Revista Facultad de Ingeniería
ISSN: 0121-1129
ISSN: 2357-5328
Universidad Pedagógica y Tecnológica de Colombia

Pastrana-Pardo, Manuel-Alejandro; Ordóñez-Erazo, Hugo-Armando; Cobos-Lozada, Carlos-Alberto

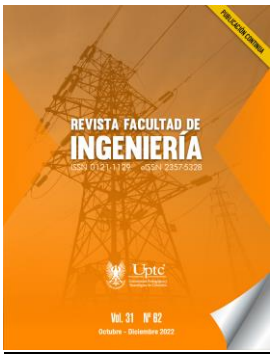
Process Model Represented in BPMN for Guiding the Implementation of Software Development Practices in Very Small Companies Harmonizing DEVOPS and SCRUM

Revista Facultad de Ingeniería, vol. 31, no. 62, e209, 2022, October-December
Universidad Pedagógica y Tecnológica de Colombia

DOI: https://doi.org/10.19053/01211129.v31.n62.2022.15207

Available in: https://www.redalyc.org/articulo.oa?id=413974254010

Revista Facultad de Ingeniería
Journal Homepage: https://revistas.uptc.edu.co/index.php/ingenieria



Process Model Represented in BPMN for Guiding the Implementation of Software Development Practices in Very Small Companies Harmonizing DEVOPS and SCRUM

Manuel-Alejandro Pastrana-Pardo[1]
Hugo-Armando Ordóñez-Erazo[2]
Carlos-Alberto Cobos-Lozada[3]

Received: September 6, 2022 **Accepted:** December 27, 2022 **Published:** December 31, 2022

Citation: M.-A. Pastrana-Pardo, H.-A. Ordóñez-Erazo, C.-A. Cobos-Lozada, "Process Model Represented in BPMN for Guiding the Implementation of Software Development Practices in Very Small Companies Harmonizing DEVOPS and SCRUM," *Revista Facultad de Ingeniería*, vol. 31 (62), e15207, 2022. https://doi.org/10.19053/01211129.v31.n62.2022.15207

[1] Ph. D. (c) Institución Universitaria Antonio José Camacho (Cali-Valle del Cauca, Colombia). mapastrana@admon.uniajc.edu.co. ORCID: 0000-0002-6506-0659
[2] Ph. D. Universidad del Cauca (Popayán-Valle, Colombia). hugoordonez@unicauca.edu.co. ORCID: 0000-0002-3465-5617
[3] Ph. D. Universidad del Cauca (Popayán-Valle, Colombia). ORCID: 0000-0002-6263-1911

Abstract

The business process model is a graphic representation mechanism that helps improve the understanding of a context, the steps undertaken, and the validations and business rules that are part of its universe. This article proposes an implementation model of practices for software development based on DevOps suggestions and how these might be executed within Scrum by the Scrum Development Team (SDT). Present a practice implementation model that integrates DevOps suggestions to be executed by a scrum development team (SDT). The practices for software development based on DevOps were identified. The moment in which the information provided is helpful for the team's continuous improvement within SCRUM was determined. With the practices identified, modeling the general process of implementing practices using BPMN was conducted, followed by detailed modeling. Finally, experts executed the evaluation of the detailed process model. A 12-question survey was implemented to understand the business process model created for implementing practices. This instrument was then made available to experts in the field to obtain feedback on what has been done. The results obtained are promising. The set of practices suggested by DevOps and its integration in Scrum allows for establishing a preventive quality approach for the best development of software products. Using business process models represented by BPMN allows companies to understand and adopt the proposed practices quickly.

Keywords: business process modeling; DevOps; SCRUM; software engineering; software quality assurance.

Modelo de procesos representado en BPMN para guiar la implememtacion de prácticas de desarrollo de software en empresas muy pequeñas armonizando DEVOPS y SCRUM

Resumen

Los modelos de procesos de negocio son un mecanismo de representación gráfica que ayudan a mejorar la comprensión que se tiene sobre un contexto, el conjunto de pasos que se llevan a cabo dentro de él, las validaciones y reglas de negocio

que hacen parte de su universo. Utilizando esto el presente artículo propone un modelo de implementación de prácticas para desarrollo de software basado en las sugerencias de DevOps y como estas pueden ser ejecutadas dentro de SCRUM por parte del Equipo de desarrollo SCRUM (SDT por sus siglas en ingles). El trabajo tiene como objetivo exponer un modelo de implementación de prácticas que integre las sugerencias de DevOps para ser ejecutadas por un equipo de desarrollo en scrum (SDT). Se identifican prácticas para desarrollo de software basado en DevOps. Se determina el momento donde la información aportada es útil para la mejora continua del equipo dentro de SCRUM. Con las practicas identificadas se realiza el modelamiento del proceso general de implementación de prácticas utilizando BPMN, seguido del modelamiento detallado. Por último, expertos evaluaron el modelo detallado de procesos. Se elabora una encuesta de 12 preguntas sobre la comprensión de los modelos de procesos de negocio creados para la implementación de las prácticas. Este instrumento es puesto a la disposición de expertos en el tema para obtener una retroalimentación sobre lo realizado. Los resultados obtenidos son prometedores. El conjunto de prácticas sugeridas por DevOps y su integración en SCRUM permiten establecer un enfoque de calidad preventiva para el mejor desarrollo de productos software. El uso de modelos de procesos de negocio representados con BPMN permite a las empresas una fácil comprensión y adopción de las prácticas propuestas.

Palabras clave: aseguramiento de la Calidad de Software; DevOps; ingeniería de software; modelado de procesos de negocio; SCRUM.

Modelo de processo representado em BPMN para orientar a implementação de práticas de desenvolvimento de software em microempresas harmonizando DEVOPS e SCRUM

Resumo

Os modelos de processos de negócios são mecanismos de representação gráfica que ajudam a melhorar a compreensão de um contexto, o conjunto de etapas que são realizadas dentro dele, as validações e regras de negócios que fazem parte de

seu universo. A partir disso, este artigo propõe um modelo de implementação para práticas de desenvolvimento de software baseado em sugestões DevOps e como estas podem ser executadas dentro do SCRUM pelo SCRUM Development Team (SDT). O trabalho visa expor um modelo de implementação prática que integra sugestões de DevOps a serem executadas por um time de desenvolvimento scrum (SDT). São identificadas práticas de desenvolvimento de software baseadas em DevOps. É determinado o momento em que as informações fornecidas são úteis para a melhoria contínua da equipe dentro do SCRUM. Com as práticas identificadas, é realizada a modelagem do processo geral de implementação das práticas em BPMN, seguida da modelagem detalhada. Finalmente, os especialistas avaliaram o modelo de processo detalhado. É elaborada uma pesquisa de 12 perguntas sobre o entendimento dos modelos de processos de negócios criados para a implementação das práticas. Este instrumento é disponibilizado a especialistas da área para obter feedback sobre o que foi feito. Os resultados obtidos são promissores. O conjunto de práticas sugeridas pelo DevOps e sua integração no SCRUM permitem estabelecer uma abordagem de qualidade preventiva para o melhor desenvolvimento de produtos de software. A utilização de modelos de processos de negócios representados com BPMN permite que as empresas compreendam e adotem facilmente as práticas propostas.

Palavras-chave: DevOps; engenharia de software; garantia de qualidade de software; modelagem de processos de negócios; SCRUM.

I. INTRODUCTION

Software development companies are constantly looking to optimize their production processes to improve profitability based on increased quality [1]. Companies should therefore have mechanisms of control that provide timely information within the development process to take the necessary corrective measures before commissioning [2].

Implementation of this control depends on the set of practices for software development prioritized by the company to have the most significant impact, taking account of the metrics and information these practices provide to the company to support its continuous improvement [3]. A critical factor in selecting such practices is the size of these companies because it determines a set of organizational culture characteristics that facilitate or limit their adoption.

According to [4], companies with up to 25 employees are classified as **very small**, between 26 and 50 as **small**, 51 to 250 as **medium,** and those with more than 250 employees as **large**.

The previous means that results from the evolution of the projects lead to extensive guarantee periods, where the errors from not having quality controls in place in conducting the project are assumed, errors which should have been identified long before the deployment so as not to generate extra costs, breach of commitments, schedule delays, or work overload. This means that the results at the financial level in the VSEs limit their cash flow, hindering processes of continuous improvement that allow exploring quality-enhancing alternatives. Added to this, by having more extensive guarantee processes, typically, people who belong to this type of company perform multiple functions in various roles, constantly overburdening themselves and leaving little time to adopt new ways of working or supporting a company improvement process.

Authors such as [5] have identified that most companies that make up the software development industry are very small entities (VSEs). Due to their small size, these companies start with an empirical software development process based on the experience of their founders and early collaborators who are part of the development

teams, lacking practices around requirements analysis, software design, development, quality, and even deployment.

As an alternative to these problems, VSEs resort to frameworks such as Scrum and DevOps that facilitate adaptation to change, adopt practices oriented towards preventive quality, and generate information that allows continuous retrospective analysis. Although these alternatives positively impact the industry, a fundamental problem is associated with the agile philosophy that structures it: there are no standards or guides that instruct what practices to implement and how to implement them. Instead, agile frameworks provide suggestions but leave companies free to explore which ones are appropriate according to their organizational culture, type of customer, and ways of working.

VSEs require alternatives that facilitate the adoption process of these frameworks and guide them as to what they can implement, allowing the creation of a phased adoption plan. Business Process (BP) models (visual representations that facilitate a single interpretation) are thus presented as a viable alternative to represent a guide that enables adopting practices. This allows an overall transversal vision of the processes conducted in an organization, identifying the variables, business rules, and requirements to be solved, historically proven in industrial process modeling and requirements analysis [1]. In addition, BPs allow us to understand the step-by-step that must be executed within a process and even the sequence of sub-processes linked for shaping a more complex process. All of this makes the BP an appropriate tool with which every one of the members of the development team can know graphically what steps are being performed at a given moment in the company.

Based on the above, this paper presents an implementation process model of practices recommended by DevOps that are common in VSEs, and that can be applied as a complement to Scrum. The rest of the paper is organized as follows: Section 2 describes the motivation scenario; Section 3 lays out the methodology; Section 4 contains the results of a survey of academic and industry experts regarding their perception of the proposed models; and finally, Section 5 provides the conclusions and future work expected to be conducted in the short term.

The difficulties in projects developed by VSEs, according to [6], point to the identification of controls associated with practices for software development that organize and centralize the management of their changes, measure the impact they have on the project, perform traceability of their history, generate early alerts, and ensure quality through their automation. This set of practices continuously puts forward information that must be analyzed by the development team at a given moment, taking corrective measures during the building phase rather than afterward to increase the quality of the deliverables. DevOps puts forward a set of suggestions per the above that may be useful for improving the quality associated with the development process and that can be adopted within Scrum to use this information as an opportunity for improvement, facilitating learning and the adoption of these by the team.

A single interpretation model (graphical model) is required to present the guide proposed in this work, capable of simplifying the understanding of the set of practices to be adopted, the gradual step-by-step of their implementation, and the holistic vision of how some practices complement each other to give form to the quality controls that would transform the VSE development process. Given the advantages of business process models, these were selected to represent the guide.

According to [7], BP models allow the detailed identification of the practices at a more abstract level - when they would be used manually or automatically, what information they should generate and what actions they can trigger in the development team. At their most detailed level, they provide the step-by-step to achieving an adequate connection.

II. METHODOLOGY

The defined process is made up of the following phases: I) Identification of essential practices for software development based on DevOps, II) Linking of the selected practices with Scrum, III) General modeling of the implementation process of practices using BP, IV) Detailed process modeling using BP, and V) Evaluation of the detailed process model by experts.

A. Identification of Essential Practices for Software Development Based on DevOps

According to [8], DevOps proposes a collaboration between the area of software development with the area of infrastructure and operations. This collaboration supports all the systems and services of companies at the hardware level, seeking to reduce reprocessing and improve the organizational culture. Its objective is to adopt practices for software development that continuously generate information, allowing development teams to analyze it, learn from it, and gradually improve.

The previous suggests that an environment of quality (preferably preventive) that constantly measures the evolution of the project to be developed must be generated, featuring a set of quality controls automatically synchronized and always available to the interested parties, as indicated in [9]. DevOps's impact on the industry is well-renowned, as the academic community is interested in continuing to delve into this topic and generate paths that facilitate its adoption [10].

To identify which practices can provide a solution to the needs of the industry, [11] presents a review of the adoption of DevOps to achieve a continuous delivery (CD) process. This process consists of adopting a set of practices that automatically ensures a step-by-step for a successful deployment, with traceability of changes and control over them. The suggested practices indicate that the source code must be versioned to ensure that all the information is always organized in a central repository available to the entire team (collective code ownership). Likewise, in [12], it is indicated that as the project evolves and the changes are versioned, they must be continuously integrated to identify if syntactic errors in the code are being uploaded to the versioner or if it is correct. This quality control is known as continuous integration (CI). To complement the two previous practices, in [13], they suggest implementing continuous deployment (CD) so that once the two previous quality filters are approved, the deployable unit is automatically placed in the quality or production environment and thus able to ensure continuous delivery. This was implemented using tools via scripts called pipelines by [14].

Likewise, in [15], it is indicated that the starting point of an effective preventive quality process starts with versioning and triggers an automatic process of continuous

integration, which inspects if the evolution of the code syntactically has not undergone negative changes (syntax errors) that prevent generating the deployable unit and allow continuous deployment to execute successfully. Additionally, the code should be standardized so that all team members develop the project the same way, thus making it easier to maintain its evolution and support. The adoption of international programming standards facilitates the understanding of the code, its readability, and the understanding of structuring. This practice can be reviewed through manual or automatic code inspections carried out by Static Code Analyzers (SCA) that not only check if the international standard is met but also add value to the measurement by reviewing aspects such as security, detecting common vulnerabilities in development practices based on the OWASP top 10 and informing them so that corrective measures are taken complying with the minimum suggested by [16]. Likewise, this tool makes it possible to measure what the industry knows as a code smell, understood as lousy development practices that prompt negative impacts within projects. Some examples of code smell include defining variables or importing libraries (libs) without using them, high cyclomatic complexity, and improper handling of constructors. The generation of this type of problem within a project causes the need to refactor the code, involving extra effort and time, known as technical debt (also measured within the SCA).

Additionally, in [15], it is indicated that the same tool is capable of reviewing unit test coverage, which suggests that this practice ought to be adopted by teams if they want to increase the benefits obtained. Unit tests are conducted to check the quality of the developed code, guaranteeing its correctness (that the functionality does what it should), in addition to checking the cases where incorrect data is entered and how it handles exceptions that control failures and prevent the software from crashing due to mishandling. The more detailed the test is in its correctness and error handling, the greater its coverage will be.

Also, in [15], it is indicated that if the source code has been versioned and the quality controls of the CI and the SCA have been approved, the automated process performs the CD, and previously recorded functional test scripts can be executed

automatically. This ensures that thorough automatic reviews of everything developed so far (functional and regression tests) can be done using tools. The previous ensures a complete cycle of continuous, high-quality deliveries with various controls executed during the development period, constantly providing feedback to the team for continuous improvement.

All these practices work correctly if the teams adopt preventive quality-oriented thinking in their organizational culture, ensuring an understanding of the benefits of its adoption to the development process and themselves [6]. Table 1 summarizes the identified practices and their benefits.

Table 1. Recommended practices and their benefits.

Recommended practice	Benefit
Code standard	It increased the scalability and maintainability of the code.
Versioning	Organization of information, collective ownership of the code, availability, traceability of changes, and recovery from failures.
Continuous integration (CI)	Syntactic control of the code, identifying if the versioned changes allow or prevent the generation of the deployable unit.
Continuous deployment (CD)	Automation of the deployment process once all the process quality filters have been approved.
Unit tests (UT)	Minimum seal of development quality. Checks correctness and exception handling, always guaranteeing the operability of the software.
UT coverage measurement	Determines how exhaustive the unit tests have been, detecting opportunities for improvement in the quality of the process.
Static code analysis (SCA)	Allows detection of security vulnerabilities, structural errors, duplicate code blocks, and technical debt from the measurement of code smell; inspects whether it complies with the international code standard for the programming language and measures PU coverage.
Automated functional tests	Ensures that once the application has been deployed, automatic functional and regression tests can be conducted, identifying whether what is expected in the project is met.

B. Connection of the Selected Practices with SCRUM

According to [17], the success of correctly implementing Scrum suggestions consists in understanding the value associated with prevention and the constant generation of information that allows the team to know what they are doing well and what they should improve. Scrum handles three roles: the Product Owner (PO), who maintains the vision of the product and ensures compliance with the client's interests; the Scrum Master (SM), who supports the team in adopting Scrum events, practices,

and dynamics; and the Scrum Development Team (SDT), in charge of preparing the project and ensuring that the results have the highest possible level of quality when it is put into production environments. The transfer of information between these roles is essential to ensure the fulfillment of commitments, the high quality of the deliverables, and the retrospective analysis that allows constant improvement during the project.

Likewise, in [17], it is indicated that Scrum has a set of important events to be completed during a project. The sprint, an event that occurs throughout the execution, allows the work to be broken down into a standard time measurement to agree on periodic reviews that will approve the job done or generate improvements based on what has been detected. At the beginning of each sprint, it is necessary to create an event called the sprint planning meeting, where the steps to be implemented during that time are defined as well as how they will do it, responding to the functional needs of the requirements analysis, collected in the product backlog and architectural needs compiled in the architecture document. In this meeting, the SM and the SDT actively define the breakdown of development activities that will complete the commitments, as well as the quality filters and practices that the execution of the project entails.

The PO will be available for consultation but is not essential for the execution of the event. After the above, the SDT begins codifying the solution, ensuring the use of the agreed practices, and verifying the tools that allow the increased quality to be continuously measured. Here, the line between development and quality disappears, becoming a single phase of the process. It is necessary to highlight that the work teams from the preparation sprint (Sprint 0), before development Sprint 1, must create the versioning repository, ensure its access, and agree on the versioning policies (inclusion, download, reversion, and integration of changes). Additionally, other practices such as CI, UT, CD, and SCA suggested by DevOps must be linked to the versioning tool and define how the functional tests will be conducted. The SM supports the review of these metrics constantly so that the entire team can learn and improve during the evolution of each sprint. The specific Scrum event that supports

this is the daily meeting, where team members answer questions such as *What work was done the day before? What work is currently being done?* and *What difficulties have you had?* Additionally, before the meeting, the SDT members and the SM must review their metrics, determining the current technical debt to avoid increasing it. This is only possible if they are aware of the recurring unacceptable programming practices in their work that cause a violation of the code standard (code smell) or avoid the generation of the application. Likewise, they must ensure with unit tests that each code unit built carries an implicit quality seal in the process.

Once the above is completed, deployment proceeds, which can be automated by CD tools, eliminating manual intervention from the process and increasing the success factor in deployments. The results are presented to the end users in the sprint review, where the agreed functionalities for the sprint are evaluated. Although end users do not know the measurements of the tools that support the practices implemented by the team, they benefit from the high quality due to the reduced possibility of failures and inconsistencies. Unit testing and functional test automation are critical to the success of this event. Finally, the team meets without the customer and holds a sprint retrospective event [17]. Here the team, using hindsight techniques and supported by metrics, can evaluate what they did well within the sprint and the possibilities for improvement. The above is summarized in Table 2.

Table 2. The relationship between Scrum and DevOps and its impact.

Scrum events	DevOps recommended practice	Impact on the development process
Sprint planning meeting	● Identification of programming practices to use.	● Create development policies for the team.
	● Design of the archetype.	● Determine the quality attributes and design patterns that guide the architecture and the test scenarios that must be faced.
Sprint 0	● Implementation of the archetype for the development baseline. ● Creation of UT guide in the archetype. ● Versioning configuration. ● Version baseline of development. ● Implementation of CI. ● Implementation of CD. ● Implementation of SCA.	● Centralize information and change management with historical traceability and failure recovery. ● Create an automated preventive quality environment that constantly reviews the evolution of the project and allows the

Scrum events	DevOps recommended practice	Impact on the development process
	● Implementation of a tool for functional tests.	team to learn and improve sprint by sprint.
Development Sprint	● Development is done with unit tests (UT). ● Use of versioning. ● Use of CI. ● Use of CD. ● USE of the SCA. ● Automation of functional tests.	● Identify vulnerabilities, code smell, duplicate blocks, technical debt, UT coverage, correctness, and exception handling to implement improvements that improve quality.
Daily Scrum meeting	● Review the metrics provided by the tools supporting the suggested practices.	● Reduce technical debt, determine the status of commitments and react to problems within the sprint.
Sprint review	● CD to ensure sprint review. ● Unit tests with coverage greater than 80% to ensure high quality. ● Automation of functional tests	● Ensure that the development is to the expectations of end users. ● Opportunity for continuous improvement.
Sprint Retrospective	● Analysis of the information generated for continuous improvement.	● Opportunity for continuous improvement.

C. General Model of the Process of Practices Implementation Using BP

Authors such as [15] and [18] agree that the first steps in practices recommended by DevOps for development teams point to versioning, integration, and continuous deployment, where versioning is the starting point of the process. To ensure the collective ownership of the code and its standardization, all members of the SDT work connected to a source code repository (versioner) where they centralize their changes, generating traceability and the possibility of immediate failure recovery. For this practice, it is suggested by [19] that only the last utterly stable version of the project is kept in the master branch and that it is put into production (what in other models is known as release or tag) so that each developer will have their own branch to build the functionalities that they must develop (which in other models such as that of Git are called features). Likewise, this same branch manages errors that have not yet reached production and are detected during the quality cycle (what in Git is called a hotfix).

After achieving the definition of completed (functionality built and tested by the developer), the developer requests to join their changes to an integrations branch where everyone's work is synchronized and mixed. The request is made through a practice called a pull request (PR), which notifies the team that someone wants to upload a change, and a developer other than the one submitting the request does a manual code inspection to ensure the change is ready to be uploaded and does not pose a risk to building the deployable unit. If the request is approved, the changes are integrated, and everyone is notified via email that a change is available (although there are other alternatives, such as slack chats). It is important to note that if there is a conflict between the local version of the developer and the integrations branch that contains the stable version, the developer must resolve them locally first before being able to request integration via the PR. The versioning model described above is shown in Fig. 1.

Since the integration branch is the synchronization point of the code, the quality controls of the other practices should be triggered from here when the PR request is approved. Additionally, in [15], they suggest that it is advisable to include the execution of the unit tests (UT) within the CI before building the deployable unit. This is done to ensure that the developers have tested every code unit. If someone from the work team has uploaded a change affecting other code units without verifying its impact, then the UT will be automatically run during the CI, reflecting the current situation and notifying the work team of this. Likewise, in [19], they indicate that if it has been possible to perform the versioning properly and the CI with its unit tests, then the CD is possible. The SCA can be included to inspect the quality of the built software.

This review corresponds to determining aspects such as vulnerabilities, duplicate code blocks, and code smell, and even measuring the coverage of unit tests since they must not only verify correctness (that the functionality responds to the desired behavior for which it was built) but it must also handle exceptions properly always to ensure the operation of the application. Therefore, it is essential to use unit tests to review the coverage if a significant quality result is to be obtained, which is allowed by the SCA practice.

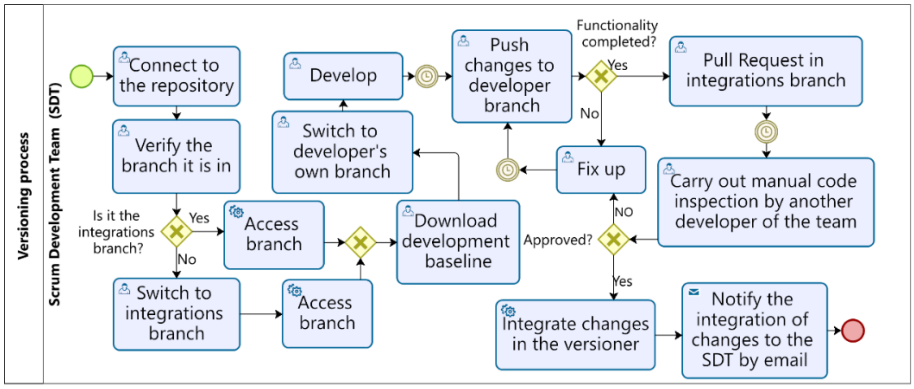


Fig. 1. Versioning model.

Moreover, triggering an automated testing process after the CD practice is advisable. These work in tools that allow the recording of scripts with the step-by-step of the desired functional tests, identifying the navigation flow, required fields, data that must be entered, and the operation to be performed on the screen. Likewise, they allow the recording of tests on exception handling, capturing the necessary steps to enter incorrect information, and identifying the messages that must be returned to control exceptions. This model is summarized in Fig. 2.

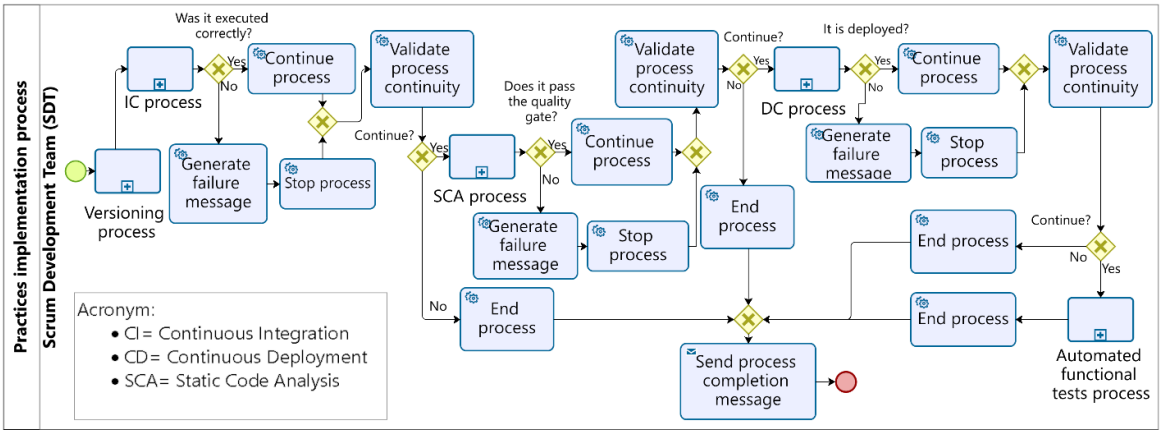


Fig. 2. General model of the process of practice implementation.

D. Detailed Model of the Process of Practice Implementation Using BP

As a prerequisite for their adoption, the team must reach a consensus on which practices will be adopted. In [20], it is recommended to adopt versioning, CI, and CD

initially. Once the members master these practices in one or two projects, the company can consider including other complementary practices such as unit tests, static code analysis, and the automation of functional tests described in the model in Fig. 3.

To ensure that the developer starts working with the latest updated version that synchronizes all the changes of all the members (the version in the integrations branch), the branch of the repository in which the developer who has connected is found must be validated. If this branch does not correspond to the integrations branch, that switch must be made. Once the developer is in the integrations branch, the latest updated version is downloaded and immediately switched to the developer's own branch to begin from there to build the functionalities that have been committed in the sprint until reaching the definition of completed (developed and tested).

When the developer builds the functionality, they must also build the unit test associated with their code to ensure correctness and proper exception handling. These changes can be saved in the appropriate branch with the desired periodicity. Once functionality has been completed, a request can be made through the practice of the PR for the inclusion of its changes in the integrations branch. Once the request is created, it waits for a team member other than the one requesting it to inspect the code and determine if it does not affect the project negatively. If the request is not approved, the developer is notified so that it can make the respective changes and request the inclusion of its development in integrations again. If the request is accepted, the changes are unified within the integration version, making it available to all development team members. This also triggers the CI, where a two-step script is executed. The first step indicates that all unit tests should be executed to ensure that no changes have functionally affected other parts of the code. If the tests fail, the process stops, notifying everyone. If the tests pass, the next step is to execute the commands required to create the deployable unit. If it is impossible to generate it, the process stops, and everyone is notified. Otherwise, the appropriate script that allows the CD is executed.

This script takes the credentials and access path of the application server where the software will be put into operation to connect with it and deliver the deployable unit generated in the previous step. For this, the file is placed in a specific path within the server, and the operating system commands required for deployment are executed. If there is a failure at this point, a message is generated notifying the entire team that the process was not completed and the cause is that the CD could not be executed. On the other hand, if the process is completed successfully, the entire team is notified, and the self-test scripts built to that point previously by the team are executed. This point presents a favorable aspect within Scrum because sprint by sprint not only functionally checks the newly developed functionalities but can also do automatic regression tests on those previously developed, ensuring that the deliverable increments have not affected what has already been done.

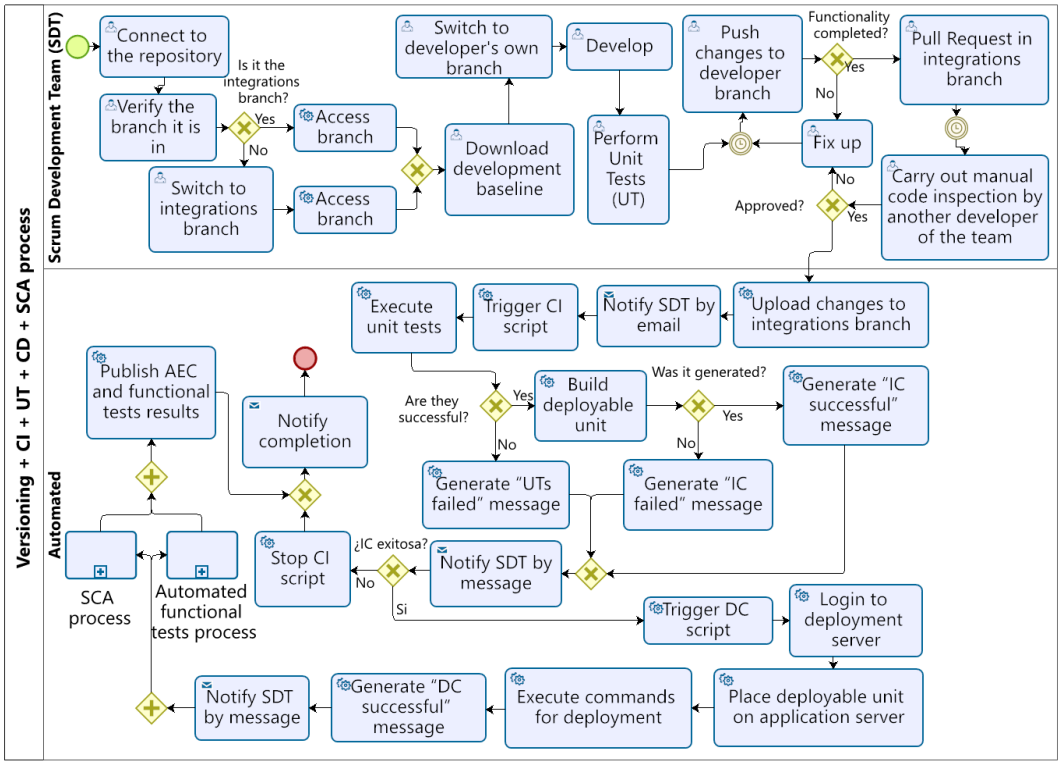


Fig. 3. Detailed model.

Finally, the SCA can be executed in parallel with the previous step. Various aspects can be measured within this point, most easily seen in Fig. 4.

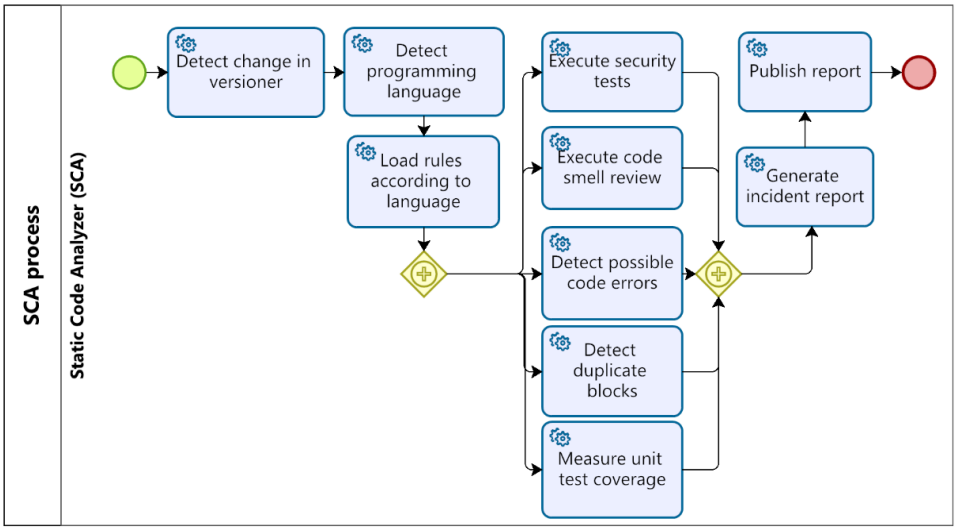


Fig. 4. SCA process.

E. Evaluation of the Detailed Process Model by Experts

For the evaluation and feedback on the "process models for the implementation of practices in software development", a survey with the following questions was used:

Position held?

If you work in Software Engineering, select the type of work you do in this area.

1. Is the versioning model understandable?

1.1. Do you have any suggestions for improvement?

2. Do you consider the versioning model replicable?
3. Does it make use of versioning as presented by the model?

3.1. Other. Which activities do you think should be added?

4. From your perspective, what help or vision do the BPMN models give you?
5. Is the general model for implementing practices for software development understandable?
6. Do you think the general model is replicable?

6.1. Do you think something else could be added?

7. Does it make use of continuous integration (CI) as presented by the model?
8. Did the detailed model help you better understand the set of practices involved?
9. Does it make use of continuous deployment (CD) as outlined in the model?
10. Do you consider the model useful for the industry?
11. Do you consider the information provided by the SCA useful?
12. Do you have any suggestions for the proposed models?

These questions were published through a Google form and disseminated to 79 professional experts in the area who work in academia and industry. The results are presented in the following section.

III. Results

Initially, the question: Position held? Allows the respondents' classification, determining whether they are researchers, teachers, students, or professionals in Software Engineering (SE). Of the total respondents (79), 2.6% (2)were researchers, 7.7% (6) were teachers, 23.1% (18) were students, and 66.6% (53) were SE professionals.

SE professionals were further classified in the survey using the option: If you work in Software engineering, select the type of work in this area. Of the total respondents (79), 49% (39) were senior developers, 33% (26) were junior developers, and 9% (7) were semi-senior developers. The previous reflects a significant interest of most experienced developers in the industry on these issues. However, junior developers have become interested in using practices as a much more efficient way of working, with less rework due to quality controls. An additional 3% of those surveyed, two people, were analysts. The remaining positions each represented a percentage of 1% of the total respondents, that is, one person. Table 3 below presents the summary of the previous measurement.

Table 3. Percentage of participation by SE professionals by position held.

SD	JD	SSD	A	QA	AS.	TL	PM	C
49%	33%	9%	3%	1%	1%	1%	1%	1%

*Senior Developer (SD), Junior Developer (JD), Semi-Senior Developer (SSD), Analyst (A), Quality Assurer (QA), Software Architect (SA), Technical Leader (TL), Project Manager (PM) Consultant (C).

Regarding Question 1, Is the versioning model understandable? 96.2% (76) of participants indicated that the model was understandable, and 3.8% (3) that it was not.

Additionally, through Question 1.1, Do you have any suggestions for improvement? The opinions of the respondents were collected to optimize the versioning process model. The first relevant suggestion was that the name of the first task should be changed from "Connecting to versioner" to "Connecting to repository" to separate the tool's connection from the repositories it may have and to give it greater clarity. This change was applied to the process model after receiving the suggestion. The second recommendation was to adjust the task of accessing the branch that seems repeated and would be better presented differently or unified. Therefore, normalization was done in that part of the flow and adjusted according to the recommendations.

Something important to highlight in the comments is that there was a strong tendency to directly associate the version process model with the Git tool, its practices, and its version model. This means that Git is a widely used tool in the market and has generated a work culture for several respondents. Based on the above, it was identified that the model is designed only for custom software development projects but not for the maintenance and scaling of products where it is necessary to consider the different versions released in production for one or more clients (releases or tags), and that must be stored in the versioner. Additionally, another recommendation was associated with the fact that when the business process model is presented, it should be indicated if it is feature branches (model proposed by Git), trunk-based development (model widely used by SVN), or another process of the team (such as that presented in this work).

Regarding Question 2, Do you consider the versioning model replicable? 86.6% (70) of those surveyed indicated that the model was replicable, and 11.4% (9) suggested otherwise. Likewise, in Question 3, it is evaluated if it is used, Does it make use of versioning as presented by the model? where 56.4% (45) indicated yes, while 43.6% (34) indicated no. This can be seen in Fig. 5.

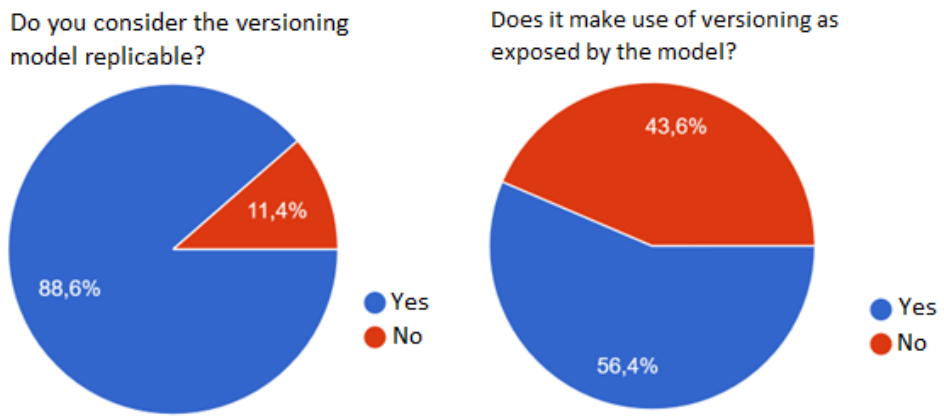


Fig. 5. Percentages of acceptance of the versioning model.

As a complement to the above, the participants were presented with option 3.1 Other. Which activities do you think should be added? The following feedback was obtained: "The model lacks post-dev activities that include maintenance and growth of the applications," and "The use of emails can be invasive, and it would be good to evaluate chat alternatives such as slack." Furthermore, it was recommended to "evaluate cases such as those proposed by GitHub for managing features, hotfixes, bugs, etc." and "include a versioning of the versions put into production."

Question 4 was intended to determine the effectiveness of the business process models made with the BPMN notation, From your perspective, what help or vision do the BPMN models give you? For 100% of the respondents, it facilitates understanding the proposed processes and makes it possible to obtain an ordered sequence and generate concise guidelines for adopting the suggested practices. Additionally, it avoids ambiguous interpretations since the model, when represented graphically, only has one understanding, eliminating assumptions within the process.

Regarding the general model that integrates all the proposed practices and that was presented to the participants within the survey, Question 5 asked: Is the general model of implementation of practices for software development understandable? In response, 88.5% (70) agreed it was understandable, while 11.5% (9) indicated it was not. Then, for Question 6, Do you consider that the general model is replicable?

84.4% (67) felt the model was replicable, while the remaining 15.6% (12) believed it was not. This is reflected in Fig. 6.

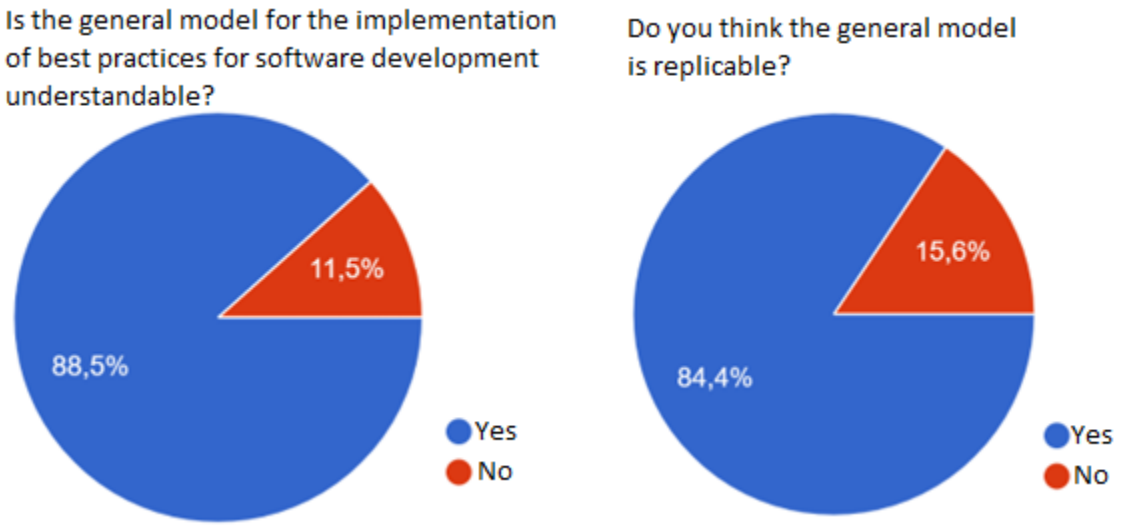


Fig. 6. Percentages of acceptance of the general model.

As a complement to question 6 of the survey, Question 6.1 was included: Do you think something else could be added? Among the main recommendations is the inclusion of a note describing the acronyms. Moreover, It was suggested that it would be necessary to complement the model with non-functional tests focused on security, performance, and component integration, and include an element of review of connection availability and consumption of APIs ("health-checks"). These validators reveal whether what is going to be consumed is available or not. If the availability query indicates no connection, the CD is not made, interrupting the process known to have failed. Instead, an alert is generated, and the current production or test version continues until the connection problems are solved.

Concerning Question 7, Do you use continuous integration (CI) as presented by the model? 56.4% (45) of respondents indicated Yes, while 43.6% (34) stated No. Additionally, for Question 8, Did the detailed model help you better understand the set of practices involved? Again, there was an excellent understanding associated with process diagrams that detail practices for development and their combined use

in a quality process. Here, 92.3% (73) of the total participants answered Yes, while 7.7% (6) responded No.

Question 9, Does it make use of continuous deployment (CD) as outlined in the model? evaluates the use of CD, for which 50.6% of the respondents (41) answered affirmatively, and the remaining 49.4% (38) indicated they did not do it that way.

Regarding the usefulness of the proposal, Question 10 was formulated: Do you consider the model useful for the industry? Of those surveyed, 72.2% (61) replied that they thought the model was useful, 19% (15) said maybe, and the remaining 3.8% (3) indicated that it was not.

The perception of the usefulness of the information provided by the Static Code Analysis model was also evaluated through Question 11, Do you consider the information provided by the SCA useful? A significant result was obtained, with 93.7% (74) in favor, while the remaining 6.3% (5) indicated that they did not.

Finally, to obtain feedback, Question 12 was posed: Do you have any suggestions for the proposed models? The first suggestion was that the proposed model be oriented toward custom software development, leaving out maintenance, growth, and product support. Likewise, mobile and serverless applications are left out, so it was recommended to expand the proposal on these points or generate new recommendations for these cases.

IV. Conclusions

The paper proposes a detailed model that allows the unifying of the practices presented by DevOps of versioning, CI, UT, CD, SCA, and automated functional tests, as part of an axis of preventive quality that constantly increases the quality of the development of a project during a sprint. Furthermore, these practices generate relevant information that connects with Scrum events, allowing continuous learning to lead teams to sprint-by-sprint quality improvement of both the product and the development process.

Additionally, the business process models have a graphic and holistic presentation that allows all team members to understand the quality process and adopt the

practices regardless of the supporting tools. They likewise represent the transversality of the processes conducted in the project. Finally, evaluation with experts makes it possible to demonstrate that the inclusion of these practices is relevant for the industry to have models that help guide understanding and implementation.

The most significant interest in adopting practices according to the characterization of the participants came from senior developers, who recognize the benefits of including these practices within the work process in an organization. Doing so generates filters and quality controls that allow taking preventive, non-corrective measures, reducing possible rework after a developed project reaches production deployment and is put into operation.

It was identified, furthermore, that several improvement comments are directly associated with Git-based tools (GitHub, GitLab, Bitbucket, etc.), so it can be noted that developers do not tend to go for a general model that can then be adapted to the company, but for a specific model associated with the tool. This creates a significant opportunity for this research proposal. There is a good understanding and acceptance of the proposed process models. However, applying the suggestions to improve the proposal further and solidifying it through a guide that allows following the procedure and independently adopting the tools is recommended.

In future work, it is recommended to identify configurations of tools, both in the cloud and on-site, that allow the implementation of the suggested practices by the proposed process models. In addition, process models that support products already built and have maintenance, support, and continuous growth must be proposed.

ACKNOWLEDGMENTS

The authors express their gratitude to the Institución Universitaria Antonio José Camacho for participating in the research project entitled *"Guide for the implementation of harmonized practices for the development of software based on SCRUM and DevOps in very small companies"*. Likewise, to the vice-rector for research office of Universidad del Cauca for the support provided to the group of researchers in the execution of this project.

AUTHORS' CONTRIBUTION

Manuel-Alejandro Pastrana-Pardo: Conceptualization, Methodology, Resources, Writing - original draft.

Hugo-Armando Ordoñez-Erazo: Conceptualization, Methodology, Resources, Writing - original draft, Supervision, Project administration.

Carlos-Alberto Cobos-Lozado: Supervision, Writing - review and editing.

REFERENCES

[1] S. Martinez-Fernandez et al., "Continuously Assessing and Improving Software Quality With Software Analytics Tools: A Case Study," *IEEE Access*, vol. 7, pp. 68219–68239, 2019. https://doi.org/10.1109/ACCESS.2019.2917403

[2] L. E. Lwakatare et al., "DevOps in practice: A multiple case study of five companies," *Information and Software Technology*, vol. 114, pp. 217–230, 2019. https://doi.org/10.1016/j.infsof.2019.06.010

[3] T. Laukkarinen, K. Kuusinen, T. Mikkonen, "Regulated software meets DevOps," *Information and Software Technology*, vol. 97, pp. 176–178, 2018. https://doi.org/10.1016/j.infsof.2018.01.011

[4] A. D. Robinson, "Very small entities (VSE); The final systems engineering (SE) frontier," in *12th Annual IEEE International Systems Conference*, 2018, pp. 1–4. https://doi.org/10.1109/SYSCON.2018.8369570

[5] M. Pastrana, H. Ordoñez, C. Cobos, "ISO 29110 en Colombia: de la teoría a la práctica - ISO 29110 in Colombia: from theory to practice," *Guillermo de Ockham: Revista Científica*, vol. 17, no. 2, pp. 71–80, 2019.

[6] A. Hemon, B. Lyonnet, F. Rowe, B. Fitzgerald, "From Agile to DevOps: Smart Skills and Collaborations," *Information Systems Frontiers*, vol. 22, no. 4, pp. 927–945, 2020. https://doi.org/10.1007/s10796-019-09905-1

[7] M. Pastrana, H. Ordóñez, A. Ordonez, L. H. Thom, L. Merchan, "Optimization of the Inception Deck Technique for Eliciting Requirements in SCRUM Through Business Process Models," *Lecture Notes in Business Information Processing*, vol. 308, pp. 649–655, 2018. https://doi.org/10.1007/978-3-319-74030-0_52

[8] R. Jabbari, N. bin Ali, K. Petersen, B. Tanveer, "What is DevOps? A Systematic Mapping Study on Definitions and Practices," in *Proceedings of the Scientific Workshop of XP*, 2016. https://doi.org/10.1145/2962695.2962707

[9] N. Tomas, J. Li, H. Huang, "An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps," in *International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2019, pp. 1–8. https://doi.org/10.1109/CyberSecPODS.2019.8884935

[10] Y. Wang, M. V. Mäntylä, S. Demeyer, K. Wiklund, S. Eldh, T. Kairi, "Software test automation maturity: A survey of the state of the practice," in *Proceedings of the 15th International Conference on Software Technologies*, 2020, pp. 27–38.

[11] M. Z. Toh, S. Sahibuddin, R. A. Bakar, "A Review on DevOps Adoption in Continuous Delivery Process," in *International Conference on Software Engineering & Computer Systems and 4th International*

Conference on Computational Science and Information Management, 2021, pp. 98–103. https://doi.org/10.1109/ICSECS52883.2021.00025

[12] S. Vadapalli, *DEVOPS: continuous delivery, integration, and deployment with DEVOPS*. Packt Publishing, 2018.

[13] S. Vadapalli, *DevOps: Continuous Delivery, Integration, and Deployment with DevOps Dive into the core DevOps strategies*. 2018. https://www.mendeley.com/catalogue/6e5717c5-64d8-3c7c-8183-a659c7018957/?utm_source=desktop&utm_medium=1.19.8&utm_campaign=open_catalog&userDocumentId=%7Bd0423fe3-94bc-4497-8289-f037a01fd183%7D

[14] V. Ivanov, K. Smolander, "Implementation of a DevOps Pipeline for Serverless Applications," *Lecture Notes in Computer Science*, vol. 11271 LNCS, pp. 48–64, 2018. https://doi.org/10.1007/978-3-030-03673-7_4

[15] A. Manuel, P. Pastrana, H. A. Ordoñez Erazo, C. A. Cobos Lozada, "Documenting and implementing DevOps good practices with test automation and continuous deployment tools through software refinement," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 9, no. 4, pp. 854–863, 2021. https://doi.org/10.21533/PEN.V9I4.2239

[16] Q.-M. Dai, R.-F. Mao, H. Huang, G.-P. Rong, H.-F. Shen, D. Shao, "DevSecOps: Exploring Practices of Realizing Continuous Security in DevOps," *Journal of Software*, vol. 32, no. 10, pp. 3014–3035, 2021. https://doi.org/10.13328/j.cnki.jos.006276

[17] K. Schwaber, J. Sutherland, *La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego*, 2020. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

[18] N. Railic, M. Savic, "Architecting Continuous Integration and Continuous Deployment for Microservice Architecture," in *20th International Symposium INFOTEH-JAHORINA*, 2021. https://doi.org/10.1109/INFOTEH51037.2021.9400696

[19] M. A. Pastrana Pardo, H. A. Ordóñez Erazo, C. A. Cobos Lozada, "Approach to the Practices of Software Development Based on DevOps and SCRUM Used in Very Small Entities," *Revista Facultad de Ingeniería*, vol. 31, no. 61, pp. e14828, 2022. https://doi.org/10.19053/01211129.V31.N61.2022.14828

[20] J. Díaz, R. Almaraz, J. Pérez, J. Garbajosa, "DevOps in practice," in *Proceedings of the 19th International Conference on Agile Software Development: Companion*, 2018, pp. 1–3. https://doi.org/10.1145/3234152.3234199