and continuous monitoring being the most visible in most of the studies found [37]. A dimension refers to all those business areas in which the entire set of practices and their related elements can be classified, among them processes, people, culture, and technology [26].

The following link (https://bit.ly/39KvTHp) presents a figure, which shows an extract of the VOWL plugin for Protégé (VOWLProtégé) [71] that implements the visual notation for OWL ontologies. This plugin provides graphic illustrations of elements of an ontology defined with OWL [72] by means of the representation of directed graphs, where the relationships and characteristics of the concepts proposed in the ontology are shown. Also, it is possible to observe how the dimensions specify practices, which are the responsibility of one or more roles. They can be supported by technological tools and are structured through activities, which in turn are made up of tasks. Thus, the practices materialize the principles.

4.5. Assessment of DevOps ontology

According to [73], three main aspects should be considered to evaluate an ontology. These aspects are conciseness, completeness and consistency. The consistency can be achieved if the knowledge from the ontology can be deduced without ambiguity. The completeness can be achieved if the ontology does not contain any missing information, meaning that all the definitions are well established, or can be inferred without confusion. Furthermore, the ontology can be considered concise if it is free of redundancy and unnecessary definitions.

The summarized ontology, presented below, was applied in the OWL ontology language [56] complemented with Protégé [61] and the HermiT Reasoner [62] to assess the consistency of the ontology structure (the ontology can be consulted in detail at: https://stanford.io/38TisW7). The consistency analysis concluded that DevOps Ontology has no inconsistencies in its structure. Afterwards, DevOps Ontology was applied in three successful implementation cases: (i) a theoretical validation in which it was evidenced how the concepts and relationships associated with the ontology are used to describe the particularities that could be found in the description of a DevOps adoption process that allows that allowed to determine the completeness of the ontology; (ii) the instantiation of the practices of integration and continuous deployment proposed by the GitLab company that allowed to identify that the ontology is concise and fits these specific DevOps practices; and (iii) the implementation of a mobile app that queries DevOps Ontology using the Resource Description Framework (RDF) and the (SPARQL) query protocol to know the relationship between practices/principles of DevOps. Additionally, SPARQL allows to answer each of the competency questions, the summary of the answers obtained can be seen in detail in Table 3.

Table 3. Competency questions expressed in SPARQL. Source: self-made

Competency question	SPARQL	Result
CQ1: What is the degree of relationship between practices and principles?	SELECT ?element1 ?relation WHERE {?relation rdfs:domain ?element1 . ?relation rdfs:range devops:Principle . FILTER (?element1 IN (devops:Practice)) }	Practice materialize Principle
CQ2: What are the key components that constitute a practice?	SELECT ?PracticeElement WHERE {?r rdfs:domain devops:Practice. ?rrdfs:range ?PracticeElement }	Policy -Rol – Principle - Product – Indicator - TechnologicalTool – Activity - Dimension

CQ3. What is the relation between the generated products and the DevOps practices, as indicated by each dimension?	SELECT ?source ?rel ?target ?rel1
	WHERE {
	?rel rdfs:range ?target .
	?rel rdfs:domain ?source .
	?rel1 rdfs:range devops:Product .
	?rel1 rdfs:domain ?target .
	}
	FILTER(?e0 IN (devops:Dimension))}
		Dimension specifies Practice
		Practice generates Product

CQ4. How are the products generated related to the DevOps principles?	SELECT ?entity1 ?relation1 ?entity2 ?relation2 WHERE {?relation1 rdfs:domain ?entity1 . ?relation1 rdfs:range ?entity2.}	Product isGenerate By Practice

Competency question	SPARQL	Result
		Practice materialize Principle
CQ5. How are the tasks associated with the DevOps roles?	SELECT ?role ?relation1 ?task ?relation2 WHERE { ?relation1 rdfs:domain ?role . ?relation1 rdfs:range ?task. ?relation2 rdfs:domain ?task. ?relation2 rdfs:range devops:Task. FILTER(?role IN (devops:Role)) }	Rol isResponsibleFor Activity Activity isComposedOf Task
CQ6. How are the products related to the DevOps roles?	SELECT ?entity1 ?relationship WHERE { ?relationship rdfs:domain ?entity1 . ?relationship rdfs:range devops:Principle . FILTER (?entity1 IN (devops:Practice)) }	Practice materialize Principle

4.5.1. Theoretical evaluation

The first implementation case involves the narration of a theoretical example of application of some of the concepts included in DevOps Ontology. The concepts used appear in italics. The model of implementation of the practices of integration and continuous deployment of the Gitlab company [74] brings to life the *value* of DevOps called automation, which determines the *principles* of early and continuous delivery, automation of processes, and deployment in the shortest possible time. These *principles* are brought to life with practices that require continuous execution, they are: integration and deployment, defined by the company's process pipeline, which are categorized in the *dimension* of processes and technology. From the practice of continuous integration, the activities of coding, review, approval, automation of the compilation of the code, automation of tests and automation of deployment are unfolded. These last two activities are also part of the continuous deployment practice and generate as a product the respective deployments for review and deployment for production. Regarding the *activity* of coding, the following key *tasks* are described that are carried out directly through the Gitlab *technological tool* and that are the responsibility of whoever assumes the *role* of developer: creation of new branches, sending improvements to the code, submitting code changes and merge operations.

4.5.2. Instantiating the practices of integration and continuous deployment proposed by GitLab

In the second implementation case, an instantiation of the elements proposed by GitLab for the practices of integration and continuous deployment was carried out. For this instantiation, the following DevOps Ontology concepts were used: value, principle, practice, dimension, activity, technological tool, role, task, and product. The following link https://bit.ly/3A6kj3X shows the instance created by DevOps Ontology, where the practices of integration and continuous deployment are created from the DevOps value of automation and involve the dimensions of technology and processes. Additionally, from the practice of continuous integration, three main activities are structured: coding, build automation and test automation. Within the activity of coding, which is the responsibility of the developers, the tasks of creating new branches, sending code corrections, sending code modifications, and merging processes in the repositories of the project are detailed. On the practice of continuous deployment side, more exhaustive test automation and deployment automation activities are structured, generating as output functional versions deployed in environments for review or production, depending on the needs of the organization. It can be seen in this instantiation that DevOps Ontology allows to represent the practices of integration and continuous deployment proposed by Gitlab (due limitations of space, the instantiation detail can be accessed through https://bit.ly/39Ofq56).

4.5.3. Supporting the visualization of concepts and relationships in DevOps ontology

The third case of implementation involves creating a mobile app that enables users to view the concepts and relationships specified in the ontology, which aid in addressing the various competency queries. Screenshots of the mobile app, including the navigation menu for accessing the concepts and relationships, the homepage, and detailed concept descriptions, are available at: https://bit.ly/31RBYxv. The information displayed on the app is generated using SPARQL queries executed on the DevOps Ontology.

4.5.4. Answering the competence questions using SPARQL

To assess whether the ontology achieved its intended purpose, all the competency question were translated to the SPARQL query structure. The outcome revealed that DevOps Ontology was able to address all the competency questions that were initially formulated during its development. The following link https://bit.ly/3rWR7Io displays the SPARQL queries for each competency question and their corresponding results. From the defined competence questions, the following ontology answers were found: (i) DevOps practices materialize principles; (ii) the most important elements related to a DevOps practice are the roles, principles, indicators, technological tools and dimensions; (iii) the dimensions specify practices which in turn generate products; (iv) a product can be generated by a practice, the practices are materializing principles; (v) roles are responsible for activities and activities are made up of tasks; and (vi) the products are generated from the execution of practices, activities or tasks that are the responsibility of a particular role.

4.5.5. Answering the competence questions using SPARQL

To guarantee a definition directly adapted to DevOps, a comparative analysis was carried out that allowed knowing the scope, objective and application of the ontologies and taxonomies identified in the literature with DevOps Ontology (see Table 4). To carry out this task, the following evaluation criteria were applied: (CR1) the ontology can be applied to any DevOps process present within a company; (CR2) the concepts include the principles and values defined by DevOps; (CR3) it can be instantiated and integrated into existing ontologies; (CR4) the ontology is described applying a formal language structure; and (CR5) the ontology can be access through a public repository.

Table 4. Ontologies and taxonomies comparison. Source: self-made

Ontology	Reference	CR1	CR2	CR3	CR4	CR5
Composable DevOps	[75]	X		X	X	
Taxonomy	Reference	CR1	CR2	CR3	CR4	CR5
Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process	[76]	X			X	
Prioritization Based Taxonomy of DevOps Challenges Using Fuzzy AHP Analysis	[77]	X			X	
Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE	[78]	X			X	
A taxonomy of software delivery performance profiles: Investigating the effects of DevOps practices	[79]	X	X		X	
An Empirical Taxonomy of DevOps in Practice	[80]		X	X		
DevOps Ontology	Own	X	X	X	X	X

According to the comparison, the ontologies are focused on aspects related to traditional and/or agile software development approaches and the evaluation of specific DevOps practices. In addition, some of the ontologies do not have public access to ensure their availability and allow feedback and assessment processes to be carried out.

5. Conclusions

DevOps Ontology is a semi-formal ontology designed to support the adoption and use of DevOps in the software industry. This ontology provides a standard terminology that organizes existing knowledge from literature concisely, reducing ambiguity and inconsistencies in the field. To create DevOps Ontology, the REFSENO formalism was used, which employs UML representation. Besides, the OWL language was used to make it more accessible to both academics and industry professionals.

Moreover, the aim of DevOps Ontology is to offer a preliminary solution to clarify the terminology around DevOps; to achieve this, an MSL was executed. DevOps Ontology was evaluated through three practical implementation cases: (i) a theoretical validation; (ii) the representation of the integration and continuous deployment practices proposed by GitLab according to the terms presented in the ontology; and (iii) the development of a mobile application that visualizes the relationship between principles and practices using

SPARQL queries. Finally, the competency questions were answered according to the results obtained after executing the queries presented in Table 3.

All the outcomes derived from creating DevOps Ontology will be utilized to perform specific tasks in the future, with the following order of priority: (i) carry out case studies in software companies that allow the ontology to be applied in operating environments; (ii) use the ontology as a conceptual basis to defining a reference model to support the adoption and evaluation of DevOps in the software industry context; and (iii) use inference mechanisms to automatize the assessment of DevOps as a support for expert consultants, allowing the comparison of results of manual assessments.

Future work related to the research work include: (i) extend the evaluation of the ontology through a comparative analysis of the model with the IEEE 2675-2021 standard; (ii) instantiate the ontology with other models, processes and workflows to ensure that it fits the different existing solutions around DevOps; and (iii) develop a tool that allows the ontology to be used by integrating it with existing processes through a conceptual and terminological verification process using natural language.

Declaration of competing interest

The authors declare that they have no known financial or non-financial competing interests in any material discussed in this paper.

Funding information

No funding was received from any financial organization to conduct this research.

Acknowledgements

We thank the anonymous reviewers. Likewise, PhD. César Pardo acknowledges the contribution of the University of Cauca (Popayán-Colombia), where he works as a full professor.

References

[1] H. Conradi and A. Fuggetta, “Improving software process improvement,” *IEEE Softw*, vol. 19, no. 4, pp. 92–99, Jul. 2002.

[2] C. E. Mokhlis, A. Elmortada, M. Sbihi, and K. Mokhlis, “The impact of ISO 9001 quality management on organizational learning and innovation: Proposal for a conceptual framework,” *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 2, pp. 944–951, 2019.

[3] Y. Andalib and C. E. Mokhlis, “The contribution of the quality certification process to the improvement of human resources management practices,” *Periodicals of Engineering and Natural Sciences*, vol. 8, no. 3, pp. 1880–1887, 2020.

[4] CMMI Institute, “CMMI V2.0 model,” *Second edition*, 2020. https://cmmiinstitute.com/cmmi (accessed Mar. 21, 2023).

[5] Davor Gornik, “IBM Rational Unified Process: Best Practices for Software Development Teams. TP026B, Rev 11/01,” Somers, New York, 2020.

[6] G. Lory, D. Campbell, A. Robin, G. Simmons, and P. Rytkonen, “Microsoft Solutions Framework v3 Overview,” 2003.

[7] ISO, “ISO 9001:2015 Quality Management Systems,” Geneva, Switzerland, 2020.

[8] ISO, “ISO/IEC TR 29110-1:2016 Systems and software engineering — Lifecycle profiles for Very Small Entities (VSEs),” Geneva, Switzerland, 2021.

[9] K. Schwaber and J. Sutherland, “The scrum guide the definitive guide to scrum: The rules of the game,” USA, 2020.

[10] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[11] K. Beck, *Test Driven Development: By Example*, 1st ed. Boston: Addison-Wesley Professional, 2002.

[12] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Boston: Addison-Wesley Professional, 2004.

[13] H. Kniberg, *Scrum and XP from the Trenches*, 2nd ed. USA: InfoQ, 2015.

[14] C. Ladas, *Scrumban-essays on kanban systems for lean software development*, 1st ed. Seattle: Modus Cooperandi Press, 2009.

[15] J. Sutherland, C. R. Jakobsen, and K. Johnson, “Scrum and CMMI level 5: The magic potion for code warriors,” in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, 2008, pp. 466–466.

[16] E. Diel, S. Marczak, and D. S. Cruzes, “Communication Challenges and Strategies in Distributed DevOps,” in *Proceedings of the 11th International Conference on Global Software Engineering (ICGSE)*, 2016, pp. 24–28.

[17] M. Rajkumar, A. K. Pole, V. S. Adige, and P. Mahanta, “DevOps culture and its impact on cloud delivery and software development,” in *Proceedings of the International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring)*, 2016, pp. 1–6.

[18] M. de Bayser, L. G. Azevedo, and R. Cerqueira, “ResearchOps: The case for DevOps in scientific applications,” in *Proceedings of the International Symposium on Integrated Network Management (IM)*, 2015, pp. 1398–1404.

[19] A. Hochstein, R. Zarnekow, and W. Brenner, “ITIL as common practice reference model for IT service management: formal assessment and implications for practice,” in *Proceedings of the International Conference on e-Technology, e-Commerce and e-Service*, 2005, pp. 704-710.

[20] G. Ridley, J. Young, and P. Carroll, “COBIT and its utilization: a framework from the literature,” in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2014, pp. 1–8.

[21] ISO/IEC, “ISO/IEC 20000:2018 Information Technology,” London, United Kingdom, 2021.

[22] M. Virmani, “Understanding Devops & Bridging The Gap From Continuous Integration To Continuous Delivery,” in *Proceedings of the Fifth International Conference on the Innovative Computing Technology (INTECH 2015). IEEE*, 2015, pp. 78–82.

[23] S. S. Samarawickrama and I. Perera, “Continuous scrum: A framework to enhance scrum with DevOps,” in *Proceedings of the 7th International Conference on Advances in ICT for Emerging Regions (ICTer)*, Sep. 2017, pp. 19–25.

[24] Cristian Barz, C. K. Jalba, Z. Erdei, and S. M. L. Hahn, “Approaches for the planning and implementation of Industry 4.0,” *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 1, pp. 375–380, 2019.

[25] G. de França, B., Jeronimo, H., & Travassos, “Characterizing DevOps by Hearing Multiple Voices,” in *Proceedings of the 30th Brazilian Symposium on Software Engineering - SBES ’16*, 2016, pp. 53–62.

[26] I. Bucena and M. Kirikova, “Simplifying the DevOps Adoption Process,” in *Proceedings of the 16th International Conference on Perspectives in Business Informatics Research*, 2017, pp. 1–15.

[27] T. Riungu, L; Mäkinen, S; Lwakatare, L; Tiihonen, J & Männistö, “DevOps Adoption Benefits and Challenges in Practice: A Case Study,” in *Proceedings of the Product-Focused Software Process Improvement*, 2016, pp. 590–597.

[28] B. S. Farroha and D. L. Farroha, “A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment,” in *Proceedings of the Military Communications Conference*, 2014, pp. 288–293.

[29] H. Chen, R. Kazman, S. Haziyev, V. Kropov, and D. Chtchourov, “Architectural Support for DevOps in a Neo-Metropolis BDaaS Platform,” in *Proceedings of the 34th Symposium on Reliable Distributed Systems Workshop (SRDSW)*, 2015, pp. 25–30.

[30] J. Wettinger, V. Andrikopoulos, and F. Leymann, “Automated Capturing and Systematic Usage of DevOps Knowledge for Cloud Applications,” in *Proceedings of the International Conference on Cloud Engineering*, 2015, pp. 60–65.

[31] S. Nagpal and A. Shadab, “Literature Review: Promises and Challenges of DevOps,” Waterloo, Canada, 2017.

[32] R. N. Jaffar, A. A. A. M. Hussain, and W. Chiad, “A new model for study of quality attributes to components based development approach,” *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 3, pp. 1177–1185, 2019, doi: http://dx.doi.org/10.21533/pen.v7i3.686.

[33] J. Michelsen, *Dysfunction Junction: A Pragmatic Guide to Getting Started with DevOps*. Boston: CA Technologies, 2014.

[34] J. Moore, G. Kortuem, A. Smith, N. Chowdhury, J. Cavero, and D. Gooch, “DevOps for the Urban IoT,” in *Proceedings of the Second International Conference on IoT in Urban Space*, 2016, pp. 78–81.

[35] M. Soni, “End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery,” in *Proceedings of the International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2015, pp. 85–89.

[36] P. Lees, K; Gardner, J & Eaton, “VMware, DevOps and Agile Development,” 2017.

[37] J. Guerrero, K. Zuñiga, C. Certuche, and C. Pardo, "A systematic mapping study about DevOps," *Journal de Ciencia e Ingeniería*, vol. 12, no. 1, pp. 48–62, 2020.

[38] F. Ahmadighohandizi and K. Systä, "ICDO: Integrated Cloud-based Development Tool for DevOps," in *Proceedings of the 14th Symposium on Programming Languages and Software Tools (SPLST)*, 2015, pp. 76–90.

[39] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "An Exploratory Study of DevOps: Extending the Dimensions of DevOps with Practices," in *Proceedings of the Eleventh International Conference on Software Engineering Advances (ICSEA)*, 2016, pp. 1–9.

[40] Floris Erich, C. Amrit, and M. Daneva, "Report: DevOps Literature Review," Enschede, Netherlands, 2014.

[41] A. Ghantous, G.B & Gill, "DevOps: Concepts, Practices, Tools, Benefits and Challenges," in *Proceedings of the Pacific Asia Conference on Information Systems (PACIS)*, 2017, pp. 1–12.

[42] V. Babenko, L. Lomovskykh, A. Oriekhova, L. Korchynska, M. Krutko, and Y. Koniaieva, "Features of methods and models in risk management of IT projects," *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 2, pp. 629–636, 2019.

[43] C. Pardo, O. García, M. Piattini, F. Pino, and M. T. Baldassarre, "A reference ontology for harmonizing process-reference models," *Revista Facultad de Ingeniería Universidad de Antioquia*, vol. 1, no. 73, pp. 29–42, 2014.

[44] Devopsdays, "Devopsdays - Organizing Guide," 2009. https://bit.ly/3i7DKUb (accessed Jul. 27, 2021).

[45] A. Caprarelli, E. Di Nitto, and D. A. Tamburri, "Fallacies and Pitfalls on the Road to DevOps: A Longitudinal Industrial Study," in *Proceedings of the DEVOPS 2019: Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, 2020, pp. 200–210.

[46] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? A Systematic Mapping Study on Definitions and Practices," in *Proceedings of the Scientific Workshop Proceedings of XP2016*, 2016, pp. 1–11.

[47] J. Guerrero, C. Certuche, K. Zúñiga, and C. Pardo, "Trends in devops: A systematic mapping of the literature," *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, vol. 2020, no. E32, 2020.

[48] K. Zúñiga and C. Certuche, "Proceso para soportar DevOps en la integración, entrega y despliegue continuo en Pymes de software," University of Cauca, 2021.

[49] F. Jones, S., Noppen, J & Lettice, "Management challenges for DevOps adoption within UK SMEs," in *Proceedings of the 2nd International Workshop on Quality-Aware DevOps - QUDOS 2016*, 2016, pp. 7–11.

[50] A. A. Khan and M. Shameem, "Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process," *Journal of Software: Evolution and Process*, vol. 32, no. 10, pp. 1–26, Oct. 2020.

[51] M. A. Akbar *et al.*, "Prioritization Based Taxonomy of DevOps Challenges Using Fuzzy AHP Analysis," *IEEE Access*, vol. 8, no. 1, pp. 202487–202507, 2020.

[52] S. Rafi, W. Yu, M. A. Akbar, A. Alsanad, and A. Gumaei, "Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE," *IEEE Access*, vol. 8, no. 1, pp. 105426–105446, 2020.

[53] M. A. Rothenberger, J. Humble, J. B. Thatcher, D. Smith, and N. Forsgren, "A Taxonomy of Software Delivery Performance Profiles: Investigating the Effects of DevOps Practices," in *Proceedings of the Americas Conference on Information Systems*, Aug. 2020, pp. 1–6.

[54] R. W. Macarthy and J. M. Bass, "An Empirical Taxonomy of DevOps in Practice," in *Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020, pp. 221–228.

[55] C. Orozco, C. Pardo, and S. Vásquez, "SCMOnto: Una ontología para soportar la gestión de la configuración de software," *Iberian Journal of Information Systems and Technologies*, vol. 38, no. 12, pp. 75–90, 2020.

[56] D. Parsons, "Agile software development methodology, an ontological analysis," in *Proceedings of the 9th International Conference on Applications and Principles of Information Science*, 2010, pp. 1–4.

[57] M. A. McCarthy, L. M. Herger, S. M. Khan, and B. M. Belgodere, "Composable DevOps: Automated Ontology Based DevOps Maturity Analysis," in *Proceedings of the International Conference on Services Computing*, Jun. 2015, pp. 600–607.

[58] I. S. Committee, "IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: IEEE Standard 2675-2021," 2021.

[59] D. Fensel, “Ontology-based knowledge management,” *Computer (Long Beach Calif)*, vol. 35, no. 11, pp. 56–59, Nov. 2002.

[60] M. Fernandez-lopez, Gomez-Perez.A, and Juristo. N, “METHONTOLOGY: from Ontological Art towards Ontological Engineering,” *Proceedings of the AAAI97 Spring Symposium*, Stanford, USA, pp. 33–40, 1997.

[61] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowledge Acquisition*, vol. 5, no. 2. pp. 199–220, 1993.

[62] T. Hikita and M. J. Matsumoto, “Business process modelling based on the ontology and first-order logic,” in *Proceedings of the 3rd International Conference on Enterprise Information Systems*, 2001, pp. 717–723.

[63] C. Tautz and C. G. Wangenheim, “REFSENO: a representation formalism for software engineering ontologies,” Fraunhofer, 1998.

[64] U. Reimer, “Schema-based and network-based rendering formats,” in *Proceedings of the Einführung in die Wissensrepräsentation*, 1991, pp. 28–78.

[65] L. F. Sikos, *Description Logics in Multimedia Reasoning*. Cham: Springer International Publishing, 2017.

[66] H. S. Pinto, A. Gómez-Pérez, and J. P. Martins, “Some Issues on Ontology Integration,” in *16th International Joint Conference on Artificial Intelligence (IJCAI’99)*, 1999, vol. 18, pp. 7–12.

[67] J. Euzenat and P. Shvaiko, *Ontology Matching*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[68] F. Ruiz and J. R. Hilera, “Using Ontologies in Software Engineering and Technology,” in *Ontologies for Software Engineering and Software Technology*, Springer, Berlin, Heidelberg, 2006, pp. 49–102.

[69] F. García *et al.*, “Towards a consistent terminology for software measurement,” *Inf Softw Technol*, vol. 48, no. 8, pp. 631–644, Aug. 2006.

[70] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “Dimensions of DevOps,” in *Proceedings of the Agile Processes in Software Engineering and Extreme Programming - XP*, 2015, pp. 212–217.

[71] S. Lohmann, S. Negru, and D. Bold, “The ProtégéVOWL Plugin: Ontology Visualization for Everyone,” 2014, pp. 395–400.

[72] W3C, “OWL 2 Web Ontology Language Document Overview,” *Second edition*, 2020. https://bit.ly/3l0GCnB (accessed Mar. 21, 2023).

[73] R. Almeida, I. Percheiro, C. Pardo, and M. M. da Silva, “An Ontology-Based Model for ITIL Process Assessment Using TIPA for ITIL,” in *Software Process Improvement and Capability Determination*, 2018, pp. 104–118.

[74] GitLab, “Introduction to CI/CD with GitLab,” 2020. https://bit.ly/2BiX7Dq (accessed Mar. 21, 2023).

[75] M. A. McCarthy, L. M. Herger, S. M. Khan, and B. M. Belgodere, “Composable DevOps: automated ontology based DevOps maturity analysis,” in *Proceedings of the International Conference on Services Computing*, 2015, pp. 600–607.

[76] A. A. Khan and M. Shameem, “Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process,” *Evolution and Process*, vol. 32, no. 10, p. e2263, 2020.

[77] M. Akbar *et al.*, “Prioritization Based Taxonomy of DevOps Challenges Using Fuzzy AHP Analysis,” *IEEE Access*, vol. 8, pp. 426–446, 2020.

[78] S. Rafi, W. Yu, M. Akbar, A. Alsanad, and A. Gumaei, “Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE,” *IEEE Access*, vol. 8, pp. 426–446, 2020.

[79] N. Forsgren, M. A. Rothenberger, J. Humble, J. B. Thatcher, and D. Smith, “A Taxonomy of Software Delivery Performance Profiles: Investigating the Effects of DevOps Practices,” 2020.

[80] R. W. Macarthy and J. M. Bass, “An empirical taxonomy of DevOps in practice,” in *Proceedings of the 46th Euromicro Conf. on Software Eng. and Advanced Applications*, 2020, pp. 221–228.