(Online) ISSN 2303-4521
**Periodicals of Engineering and Natural Sciences** *Original Research*
Vol. 13, No. 2, April 2025, pp.305-326

# Improving understanding of the DevOps framework using Essence a visual representation

**Paola Noreña-Cardona[1], Claudia Durango[2], Elizabeth Suescún[1], César Pardo[3*]**
[1] Escuela de Ciencias Aplicadas e Ingenierías, Universidad EAFIT, Medellín, Colombia
[2] Facultad de Ingenierías, Universidad de San Buenaventura, Medellín, Colombia
[3] University of Cauca, IDIS Research Group. Calle 5 No. 4-70, Popayán, Colombia

*Corresponding author E-mail: cpardo@unicauca.edu.co

**ABSTRACT**

DevOps aims to achieve effective integration between software development (Dev) and system operations (Ops) through the implementation of agile practices. These practices allow for a quick response to frequent changes that arise in the development cycle for a software product. In the advancements of agile practices in DevOps, it is common to find a lack of conceptual and visual integration in the DevOps development cycle, which complicates the understanding of these practices in the various DevOps phases. In this paper, we propose a visual representation of the practices in the DevOps using Essence, a standard framework for creating, adopting, and improving best practices in software engineering. Such a visual representation is an addition to the DevOps framework, which integrates processes, practices, principles, values, metrics and main concepts of DevOps. The proposed solution aims to enhance the understanding and impact of practices in DevOps, providing a comprehensible vision for industry professionals. Essence serves as a structured method to encapsulate and visualize the various elements of practices, ensuring that they are accessible and easily understandable. This visual representation helps bridge the gap between theoretical concepts and their practical application in a DevOps environment. By doing so, it facilitates better communication and collaboration among team members, leading to more efficient and streamlined operations. The results indicated the understandability and clarity of the representation of practices in the DevOps lifecycle, making it a useful tool for industry professionals seeking to optimize their DevOps workflows.

**Keywords**: Agile practices, Best practices, DevOps, Essence, Software development.

## 1. Introduction

DevOps integrates software development (Dev) and operations (Ops) teams to streamline the entire software lifecycle, focusing on process optimization through automation [1], [2]. This approach speeds up deliverables, enhances quality and value, and significantly reduces costs [3]. DevOps culture emphasizes an organizational shift towards technological transformation, linking improved organizational performance directly with software quality. The DevOps lifecycle encompasses several phases in a collaborative and agile environment: Planning, Development, Integration, Monitoring, Deployment, and Operation. These phases crosscut from the inception of software project ideas to the automated deployment and continuous updating of software in production environments, including automated testing and monitoring [3], [4].

Essence is a standard with a set of common visual elements for all software development projects. These elements help development teams identify the best practices and assess the status and health of the software project. To enhance understanding, the elements are organized into three areas of concern: in (i) the *customer*, the team needs to understand the stakeholders and the opportunity to be addressed; in (ii) *solution*, specification, and development of the software system and in (iii) *endeavor*, everything to do with the team, and the way that they approach their work. Each focus on a specific dimension of software development. “Throughout the

diagrams in the body of the kernel specification, the three areas of concern are distinguished with different color codes where green stands for customer, yellow for solution, and blue for endeavor" [5]–[7].

In the scientific literature, we found DevOps utilizes agile practices (also called capabilities), which involve applying agile principles and frameworks in software development and IT operations to achieve rapid, iterative, and high-quality software delivery. According to the review [3], [8]–[14], the general practices in DevOps are: (i) *Continuous delivery*, which enables quick, safe, and sustainable changes in software development, including features, configuration, bug fixes, and production experiments. (ii) *Technical or architectural management*, involving architectural decisions that impact software delivery performance and architecture effectiveness. (iii) *Product and process management*, which involves understanding the product through small deliveries of a minimum viable product (MVP) and understanding the workflow from business to product. (iv) *Lean management*, focusing on creating customer value through systemic thinking, quality assurance, flow creation, and respectful leadership. Initially applied to manufacturing by Toyota and Honda, Lean was adapted to software development in 2003 by Mary and Tom Poppendieck. (v) *Cultural management*, reflecting cooperation and collaboration among organization members, and defining norms for work, communication, and problem-solving [3].

Some works represent conceptual and graphical DevOps [14]–[25]. However, such works lack representation in the DevOps lifecycle and the elements related to practices (activities, work products, roles, and competencies). This gap presents three challenges: (i) lifecycle integration, existing representations do not map DevOps practices across its entire lifecycle, making it difficult to understand how different practices interconnect and evolve over time, (ii) fragmentation of key elements, some approaches focus on principles and high-level concepts but do not detail the concrete activities, roles, and work products operationalize DevOps, and (iii) absence of a standardized conceptual framework, without a common structure, organizations struggle with inconsistent terminology and interpretations, leading to inefficiencies in collaboration and process optimization.

In this context, we propose a visual and conceptual representation of practices in the DevOps lifecycle by using Essence, along with their activities, work products, roles, and competencies. Such a representation is an addition to the DevOps Framework, which is a compendium of key practices, principles, and dimensions essential for DevOps implementation. Our approach attempts to address the gaps by providing lifecycle coverage, we integrate DevOps practices within a structured lifecycle representation, ensuring visibility into how they evolve and interact. Detailing key practice elements, our approach includes activities, work products, roles, and competencies, offering a structured understanding of DevOps implementation. Standardizing concepts through Essence, we establish a shared terminology and a flexible framework for facilitating communication, collaboration, and continuous improvement within DevOps teams. Similarly, it is a supporting strategy for continuous improvement, as it provides a common language and a flexible framework for implementing changes to optimize processes.

The paper is structured as follows: in Section 2, we present a theoretical framework and background for conceptualizing agile practices, DevOps, and Essence. In Section 3, we related work on DevOps agile practices and the representation of their elements in the DevOps lifecycle. In Section 4, we present the DevOps framework with the addition of the proposal. In Section 5, we propose a representation of agile practices in the DevOps lifecycle using Essence. In Section 6, we provide a validation of the proposal. Finally, in Section 7, we present conclusions and future work.

## 2. Theoretical framework and background

### 2.1 Agile practices

Agile practices emerged in the 1990s as an alternative approach to software development, aiming to satisfy customer needs by improving operational requirements and preventing failures through incremental processes. Observation is therefore critical for identifying changes and limitations in software development [27]. In 2001, the Agile Manifesto was drafted, setting values and principles that prioritize flexibility and collaboration over adhering to rigid pre-established plans. The Agile Manifesto advocates an iterative approach, with teams conducting short production cycles, frequently delivering functional software that can adapt to changes as the project progresses. It also values interaction among individuals and the quality of the software over extensive documentation and bureaucratic processes [28], [29].

### 2.2 DevOps

DevOps represents an integral approach to software development and operation, involving philosophy, agile culture, and practices, and tools designed to streamline software development deliveries. Its name comes from the integration of two teams that used to operate independently: the development team (Dev) and the operations team (Ops). This approach focuses on optimization, mainly through automation of processes throughout the entire software life cycle [1], [2]. The philosophy of DevOps culture takes an organizational perspective, where improving organizational performance is directly influenced by software quality, quick feedback, and performance. In this sense, the adoption of DevOps practices becomes a journey toward the technological transformation of the organization, characterized by accelerated deliverables, improved quality and value, and significant cost reduction [3].

In this approach, a series of phases are used in software development and IT operations in a collaborative and agile environment during of DevOps life cycle [4]: (i) *Planning*, the ideas to start the software project; (ii) development, the software and components of the architecture are implemented; (iii) *Integration*, integration testing activities, functional tests, unit tests, among others, are automated; (iv) *Monitoring* evaluates the process, personnel, and resources; (iv) *Deployment*, the deployment of the software in a production environment is automated without human intervention and (v) *Operation*, features of the architecture are implemented, these features are updated and the code is updated [3].

### 2.3 Essence

Essence is a proposal that has its origin in SEMAT's Essence concept, it is a standard framework for the creation, use, and improvement of practices in software engineering, in the present work it serves for the representation, specification, and description of fundamental elements such as methods and practices related to the DevOps framework. Essence is a standard that provides a collection of universal elements applicable to all software development projects. Elements of the Essence notation used in the representation are presented in Figure 1.

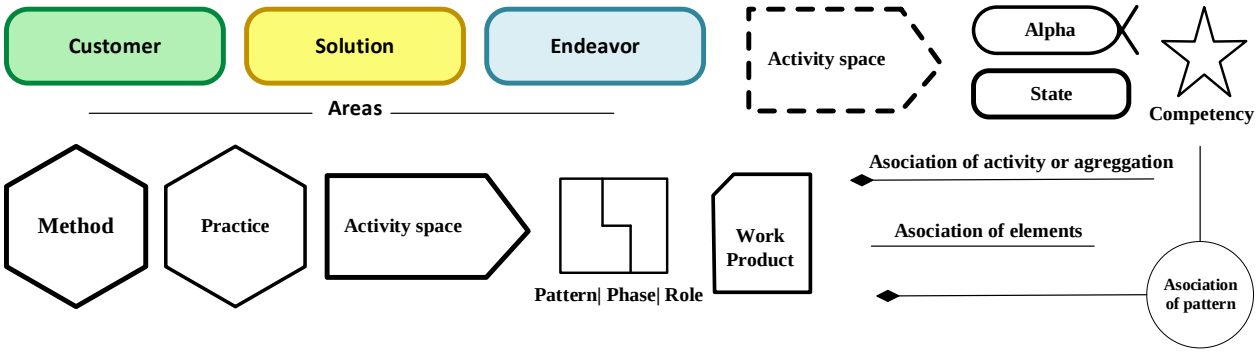



Figure 1. Essence notation [7]

In Figure 1 Essence areas are highlighted using three colors: *customer* (green), which encompasses the users and customers of the software system; *solution* (yellow), which covers everything related to the development of the system aimed at solving the problem; and *endeavor* (blue), which includes everything related to the development team and their activities [7]. The other elements are related to the areas, so they are represented with the color corresponding to the area: *activity space* (things to do), *alpha* (things to work with), *state* (progress of alphas), and *competence* (skill to perform an activity), which are defined according to the Essence standard. The other elements are defined based on the components of the method or practice for being represented: *method* (a set of practices that guide a framework), *practice* (actions for process improvement and standardization in software engineering), *activity* (one or more types of work elements), *pattern* (repeatable elements such as property, milestone, role, and phase, among others), and *work product* (a valuable artifact result from the effort of the activities performed in the process) [5]–[7].

Essence is a standardized framework designed to support the creation, use, and improvement of software engineering practices. At its core lies a kernel composed of universal elements—Alphas, States, Activity Spaces, and Competencies, which represent the essential aspects of any software endeavor. Alphas model critical entities (e.g., requirements, software system, team) and evolve through defined states, allowing teams to track progress and assess project health independently of specific methodologies. Activity spaces define what needs to be done, while competencies identify the skills required. Practices and methods are then constructed by composing these

elements, enabling teams to reflect, adapt, and improve continuously. This paper applies Essence to visually represent DevOps-related practices and artifacts, as detailed in Figure 1 and Table 1 [5]–[7].

Table 1. Key Essence elements and their application in DevOps

| Essence element | Description | Application in DevOps |
|---|---|---|
| Area: Customer (green) | Encompasses users and stakeholders of the software system. | Defines needs, expectations, and validates the value delivered continuously. |
| Area: Solution (yellow) | It covers all aspects related to the system being developed to solve a problem. | Represents artifacts such as code, infrastructure as code, deployed services, etc. |
| Area: Endeavor (blue) | Includes the development team and its organizational activities. | Visualizes DevOps workflows, shared responsibilities, automation, and continuous improvement efforts. |
| Alpha | Core element that evolves during the software development process. | Tracks the status of key components (e.g., requirements, software system, team) throughout the DevOps cycle. |
| State | Stages that represent the progress of an alpha. | Helps assess progress (e.g., refined backlog, tested system, ready for deployment). |
| Activity Space | Defines what needs to be done areas of work. | Highlights key DevOps practices such as CI, monitoring, automated testing, and deployment. |
| Competence | Skills required to perform activities effectively. | Assesses and develops technical capabilities (e.g., automation, scripting, observability). |
| Method | A structured set of practices guiding the work. | Specifies how DevOps practices are integrated into the software development life cycle. |
| Practice | Actionable steps aimed at improving or standardizing software processes. | Examples include continuous delivery, early feedback, and configuration management. |
| Activity | A unit of work carried out by a practitioner. | Tasks such as pipeline configuration, test execution, and metrics review. |
| Pattern | Reusable structure is used to model practices. | Includes roles (e.g., DevOps engineer), milestones (e.g., release-ready), and phases (e.g., deployment). |
| Work Product | Tangible artifacts are produced as a result of activities. | Examples: automation scripts, Docker containers, monitoring dashboards. |

The strength of Essence in DevOps environments lies in its ability to graphically represent the evolving state of each key aspect of a software project, enabling a clear and shared understanding among various stakeholders—developers, operators, testers, and managers. In DevOps contexts, where collaboration and continuous automation are critical, having a visual model that reflects the state of the alphas helps align teams, identify bottlenecks, and foster evidence-based continuous improvement.

## 3. Related work

Related work includes a comprehensive review of the segmentation of practices according to the five general DevOps practices. This review was conducted following a structured two-phase methodology. In the first phase, we systematically reviewed the practices and their forms of visualization in previous studies. A detailed overview of these practices, including their descriptions, visual representations, and references, is presented in Table 2 [3], [8]–[12], [14]. In the second phase, we focused on the representations of the DevOps lifecycle. Prior representations of the DevOps lifecycle are compiled in Table 3. From our review, we observe that these practices are predominantly defined conceptually along with their constituent activities. However, they are not explicitly aligned with the phases of the DevOps lifecycle, which complicates their practical application in specific phases. Additionally, existing representations of DevOps lifecycle elements are found in various formats, such as tables, conceptual maps, activity diagrams, UML diagrams, and ontologies. Nevertheless, these representations often lack the necessary elements to elucidate the interrelationships between practices, phases, activities, work products, roles, and the competencies required for effective DevOps implementation [14]–[25].

Table 2. DevOps practices compendium

| General practice | Agile practice | Description | Visualization | References |
|---|---|---|---|---|
| Continuous delivery (CD) | Continuous Integration (CI) | Practice for making significant changes to the source code frequently and regularly. | Conceptual and graphical | [3], [9], [11], [14] |
| | Version control | Practice performing CI with modifications to the code, storing it in a shared manner for all team members. This allows teams to work in parallel, accelerate the release, and quickly fix errors. | Conceptual | [9], [12] |
| | Configuration management (CM) | Practice managing software changes utilizing version control in a standard and repeatable way, configuring infrastructure, and software dependencies. | Conceptual | [9], [10], [14] |

| General practice | Agile practice | Description | Visualization | References |
|---|---|---|---|---|
| | Trunk-based development | Software change or version control practice, where the trunk is the branch or line for facilitating change management in development in a visible way for the team. When the trunk is ready, CI, CD, and testing practices are carried out. | Conceptual | [3], [9] |
| | Hypothesis-driven development (HDD) | An approach to software development that relies on hypotheses about the impact of a new product, feature, or change on users, based on hypothesis. | Conceptual and graphical | [13] |
| | Continuous and automated testing | Practice for testing the most up-to-date version of the source code. Developers must first create and maintain experimental manual tests and acceptance tests, which are automated by testers in the Quality Assurance (QA) team to then fix errors in each version change. An important practice is Test-Driven Development (TDD), where developers create more testable designs. | Conceptual | [9], [11], [14] |
| | Test data management | Practice for running required automated tests with the purpose of validating and improving software functions. | Conceptual | [3], [9], [14] |
| | Continuous Deployment (CD) | Practice deploying the software to production once it is ready and available for use, following testing. | Conceptual and graphical | [3], [9] |
| **Technical or architectural management** | Loosely coupled architecture | The architectural paradigm for creating individual components independently (without being tightly coupled) using asynchronous communication mechanisms between components such as web services, messages, or events, enables developers to deliver software more quickly. | Conceptual | [9], [11] |
| | Infrastructure as code (IaC) | Practice automating infrastructure using configuration files or scripts instead of relying on physical hardware configuration. | Conceptual | [9], [11] |
| | Cloud infrastructure | Practice for integrating computing resources and services from cloud services such as servers and virtual machines. | Conceptual | [8], [9], [11], [14] |
| | Security as code (Sac) | Practice for incorporating security principles into software development through DevOps tools in code and infrastructure changes. | Conceptual | [8], [9], [11] |
| | Security by design | Practice for implementing security from architecture definition, requirements, and risks to development. | Conceptual | [9], [11], [14] |
| **Product and process management** | Requirements management | Practice for tracking, testing, analyzing, visualizing, and communicating requirements to stakeholders and developers. | Conceptual | [9], [10] |
| | Feedback and Continuous Improvement | Practices for promoting continuous improvement of the process, personnel, and product, as well as innovation in continuous learning, allow software teams to use solutions to problems, reduce risks, and achieve more resilient software. | Conceptual | [10], [11], [14] |
| | Working in small batches | Practice quickly testing hypotheses about a particular improvement to ensure it has the desired effect. Although this practice can be applied to any type of change, including organizational transformation and process improvement, it primarily focuses on software delivery. | Conceptual | [9], [11] |
| | Team experimentation | Practice for teams to experiment by quickly creating prototypes and testing ideas tailored to allow them to discover aspects about users and the problem, and design solutions based on the software. Then, teams incorporate what they learned into the product or service design to add value to the organization. | Conceptual | [9], [11] |
| **Lean management** | Change Approval | Practice reviewing and approving proposed changes or modifications to software before implementing them in production. | Conceptual | [9], [11] |
| | Proactive Fault Notification | Practice for continuously maintaining the socialization and/or documentation of the team learnings. | Conceptual | [9], [10] |
| | Visibility of work | Practice for teams to gain an understanding of the workflow, the state of the software product, and its features, allowing them to focus on completing current tasks. | Conceptual | [9] |
| | Limit Work in Progress (WIP) | Practice for ensuring the team is not overloaded and exposes workflow obstacles. It also allows for guiding timelines, the performance of deliverables, and having a visual display of feedback. | Conceptual | [9], [11] |
| | Continuous monitoring and observability | Practice observing and understanding the state of their systems through metrics or logs and actively debugging their system to explore required properties. | Conceptual and graphical | [9], [11], [14] |

| General practice | Agile practice | Description | Visualization | References |
|---|---|---|---|---|
| Cultural management | Generative organizational culture | A generative organizational culture should optimize the flow of information. Additionally, it indicates that system security includes human factors, so information flows influence the culture through timely and effective questions and answers. | Conceptual | [9], [11] |
| | Continuous learning | Learning is a key to improvement and an investment for the organization. To achieve this, organizations allocate a budget for training, resources for informal learning, and time to provide learning opportunities. | Conceptual and graphical | [9], [11], [14] |
| | Team Collaboration | Practice in which development and operations teams work together from the initial stages of design and planning, defining requirements and architecture, with other teams such as quality and security also collaborating. | Conceptual | [9] |
| | Performance measurement | Practice for enabling the measurement of capabilities driving software delivery and organizational performance to determine improvement actions. | Conceptual | [9], [11], [14] |
| | Job satisfaction | Generate employee well-being in professional development, workplace environment, and their physical, emotional, and mental health. | Conceptual | [9] |
| | Transformational leadership | It is a leadership style that involves five dimensions: vision, inspirational communication, intellectual stimulation, supportive leadership, and personal recognition. | Conceptual | [9], [10], [14] |
| | DevOps Education | Practice related to continuous learning but focused on DevOps practices and tools that drive software delivery improvements and organizational performance. | Conceptual | [10], [14] |

Table 3. Related work representing the DevOps lifecycle

| No. | Identified Elements | Representation | Reference |
|---|---|---|---|
| 1 | Practice and activities | Activity diagram and conceptual representation | [17] |
| 2 | DevOps models | Conceptual | [19] |
| 3 | Phases, activities, products, processes, and metrics | Diagram with phases and activities and conceptual representation | [25] |
| 4 | Practices and quality attributes | Elements diagram | [22] |
| 5 | Practices, dimensions, and values | Table | [14] |
| 6 | Tools, practices, infrastructure, and work team | Table | [24] |
| 7 | Practices | Table | [15] |
| 8 | Practices | Conceptual map and table | [16] |
| 9 | Practices | Conceptual | [18] |
| 10 | Dimensions, principles, practices, tools, roles, activities, products, and metrics | UML (Unified modeling language) representation and Ontology | [23] |
| 11 | Phases, principles, and practices | Diagram with phases, conceptual map, and Flow diagram | [21] |
| 12 | Practices and roles | Conceptual | [20] |

In DevOps, where agility, automation, and cross-team collaboration are essential, Essence provides a method-agnostic way to visualize and track the progress of key engineering elements. Its standardized structure supports alignment across diverse roles and practices. At Fujitsu UK, Essence was applied to integrate agile and traditional methodologies, enabling a shared understanding of software processes through its kernel concepts—Alphas, States, and Lenses. Practical workshops facilitated organizational buy-in, while the multi-level adaptation of Essence helped harmonize local practices with global delivery goals [26].

Although Essence has been applied in various contexts —such as agile adoption, systems engineering, and method unification— it has not been extensively explored within DevOps environments. This gap represents an opportunity to investigate how Essence can support the visualization of DevOps-specific practices and workflows. The present work addresses this gap by proposing a visual representation that aligns DevOps principles with the kernel concepts of Essence, enabling teams to reflect on progress, coordination, and process maturity across the development and operations spectrum.

## 4. DevOps framework

The DevOps framework is an approach for integrating key practices, principles, and dimensions essential for DevOps implementation [10]. Such a framework aims to provide clear guidelines for automating processes, improving collaboration, and ensuring continuous delivery and integration. Its structure is based on the importance of a well-defined, consistent, and scalable DevOps model that supports organizational goals and enhances productivity. Figure 2 presents the elements of the DevOps framework and the addition of the visual representation of agile practices in the DevOps lifecycle using Essence as the proposal of this work.

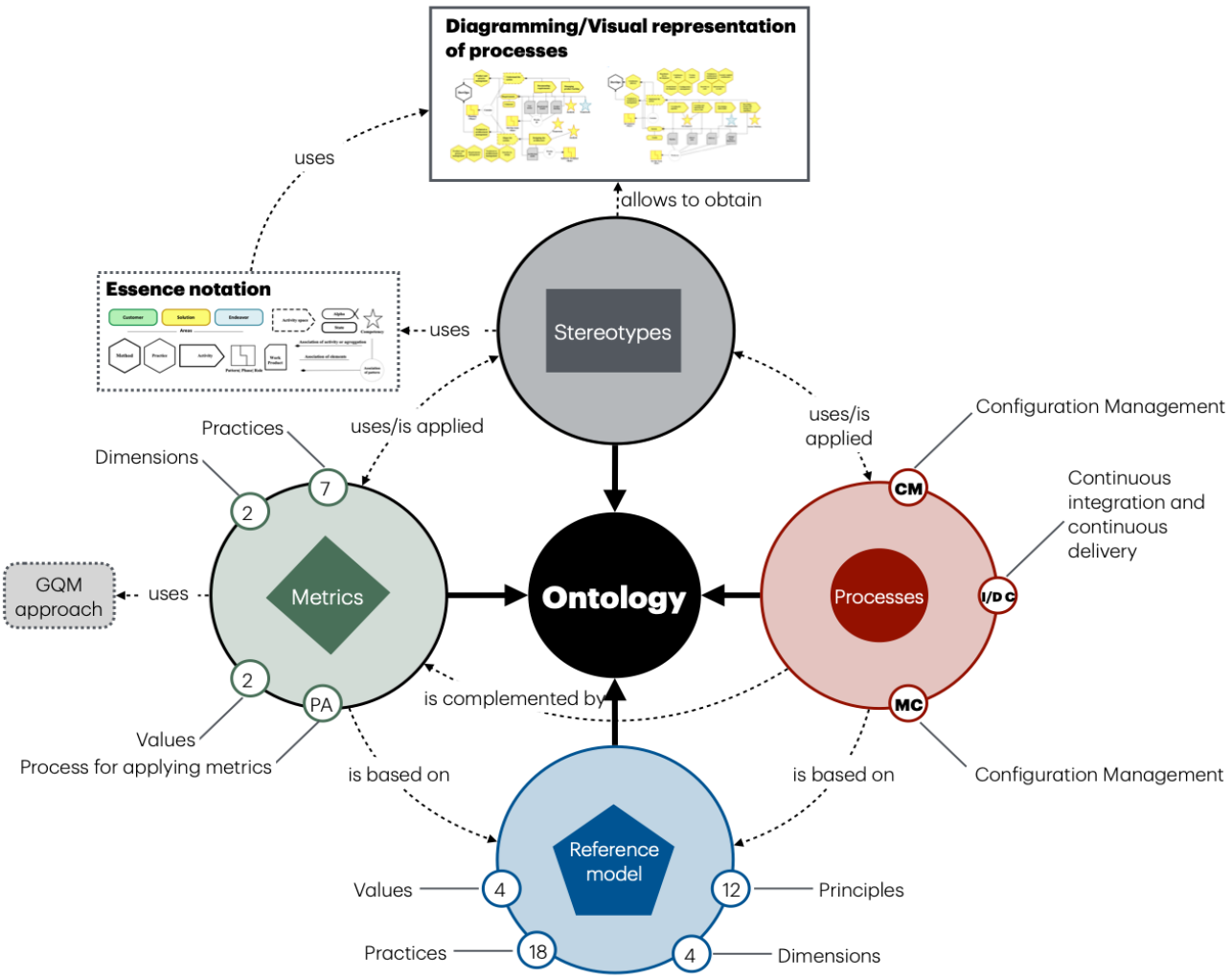



Figure 2. DevOps framework adding the proposal visual representation. The author based on [10], [14], [23]

### 4.1. Reference model

The proposed DevOps reference model is composed of four elements, the first element describes four values related to: (i) automation, (ii) collaboration, (iii) measurement and (iv) communication. The second element suggests 12 principles adapted from the Agile Manifesto. The third element describes 18 practices, 12 fundamental and 6 complementary. Finally, the fourth element proposes 4 dimensions that allow categorizing the proposed practices, the suggested dimensions are: (i) People, (ii) Culture, (iii) Processes and (iv) Tools. The four proposed elements are described in detail in [30], likewise, a case study which presents the findings of its application through a software development organization are presented. According to Figure 2, the reference model is the conceptual foundation of the DevOps framework, therefore, the practices are represented and extended in the visual representation of our proposal.

### 4.2. Metric model

To support the assessment of practices, dimensions, and values proposed for the implementation of DevOps in software companies, a metrics model was created as described in [14]. This model was developed by harmonizing the DevOps process elements identified through a literature mapping described in [14], to understand the current state of methodological solutions and tools for evaluating DevOps in the industry. Besides, in [14], the process elements were identified, compared, and integrated into a common structure, defining a total of 11 metrics used in the GQM (Goal-Question-Metric) approach. Such a metrics model was evaluated by a focus group composed of DevOps experts, who concluded that it is a clear, easy-to-implement model that provides valuable information to companies to improve their DevOps-related practices. The purpose of the metrics model is to assess the implementation of DevOps through a set of questions that measure compliance with DevOps practices, dimensions, and values. The results obtained from applying this model enable a software development company to determine the extent to which they have adopted DevOps and reveal

areas for improvement related to how they apply their practices, as shown in [14] through two case studies. In this regard, the metrics model described in these works can be integrated into the present proposal to support the understanding and evaluation carried out by a consultant using metrics following the formalism of the GQM approach; evaluate the level of implementation of agile practices in the DevOps lifecycle using Essence in software companies as a standard framework for creating, adopting, and improving practices in software engineering; and identify areas for improvement in the processes used by companies to adopt and apply DevOps. According to Figure 2, the metrics model contains practices, dimensions, values, and processes for applying the GQM approach and is based on the reference model.

### 4.3. Processes

The framework also suggests a process to encourage and support the adoption of DevOps in small and medium-sized software companies [31]. Its elements were designed by identifying elements suggested in the literature through a systematic mapping presented in [32], [33]. The identified elements were harmonized, compared, and integrated, resulting in a clear, homogeneous process without terminological conflicts since it makes use of the developed ontology [23]. The process is composed of three subprocesses that describe: roles, activities, artifacts, tools, and process flow in BPMN. The three subprocesses are: configuration management (CM), integration, deployment (CI/CD), and continuous monitoring (CM) from a fundamental and complementary approach to suggested practices. In total, the process describes a total of 82 activities, 16 artifacts, 9 roles and recommends 13 technological tools to support the automation of each of the activities related to each subprocess, which will guide professionals and companies to reduce subjectivity in the understanding and adoption of DevOps. In addition, the proposed process was evaluated through a focus group made up of DevOps experts, who considered it to be relevant, clear, complete, and applicable in small and medium-sized software companies. Then, such elements also are represented in the visual representation of our proposal. According to Figure 2, the element “processes” is based on the “metrics model”.

### 4.4. Ontology

The ontology proposed 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. In this way, DevOps Ontology can be used to support the understanding of DevOps in the academy and the software industry. DevOps Ontology was integrated with several concepts from the Software Measurement Ontology (SMO) sub-ontology, which allows to clarify and establish the essential elements in defining software measures and terminology related to software measurement actions [23]. Some DevOps concepts related to ontology are approach, dimension, indicator, measure, measurement, value, practice, principle, product, role, scale, task, and technological tool [23]. According to Figure 2, the ontology supports all elements of the DevOps Framework.

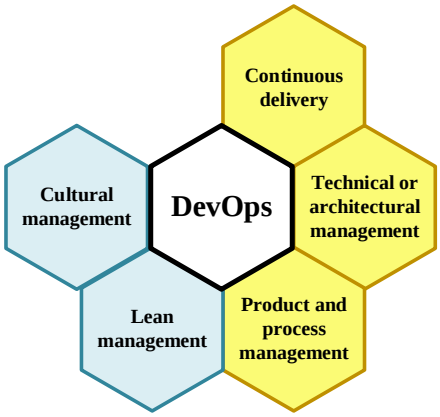
## 5. Representation of DevOps framework practices using Essence

The process followed to create the representation can be described as follows: initially involved the collection of practices and elements identified through a review of existing literature and previous work and described in [10]. Subsequently, the sub-practices were grouped within the general DevOps practices to achieve a representation of each general practice in Essence. Finally, the phases of the DevOps lifecycle were detailed using Essence to generate a complete and comprehensible representation, establishing the relationships between the identified practices and the elements involved in each of these stages.

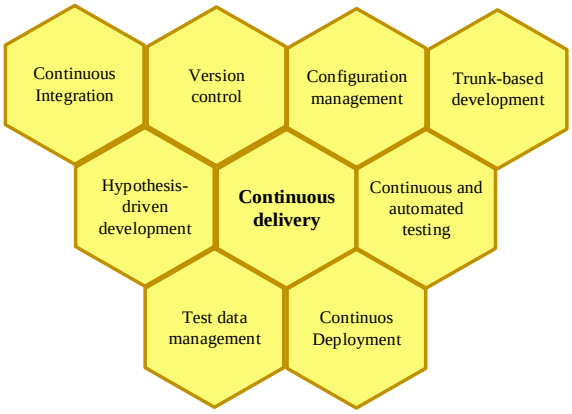
### 5.1. Identify essential DevOps practices and elements

We initially depicted the general DevOps practices based on the Essence notation in Figure 3 (a), they are product and process management, continuous delivery, and technical or architectural management are represented in yellow, reflecting their focus on software solutions. Conversely, cultural management and Lean management are highlighted in blue, as they are linked to the team and their way of working. Subsequently, the sub-practices are integrated within each general practice, following the same structure established according to the corresponding practices detailed in Table 2. This integration is illustrated in Figure 3, demonstrating how each sub-practice is articulated within the framework of the general DevOps practices (five): (i) Continuous delivery is composed of eight practices in Figure 3 (b): continuous integration, version control, configuration management, trunk-based development, hypothesis-driven development, continuous and automated testing, test

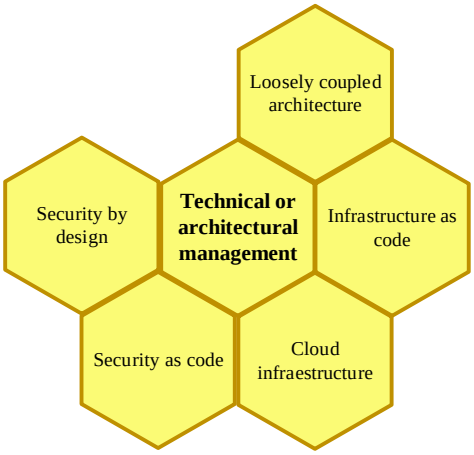
data management, and continuous deployment. (ii) Technical or architectural management is composed of five practices in Figure 3 (c): loosely coupled architecture, infrastructure as code, cloud infrastructure, security as code, and security by design. (iii) Product and process management is composed of four practices in Figure 3 (d): requirements management, feedback, continuous improvement, working in small batches, and team experimentation. (iv) Lean management is composed of five practices in Figure 3 (e): change approval, proactive fault notification, visibility of work, limit work in progress, continuous monitoring, and observability. (v) Cultural management is composed of seven practices in Figure 3 (d): generative organizational culture, continuous learning, team collaboration, performance measurement, job satisfaction, transformational leadership, and DevOps education (see concepts in Table 2).



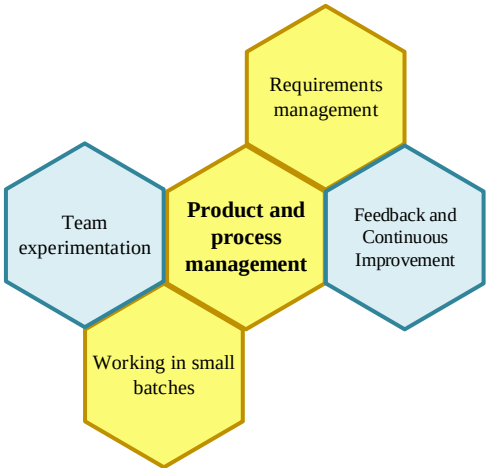

(a) General practices DevOps



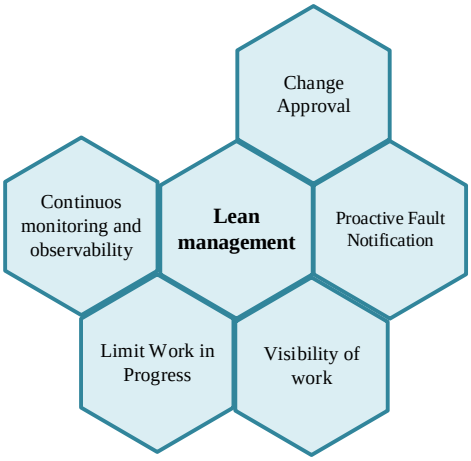

(b) Practices on Continuous delivery




(c) Practices in Technical or architectural management




(d) Practices in Product and process management




(e) Practices in Lean management


(f) Practices in Cultural management

Figure 3. Notation and practice DevOps in Essence. The authors

### 5.2. Description of DevOps practices in the Essence context

We represented the DevOps lifecycle in Figure 4, which encompasses the following six phases: planning, development, integration, monitoring, deployment, and operation, starting in the planning phase and ending in the operation phase. Such representation provides a high-level view of the phases of the DevOps lifecycle, which are represented using Essence notation, allowing visualization of how the phases relate within the software development and operation process under the DevOps approach.

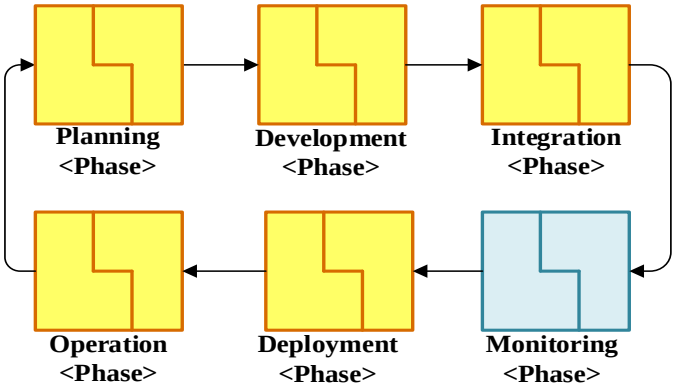



Figure 4. Phases of the DevOps lifecycle in Essence. The authors

Also, we identified practices, activities (including roles and competencies), and work products associated with each of these phases. This analysis is presented in Table 4, which details the elements corresponding to each phase of the lifecycle based on Essence elements, with contributions from literature review and validated with experts in the field.

Table 4. Description of DevOps practices in the Essence context. The authors

| Phase | Practices | Activities (roles and competences) | Work products |
|---|---|---|---|
| Planning | • Product and process management: Requirements management<br>• Technical or architectural management: Security by design | • Documenting requirements<br>• Managing product backlog<br>• Role: DevOps team and Competences: Analysis and Teamwork<br>• Designing the architecture<br>• Role: Software Architect and competence: Systems Thinking | • User stories<br>• Requirements models<br>• Product backlog<br>• Architecture model |
| Development | • Continuous delivery: Version control, Configuration management, Trunk-based development, Hypothesis-driven development<br>• Technical or architectural management: Loosely coupled architecture, Infrastructure as code, Security as code | • Creating the pipeline<br>• Creating and updating the source code<br>• Developing deliverable<br>• Recording version and changes in repository<br>• Role: DevOps team and Competencies: Development, Communication, Systems Thinking | • Pipeline<br>• Source code<br>• Delivery<br>• Change log in Repository |
| Integration | • Continuous delivery: Continuous Integration, Version control, Continuous and automated testing, Test data management | • Performing tests<br>• Role: QA Team and Competencies: Systems Thinking and Testing | • Test execution report |

| Phase | Practices | Activities (roles and competences) | Work products |
|---|---|---|---|
| Monitoring | ● Product and process management: Requirements management, Feedback and Continuous Improvement, Working in small batches, Team experimentation<br>● Cultural management: Generative organizational culture, Continuous learning, Team Collaboration, Performance measurement, Job satisfaction, Transformational leadership, DevOps Education<br>● Lean management: Change Approval, Proactive Fault Notification, Visibility of work, Limit Work in Progress, Continuous monitoring and observability | ● Tracking team<br>● Tracking process<br>● Tracking resources (software, network, infrastructure, among others)<br>● Measuring performance<br>● Role: DevOps team and competencies: Systems Thinking, Collaborative work, and Leadership | ● Progress of the process<br>● Training plan<br>● Event log (feedback and incidents)<br>● Measurement report |
| Deployment | ● Continuous delivery: Continuous deployment, Configuration management, Version control | ● Configuring the production environment<br>● Generating configuration files<br>● Compiling code in the production environment<br>● Role: DevOps team and competencies: Systems Thinking and Development | ● Configuration Files<br>● Change Log in Repository<br>● Deployment Scripts |
| Operation | ● Continuous delivery: Version control, Configuration management<br>● Technical or architectural management: Infrastructure as code, Security as code, Cloud infrastructure | ● Implementing/Updating architectural features<br>● Updating source code (maintenance)<br>● Role: DevOps team and competencies: Systems Thinking and Development | ● Management Document<br>● Source Code<br>● Change Log in Repository |

### 5.3. Visual representation of practices in the DevOps lifecycle using Essence

We propose a representation of the DevOps lifecycle, encompassing phases, practices, activity spaces, alphas, activities, roles, competencies, and work products. This representation is designed by Essence elements, providing a standardized notation. This common notation facilitates the understanding and application of practices within each phase of the DevOps lifecycle.

Nowadays, the details of DevOps practices are widely described in texts and books. A visual representation such as Essence will allow us to show the concepts in a clearer and more structured way and the information can be interpreted quickly. In [36] it is explained why graphical representations can be more effective than text for understanding and problem solving, which supports their use in disciplines such as software engineering. Knowing this advantage, the use of Essence in organizations would facilitate communication, transmit ideas more effectively, managing the ambiguity of multiple interpretations of DevOps practices. In addition to that, Essence is allowed to represent at different levels of abstraction and highlight relevant aspects without getting lost in unnecessary detail, which happens more often in textual notations. Essence also provides a standardized notation that facilitates model reuse and understanding across teams, even if they are geographically distributed.

#### 5.3.1. Planning

In this planning phase are involved two practices: product and process management and architecture management, which are requirements management and security by design. These practices are carried out using activities such as documenting requirements, managing the product backlog, and designing the architecture. As

a result, various work products are generated, such as: user stories (to document requirements) and a product backlog (a prioritized list of user stories), requirements models (diagrams and prototypes), and architecture documents or models (which involve the infrastructure aspects of the software product). The main roles in this phase are the DevOps team (development and operations) and the software architect. We show the details of this phase in Figure 5.

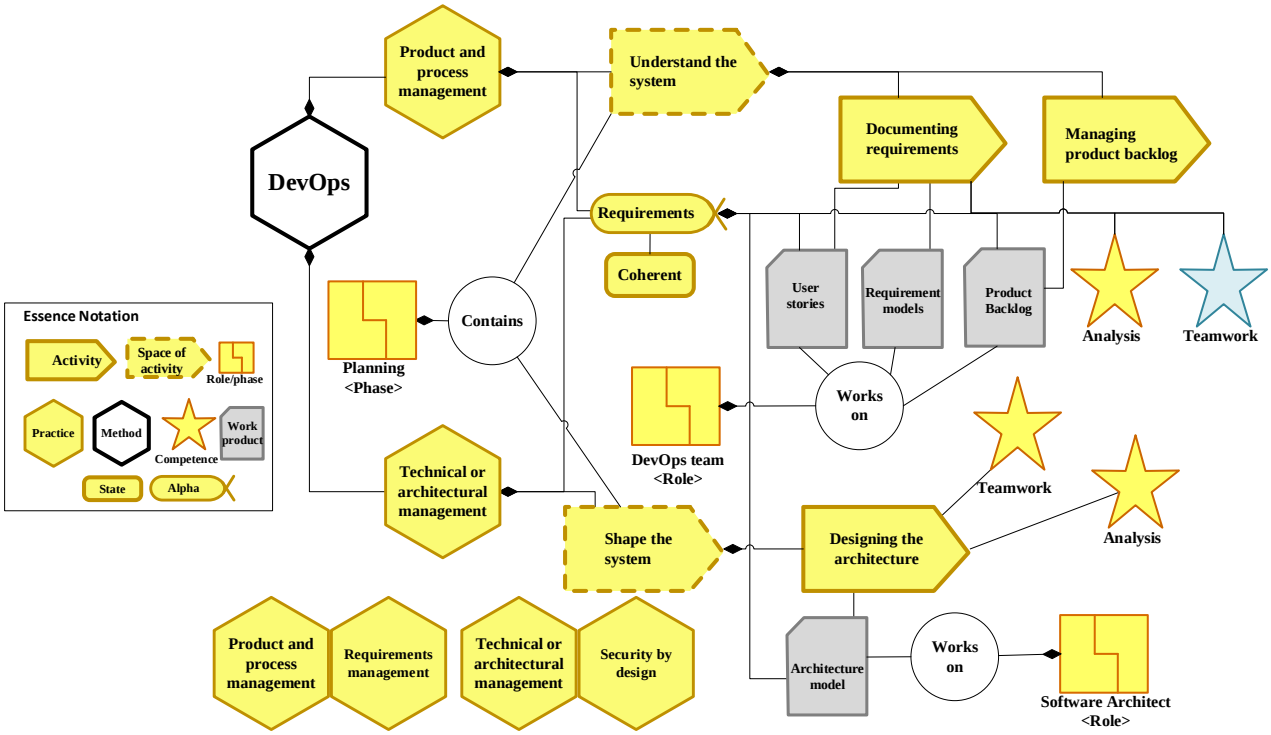



Figure 5. Planning phase. The authors

### 5.3.2. Development

Four practices related to continuous delivery are involved in the development phase such as: hypothesis-driven development, trunk-based development, version control, and configuration management and three practices of architecture management: loosely coupled architecture infrastructure as code, and security as code. These practices are applied in activities such as: creating a pipeline, creating and updating the source code, developing the deliverable, and recording versions and changes in the repository, resulting in the following work products: pipeline (an artifact to document the sequence of automated steps of DevOps lifecycle practices or phases), source code, deliverable (a functional MVP), and change log in the repository (a file that records significant changes made to the source code). The role involved in this phase is the DevOps team. We present the details of this phase in Figure 6.

### 5.3.3. Integration

Four practices related to continuous delivery are involved in the integration phase, such as: continuous automated testing, continuous integration, test data management, and version control. These practices are applied during the testing activities, which can include functional, acceptance, and integration testing, among others. As a result, a work product known as the test execution report is generated to detail the results of the tests conducted to assess the quality status of the software product. The QA team (Quality Assurance) is primarily responsible for this phase. We provide more details about this stage in Figure 7.

### 5.3.4. Monitoring

In this phase are incorporated four practices related to product and process management: requirements management, feedback and continuous improvement, working in small batches, and team experimentation; five practices of lean management: change approval, proactive fault notification, visibility of work, limit work in progress, continuous monitoring and observability; and seven practices of cultural management: generative organizational culture, continuous learning, team collaboration, performance measurement, job satisfaction,

transformational leadership, and DevOps education. These practices are applied in activities such as: tracking teams, processes, and resources and measuring performance, resulting in the creation of various work products. These include a training plan (to track team skills), an event log (to provide feedback on incidents), the progress of the process (to monitor software product development), and a measurement report (to evaluate the performance of a process, software product, and team) and the DevOps team is essential in this phase. We present the details of this phase in Figure 8.

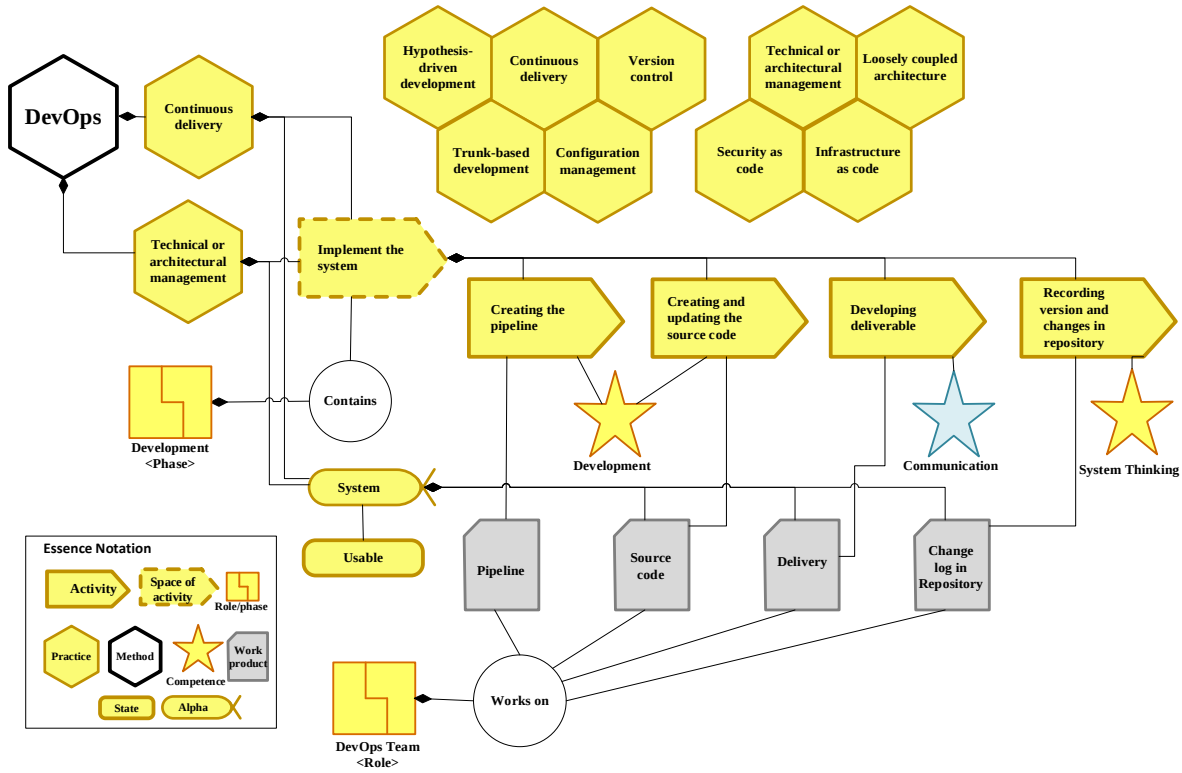



Figure 6. Development phase. The authors

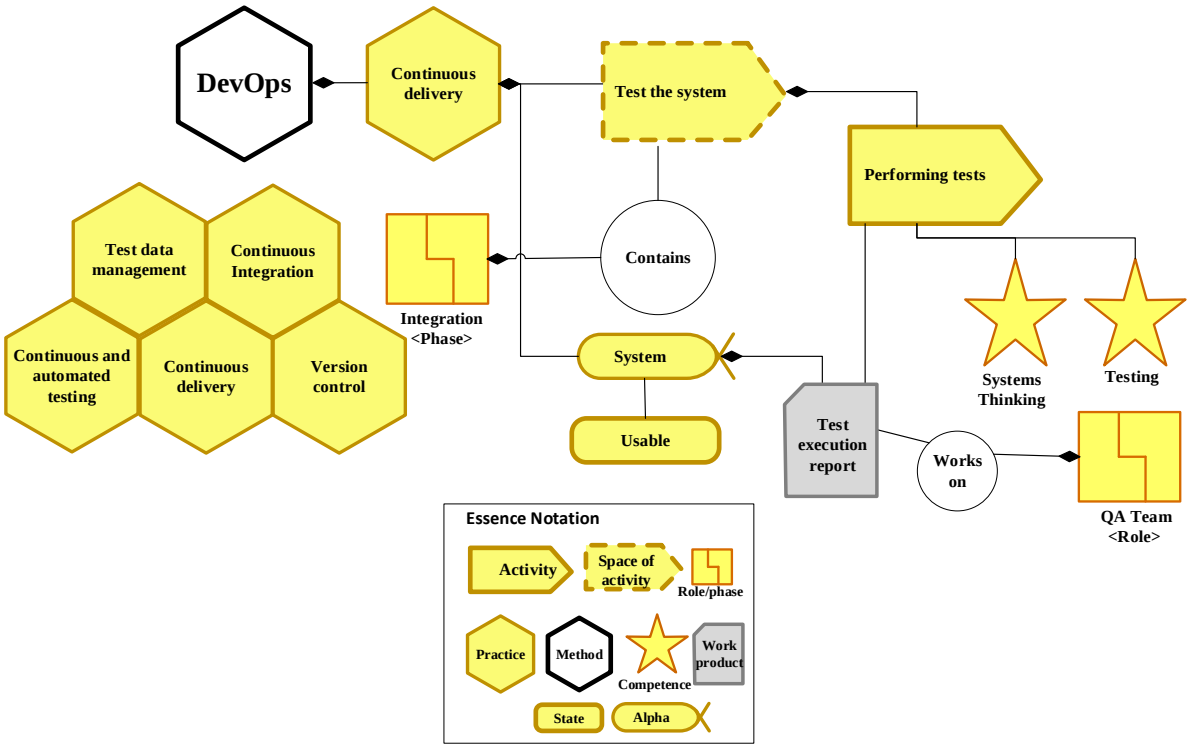



Figure 7. Integration phase. The authors

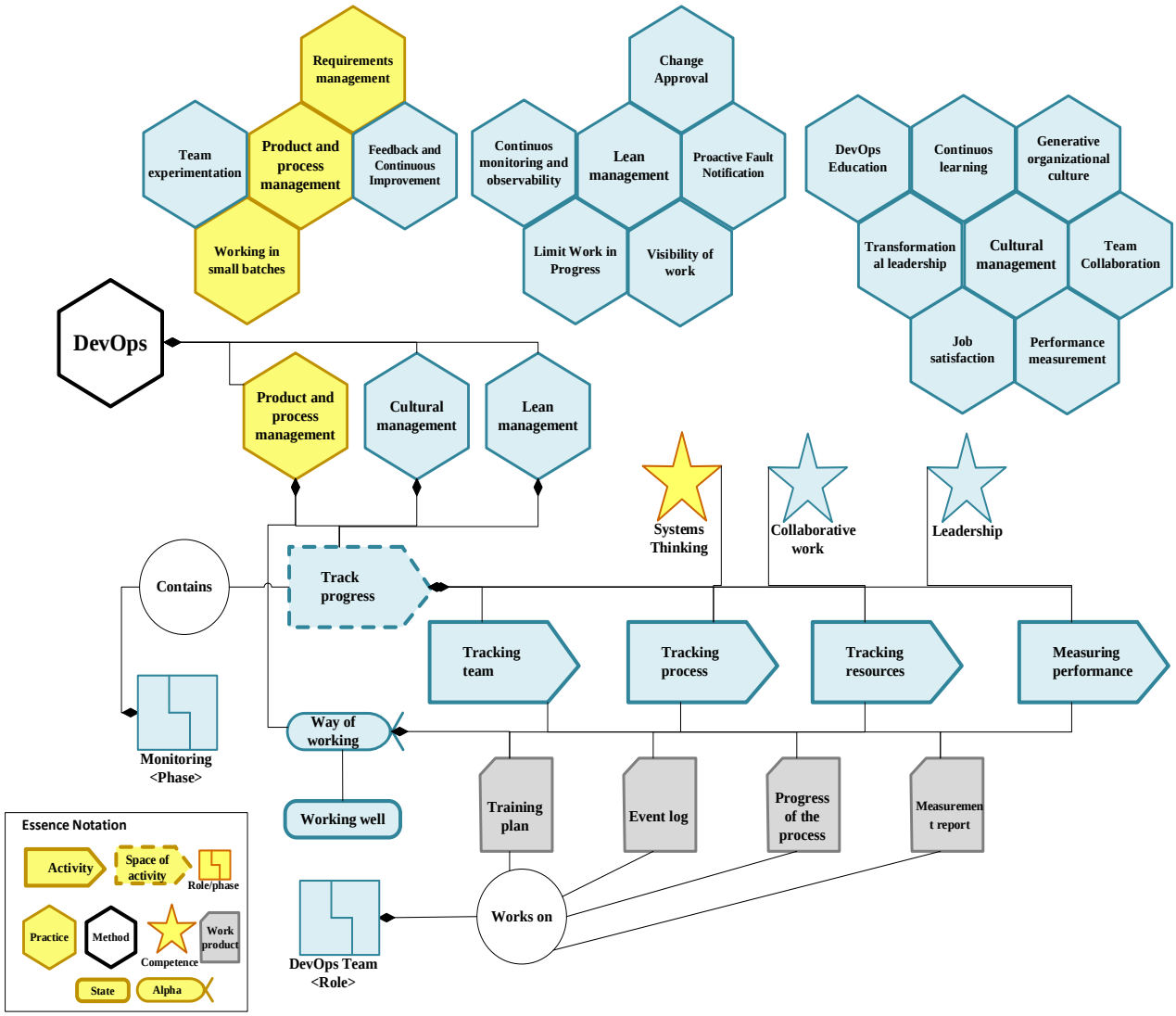



Figure 8. Monitoring phase. The authors

### 5.3.5. Deployment

In this phase there are involved three practices of continuous delivery such as: continuous deployment, configuration management, and version control. Such practices are applied in activities such as configuring the production environment, generating configuration files, and compiling code in the production environment resulting in the following work products: configuration files and deployment scripts (files that contain instructions for executing a sequence) and change log in repository. Additionally, containerization (code packaging) and virtualization (environment creation) practices are integrated into the deployment and the role involved is the DevOps team. We relate the details of this phase in Figure 9.

### 5.3.6. Operation

This phase involves implementing two practices of continuous delivery: version control and cloud infrastructure and three practices of architecture management: security as code, configuration management, and infrastructure as code. The goal is to provide high availability, scalability, persistence, resilience, security, and other architectural features [3]. These activities and practices are carried out with the expectation of obtaining several work products, including an environment management document, which describes how the different environments, the source code, and the version and change log are configured, maintained, and managed in the repository. The DevOps team plays a fundamental role in this phase. We show the details of this phase in Figure 10.

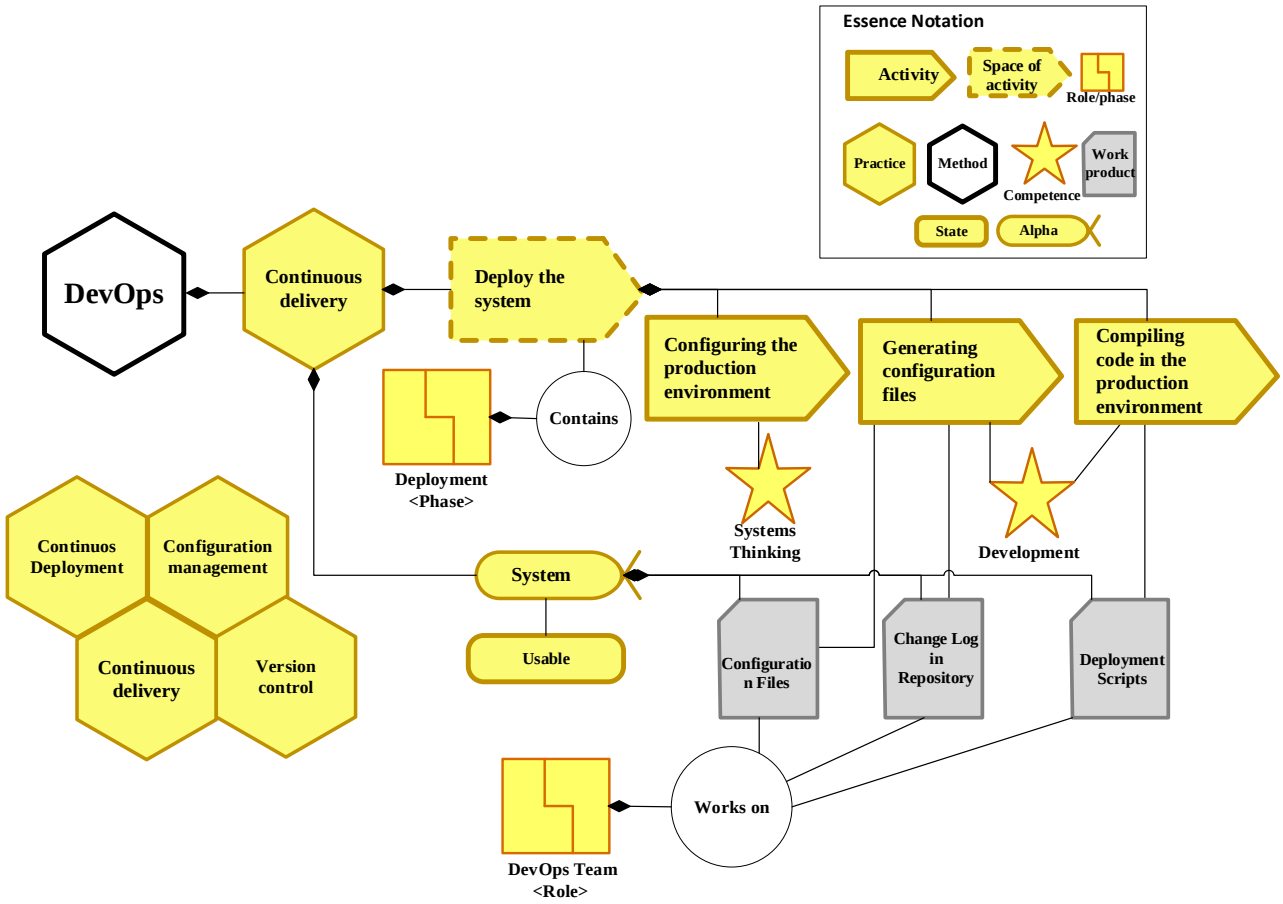



Figure 9. Deployment phase. The authors

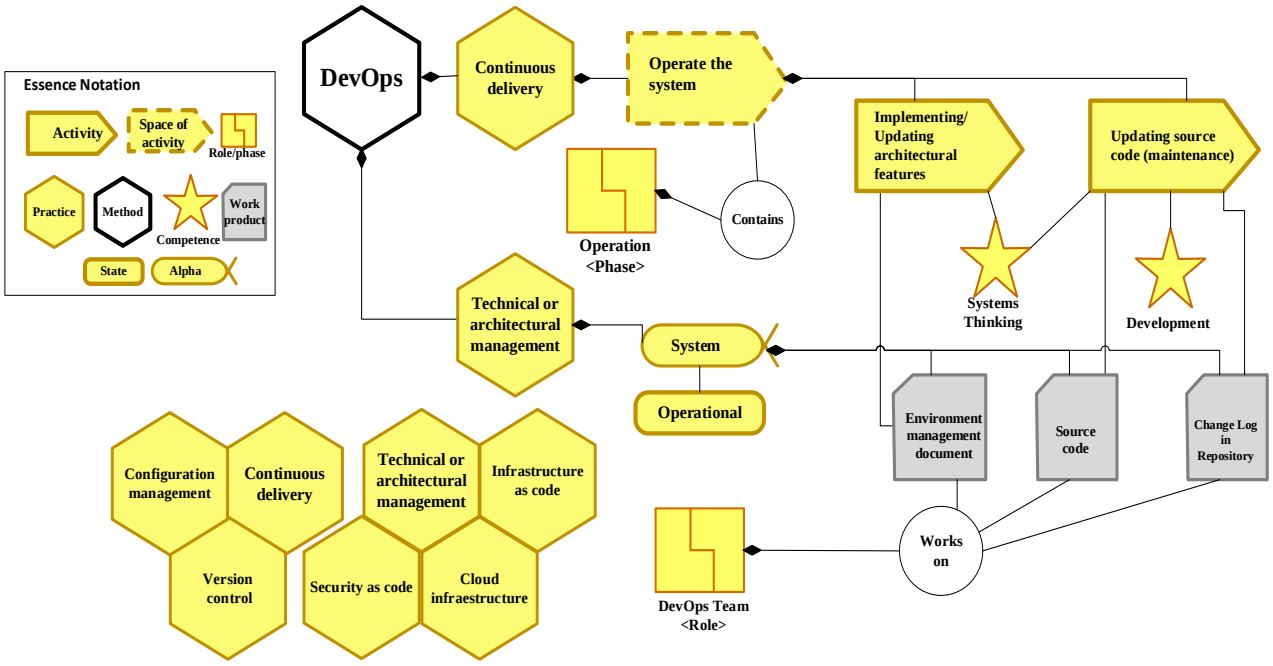



Figure 10. Operation phase. The authors

### 5.3.7. Limitations of the visual representation and the DevOps framework

The Essence standard for visual representation is flexible, but it can be challenging for teams to learn and apply. Teams unfamiliar with it may need training and time to effectively use it, as it introduces a new way of representing practices. The standardized structure, while useful for simplification, can be rigid when adapting or representing complex practices that don't fit neatly into the predefined visual models. Other standards, such as business process model and notation (BPMN), are also used in industrial environments for process

representation. Users are encouraged to use Essence notation together with other diagrams if they find them pertinent for a complete understanding.

The labels and names used to the practices and their elements were found in the literature. However, users can adapt the names of practice labels and their elements if considered pertinent.

A website is under construction to consolidate the elements of the DevOps framework. While it may be challenging to search and understand the specifications of this framework, current references are available at the following link for more details: https://github.com/cesarpardo/DevOpsFramework.

## 6. Validation: evaluation of the metric for the understandability of the DevOps lifecycle representation from Essence

A focus group was carried out as a qualitative research technique, this was based on the steps proposed by Kontio *et al.*[34], [35] the steps were: (i) defining the research problem, (ii) selecting participants, (iii) conducting the focus group session, (iv) data analysis and reporting, (v) improvement actions, (vi) research construction, and (vii) limitations. The focus group method proposed by Kontio *et al.* [30], [31] is an essential tool in software engineering for requirements gathering and analysis, as it encourages the active participation of stakeholders and ensures that the final product aligns with their needs and expectations. One of the main advantages of this method is its ability to identify needs and problems that may not be evident through other techniques, facilitating a deeper and more accurate understanding of the project requirements. Additionally, by promoting consensus among participants, the focus group method helps prioritize requirements effectively, ensuring that project resources are optimally utilized. Open discussion guided by an expert moderator not only enriches the information-gathering process but also strengthens the collaboration and commitment of all involved, resulting in a more robust software product well adapted to its context of use.

### 6.1.1. Defining the research problem

The main objective of this evaluation was to obtain an expert perception of this proposal through group interaction of professionals, in this sense, we sought to collect suggestions, opportunities for improvement, and recognition of the validity of the proposed solution. The focus group gathered the opinions and perceptions of professionals with experience in DevOps, evaluating the proposal in five variables: (i) completeness, (ii) suitability, (iii) applicability in agile approaches, (iv) ease of understanding, and (v) general features to assess the percentage of understandability of the representation (the metric proposed in the mechanism).

The evaluation of these elements allowed us to meticulously structure the execution of the debate protocol, ensuring a clear and coherent guide for the discussion. Additionally, relevant documents were shared with participants so that everyone had the information necessary to contribute in an informed manner. Robust and systematic methods were established to capture and record information, allowing for accurate and complete data collection. Finally, a detailed analysis of the information obtained during the debate was carried out, using advanced coding and categorization techniques to identify emerging patterns and themes. This comprehensive approach facilitated the acquisition of high-quality qualitative data and provided a solid foundation for the interpretation and application of the findings in the context of software development.

### 6.1.2. Selecting participants

In this step, the profiles of the participants and the corresponding selection criteria were carefully defined. The established criteria included possessing students, and professionals with and without knowledge of DevOps, thus selecting forty-tree participants who completely satisfied the required profile. Once selected, they were sent a formal invitation to participate in the focus group, which was positively accepted by all. The date and time of the discussion session were then coordinated, ensuring a three-week window for proper planning and preparation. With the discussion session scheduled, proposal documents were sent to participants for prior review, ensuring that everyone had access to the information necessary for a productive and focused discussion.

### 6.1.3. Conducting the focus group session

This step was carefully organized by a member of the research group who acted as moderator, along with another member who assumed the role of rapporteur. The structure and sequence of the session were planned and communicated to the participants in advance, ensuring that everyone was informed about the development of the activity. During the session, the rapporteur was responsible for carefully recording each participant's observation and comment, using structured note-taking techniques to ensure complete and accurate capture of

information. This systematic recording allowed for rigorous monitoring of the discussions and facilitated subsequent analysis. At the end of the session, participants were asked to complete a questionnaire specifically designed to address the questions presented in Table 5. This questionnaire not only allowed for the collection of additional data and clarification of points discussed during the session but also offered participants the opportunity to reflect and expand on their responses, thus contributing to a deeper and more comprehensive understanding of the topics discussed.

The proposed questionnaire consisted of twelve questions, of which the first six were related to defined variables related to skill participants (name, mail, student type, professional role, filiation, and DevOps expertise) and the last six were related to representation, the last question was open (see Table 5). From the six questions related to representation (see Table 5), the first five questions were evaluated using a 5-point Likert scale, ranging from: (1) Strongly Disagree, (2) Disagree, (3) Neither Agree nor Disagree, (4) Agree, and (5) Strongly Agree. It should be noted that each question was evaluated based on aspects such as clarity, simplicity, neutrality, scope, ambiguity, and coherence between the evaluated variables, among others. In addition, an open question was defined to collect suggestions from the representations, and based on the suggestions, make improvements to the proposal from an open perspective.

Table 5. Research questions applied in the focus group and results of the survey. The authors

| **Understandability** | | **Scale** | | | | | | | | | | **Total %** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **(1) Strongly Disagree** | | **(2) Disagree** | | **(3) Neither Agree nor Disagree** | | **(4) Agree (%)** | | **(5) Strongly Agree (%)** | | |
| **#** | **Questions** | **Q** | **%** | **Q** | **%** | **Q** | **%** | **Q** | **%** | **Q** | **%** | |
| Q1 | Phases of the DevOps lifecycle | 1 | 2.3 | 0 | 0.0 | 7 | 16.3 | 13 | 30.2 | 22 | 51.2 | 100 |
| Q2 | Agile practices involved in each phase | 1 | 2.3 | 4 | 9.3 | 7 | 16.3 | 19 | 44.2 | 12 | 27.9 | 100 |
| Q3 | Activities of process | 1 | 2.3 | 1 | 2.3 | 5 | 11.6 | 20 | 46.6 | 16 | 37.2 | 100 |
| Q4 | Work products in each phase | 2 | 4.6 | 0 | 0.0 | 5 | 14.0 | 22 | 51.2 | 12 | 30.2 | 100 |
| Q5 | Roles and competences | 1 | 2.3 | 4 | 9.3 | 7 | 16.3 | 18 | 41.9 | 13 | 30.2 | 100 |
| Q6 | An open question to collect suggestions for representations | Open question | | | | | | | | | | |

### 6.1.4. Data analysis and reporting

After conducting the focus group, the information obtained during the proposal discussion session was analyzed through the questionnaires completed at the end of the session. This activity was carried out following the strategies defined in the information capture and registration phase. Table 5 shows the count of the participants' responses for each of the questions according to the established scale. Likewise, it presents the percentages for each of the questions for each of the five points of the Likert scale. Figure 11 presents the consolidated results and the distribution of the answers to the questions.

Figure 11 presents the consolidated results and the distribution of the answers to the questions, where it can be seen that the evaluation results predominate: (i) strongly agree (51.16%) and agreement (30.23%) regarding the *phases of the DevOps lifecycle*, (ii) concerning the *agile practices involved in each phase*, predominance of agreement (44.19%) and strong agreement (27.91%), (iii) about the *activities of process*, predominantly agree (46.6%) and strongly agree (37.21%), (iv) regarding the *work products in each phase*, the majority agree (51.2%) and strongly agree (30.23%), and finally; (v) about *roles and competences*, agreement predominates (41.86%) and strong agreement (30.23%). In this sense, it can be concluded that the participants tend to agree with the statements and evaluations of the phases, agile practices, activities, products, and roles in the DevOps cycle. Likewise, and based on the results obtained, most participants express understandability and clarity of the elements in the representation. On the other hand, open question Q6 allowed participants to propose adjustments and improvements to the proposal, some of which will be addressed in the following section.

Visual representation enhances DevOps practices by providing a clear and concise understanding of key software development concepts. This facilitates the integration of agile practices within DevOps by highlighting its essential elements such as activities, work products, roles, and competencies. Moreover, the clarity provided by the visual representation helps to strengthen the effective implementation of DevOps in project development. In addition, and as can be seen from the results of a focus group study conducted, it was possible to observe that participants agreed that visual representation improves the comprehensibility and clarity of the elements in the DevOps lifecycle. Specifically, 51.2% of participants 'strongly agreed' and 30.2% 'agreed' with the clarity of the phases of the DevOps lifecycle.

### 6.1.5. Improvement actions

The results, comments, and opinions of the participants were meticulously analyzed and considered implementing improvement actions in the proposal. This review process allowed us to identify key areas to significantly improve and refine the proposal, identifying several opportunities to improve the understanding and usefulness of the representation, among them we highlight: (i) some participants pointed out the need for a clear explanation of the colors used, which would facilitate the interpretation of the visual content, (ii) regarding the phases of the DevOps cycle, it was suggested that there could be variations in the order of the interaction, so it was recommended to clarify this flexibility, (iii) Other comments highlighted that the names of the DevOps practices are sufficient for their understanding without the need for extensive definitions, (iv) in addition, it was observed that the document is aimed at a diverse audience, from individuals without prior knowledge to experts on the subject, for example. Therefore, it is recommended that each figure was a table of conventions. Therefore, it was added for providing a clear reference to the reader about the type of content, whether it is a deliverable, an activity, or another element, (v) finally, it was emphasized the importance of clarifying the five pillars of DevOps and the processes involved in each, as well as their meaning, to ensure a complete and accurate understanding of these fundamental concepts.

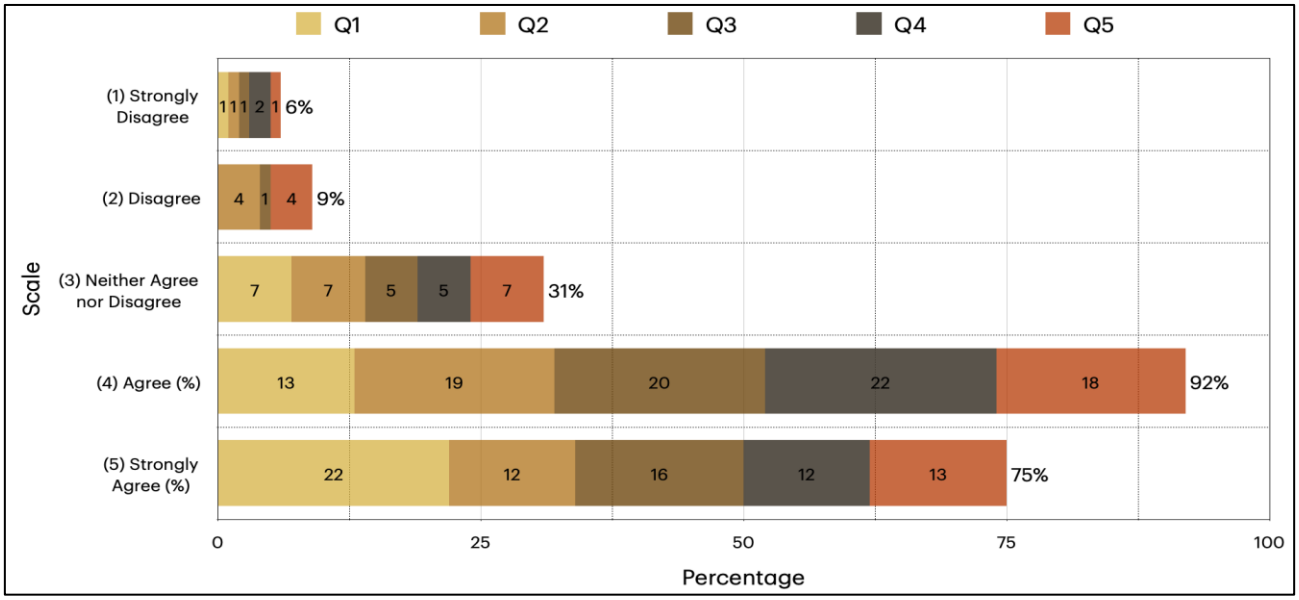



Figure 11. Results of understandability. The authors

### 6.1.6. Research construction

To ensure that the research construct in this study was valid and aligned with our objectives, three fundamental techniques were implemented. First, the established content and format for the focus group session were rigorously defined and maintained. This structural consistency was crucial to ensure that the data collected was comparable and relevant, thus facilitating a coherent and uniform analysis. Special attention was paid to every detail of the session design, from the questions posed to the dynamics of interaction between participants, ensuring a structure that promoted the obtaining of valuable and meaningful information. Second, to minimize instrumentation errors, it was decided to audio record all discussion sessions. This measure allowed for an accurate and detailed review of all participants' interventions and comments, avoiding loss of information and ensuring that even the most subtle nuances of the conversation were captured. Transcription of these recordings provided a solid foundation for analysis, allowing researchers to revisit the discussions as many times as necessary to ensure accuracy in the interpretation of the data. Finally, to minimize possible biases in the interpretation of the results, a person external to the research was incorporated to review all the interpretations made during the analysis. This external review was essential to provide a critical and unbiased perspective, which contributed significantly to the objectivity and accuracy of the study's conclusions. The involvement of this external reviewer not only provided an impartial perspective but also reinforced the credibility of the findings, ensuring that the interpretations were robust and well-founded.

Together, these techniques not only ensured the validity of the research construct but also improved the quality and reliability of the data collected and the conclusions derived, providing a solid foundation for future research, replicate, extend, and practical applications in the field studied.

### 6.1.7. Limitations and bias

During the focus group, some limitations were identified, and solutions were implemented to effectively address them. First, there was an initial challenge for the moderator in controlling the discussion style of the less active participants, which was quickly resolved through the intervention of more experienced researchers. Second, invalid data occurred caused by incorrect responses from the participants, which were mitigated through the active intervention of the moderator, who facilitated the discussion constructively. Third, to mitigate the risk of limited knowledge and understanding, individuals with similar experiences were selected, prior reading material was provided, and complex topics were broken down into more manageable chunks.

To reduce selection bias and encourage equitable participation, several measures were taken. The risk of not adequately representing the diversity of relevant opinions or perspectives was recognized and was addressed by ensuring the inclusion of a variety of relevant perspectives and experiences. Additionally, to prevent some participants from having a disproportionate influence on the discussion, rules were established that promoted an environment in which all participants felt comfortable sharing their opinions. To ensure that the moderator did not influence the direction of the discussion, an impartial moderator was used to guide the discussion neutrally. Finally, the difficulty of generalizing the results was considered due to the specific and small nature of the samples in the focus groups. To address this limitation, the focus group was complemented with a perception survey applied individually, which allowed for obtaining a more complete and generalizable vision of the research object.

## 7. Conclusions and future work

DevOps framework and practice representation in the DevOps lifecycle by using Essence contributes to conceptual understandability. This is concluded by obtaining approval among respondents on key aspects such as roles, competencies, work products, and development cycle activities. DevOps provides teams in productive environments with a clear and concise understanding of key software development concepts. Furthermore, it facilitates the understanding of agile practices because they are effectively integrated within the context of DevOps, highlighting their essential elements: activities, work products, roles, and competencies. Similarly, an opportunity is identified, as the proposal improves clarity on how the practices support the overall process and can help strengthen the effective implementation of DevOps in project development.

On the other hand, by adopting Essence the standard for visualizing practices in DevOps, organizations can establish a common and consistent framework for software development, facilitating collaboration and communication among teams. The representation carried out, in Essence, can be adapted to different contexts and software development needs, allowing organizations to customize and adjust their processes in DevOps as necessary. Essence as a graphical notation not only simplifies the representation of DevOps practices but also improves communication, decision-making, and efficiency in software development, making it a fundamental tool for software engineering in organizations.

Essence can be used as a standard for visualizing practices in DevOps. In addition to this, this proposal can be used in conjunction with the DevOps Framework for explaining and detailing the processes and practices of the reference model, so that individuals can assess the degree or percentage of implementation of practices in DevOps individually and/or collectively, at both a general and granular level. This leads to the description of one of the limitations of this work, reaching the evaluation level, which can be seen as future work since the purpose was to showcase Essence as an alternative for representing and tracing the mentioned practices. This representation of DevOps using Essence is an original outcome of the research and was empirically validated through focus groups during the evaluation session described in Section 6.

Finally, as future work, it is suggested to conduct case studies applied in real environments related to software development where the phases, practices, and activities related to the representation are applied. Additionally, these practices can be related to work in sustainable software development, as the adoption of DevOps practices can contribute to reducing resource waste, improving code quality, optimizing resource usage, and facilitating quick adaptation to changes in the operational environment and business requirements.

## Acknowledgments

This paper is a product of the project *Human Sustainability Model in Software Development Products with Limited Technological Infrastructure* in the 2023 Women in Science program sponsored by Minciencias, Icetex,

L'Oréal-Unesco, and the executing university EAFIT. Professor César Pardo is grateful for the contribution of the Universidad del Cauca, where he currently works as a full professor.

**References**

[1] P. Dubey and R. Raja, A Beginners Guide to Amazon Web Services. United States: CRC Press, 2023.

[2] 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*, pp. 1–11, 2016.

[3] N. Forsgren, J. Humble, and G. Kim, Accelerate. The science of Lean Software and DevOps. Bulding and Scaling High Performing Technology Organizations. Portland, Oregon, 2018.

[4] A. Felipe and F. Núñez, "DevOps: un vistazo rápido," *Ciencia Huasteca Boletín Científico de la Escuela Superior de Huejutla*, vol. 10, no. 19, p. 35–40, 2022.

[5] C. Durango-Vanegas, C. Zapata-Rueda, and C. Zapata-Jaramillo, "Representación en el Núcleo de la Esencia de *Semat* de las Competencias de un Equipo de Desarrollo de Software," *Información tecnológica*, vol. 30, no. 4, p. 217–226, 2019.

[6] I. Jacobson, P. Ng, P. McMahon, I. Spence, and S. Lidman, "The essence of software engineering," *Communications of the ACM*, vol. 55, no. 12, p. 42, 2012.

[7] "Essence - Kernel and Language for Software Engineering Methods Version 1.2," Presented by Object Management Group® (OMG®), 2018.

[8] J. Bird, "*DevOpsSec,"* Presented by O'Reilly Media Inc., 2016.

[9] "2023 Final Report - State of DevOps," Presented by Deveps Research & Assessment (DORA), 2023.

[10] J. Guerrero, D, Modelo de referencia para la adopción de DevOps en empresas de desarrollo de software, Universidad del Cauca, 2021.

[11] N. Kim, G. Humble, J. Debois, P. Willis, and J. Forsgren, The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. Portland, Oregon: IT Revolution, 2021.

[12] A. Martínez, E. Suescún, P. Noreña, L. González, and C. Pardo, "Implementación de prácticas DevOps en un Sistema de Mainframe Legado," *Investigación e Innovación en Ingenierías*, vol. 10, no. 2, p. 129–146, 2022.

[13] P. Noreña, E. Suescún, and J. Mejía, "Desarrollo dirigido por hipótesis en productos de software sostenible: Un mapeo sistemático," in *Jornadas Argentinas de Informática (JAIIO 53) de la Conferencia Latinoamericana de Informática (CLEI)*, 2024.

[14] C. Orozco-Garcés, C. Pardo-Calvache, and E. Suescún-Monsalve, "Metrics Model to Complement the Evaluation of DevOps in Software Companies," *Revista Facultad de Ingeniería*, vol. 31, no. 62, p. e14766, 2022.

[15] M. Akbar, S. Rafi, A. Alsanad, S. F. Qadri, A. Alsanad, and A. Alothaim, "Toward Successful DevOps: A Decision-Making Framework," *IEEE Access*, vol. 10, p. 51343–51362, 2022.

[16] M. Tanzil, M. Sarker, G. Uddin, and A. Iqbal, "A mixed method study of DevOps challenges," *Information and Software Technology*, vol. 161, p. 107244, 2023.

[17] L. Welder, G. Pinto, and R. Bonifácio, "Adopting DevOps in the real world: A theory, a model, and a case study," *Journal of Systems and Software*, vol. 157, p. 1–16, 2019.

[18] N. Azad and S. Hyrynsalmi, "DevOps critical success factors — A systematic literature review,"

*Information and Software Technology*, vol. 157, 2023.

[19] M. Gasparaite, K. Naudziunaite, and S. Ragaisis, "Systematic literature review of DevOps models," *Communications in Computer and Information Science*, vol. 1266, p. 184–198, 2020.

[20] R. Grande, A. Vizcaíno, and F. García, "Is it worth adopting DevOps practices in Global Software Engineering? Possible challenges and benefits," *Computer Standards and Interfaces*, vol. 87, 2024.

[21] A. Jha *et al.*, "From theory to practice: Understanding DevOps culture and mindset," *Cogent Engineering*, vol. 10, no. 1, 9. 1 From theory to practice: Understanding DevOps culture and mindset 31, 2023.

[22] A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," *Computer Science Review*, vol. 38, p. 100308, 2020.

[23] J. Guerrero, C. Pardo, and C. Orozco, "DevOps Ontology - An ontology to support the understanding of DevOps in the academy and the software industry," *Periodicals of Engineering and Natural Sciences*, vol. 11, no. 2, p. 207, 2023.

[24] R. Rajapakse, M. Zahedi, M. Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: A systematic review," *Information and Software Technology*, vol. 141, p. 106700, 2021.

[25] R. Srinivasan, S. Eppinger, and N. Joglekar, "The structure of Devops in product-service system development," in *Proceedings of the International Conference on Engineering Design*, pp. 3111–3120, 2019.

[26] "Using SEMAT and Essence at Fujitsu UK," InfoQ. [Online]. Available: https://www.infoq.com/articles/semat-essence-fujitsu/. [Accessed: Mar. 25, 2025].

[27] M. Sletholt, J. Hannay, D. Pfahl, and H. Langtangen, "What Do We Know about Scientific Software Development's Agile Practices?" *Computing in Science & Engineering*, vol. 14, no. 2, p. 24–37, 2012.

[28] M. Darrin and W. Devereux, "The Agile Manifesto, design thinking and systems engineering," in *Annual IEEE International Systems Conference*, Montreal, QC, Canada, pp. 1–5, 2017.

[29] T. Dingsøyr, T. Fægri, T. Dybå, B. Haugset, and Y. Lindsjørn, "Team Performance in Software Development: Research Results versus Agile Principles," *IEEE Software*, vol. 33, no. 4, p. 106–110, 2016.

[30] C. Pardo, Guerrero, J., and Monsalve, E. DevOps model in practice: Applying a novel reference model to support and encourage the adoption of DevOps in a software development company as case study. *Periodicals of Engineering and Natural Sciences*, vol. 10, no.3, p. 221–235, 2022.

[31] C. Orozco, C., Pardo, K. Zúñiga, and S. Certuche, "Proceso para fomentar y apoyar la adopción de DevOps en PyMEs de software" *Revista Científica*, vol. 45, p. 422–437, 2022.

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

[33] C. Orozco-Garcés, C. Pardo-Calvache, and Y. Salazar-Mondragón, "What is There About DevOps Assessment? A Systematic Mapping.," Revista Facultad de Ingeniería, vol. 31, no. 59, p. e13896, 2022.

[34] J. Kontio, J. Bragge, and L. Lehtola, "The focus group method as an empirical tool in software engineering," *Guide to Advanced Empirical Software Engineering*, Springer London, pp. 93–116, 2008.

[35] J. Kontio, L. Lehtola, and J. Bragge, "Using the focus group method in software engineering: obtaining practitioner and user experiences," in *International Symposium on Empirical Software Engineering*, Redondo Beach, USA: IEEE Press, pp. 271–280, 2004.

[36] J. H. Larkin and H. A. Simon, “Why a diagram is (sometimes) worth ten thousand words,” Cognitive Science, vol. 11, no. 1, pp. 65–100, 1987.