Guidelines for Data Engineering Documentation in a DevDocOps Approach

Stephany Mendes Oliveira
stephanymendes@estudante.ufscar.br
Federal University of São Carlos
Sorocaba, SP, Brazil

Daniel Lucrédio
daniel.lucredio@ufscar.br
Federal University of São Carlos
São Carlos, SP, Brazil

ABSTRACT

The software development process has been studied since the beginning of technological evolution. Development practices have evolved, requiring processes capable of supporting intensive work, paving the way for agile methodologies. With the growing need for continuous integration (CI) and continuous deployment/delivery (CD), new data architectures have emerged, allowing for scalable, maintainable, and reusable environments, collectively known as DevOps (Development + Operations). In this context, the DevDocOps approach integrates continuous documentation into the software development lifecycle. However, little has been published regarding the benefits of this approach. To address this, an empirical study was conducted, applying findings from the literature to a real development environment by integrating continuous documentation into the data engineering development lifecycle. Based on feedback from developers and technical lead, the results highlight the importance of technical documentation in an agile development environment and demonstrate how automating this process can improve the quality and efficiency of software deliveries.

CCS CONCEPTS

• **Software and its engineering → Automatic programming**; **Software configuration management and version control systems**.

KEYWORDS

Documentation Automation, continuous deployment (CD), data engineering, technical documentation, DevDocOps

1 INTRODUCTION

DevOps, a term that unites concepts related to development (dev) with operation (ops), emerged as a set of practices that focus on reducing the time between commits and deployment into production, through team integration and automated tasks [2]. Two important DevOps concepts are Continuous Integration (CI) and Continuous Deployment (CD). These activities are becoming a growing need, given the benefits they bring in terms of agility, quality and confidence in the development life cycle [8].

In today's era, in which the volume of data is growing exponentially, the automation of data documentation is also emerging as a crucial need. This approach minimizes the manual work required to update existing documentation. Automation makes it easier for data teams to focus on larger-scale projects and ensure that documentation is constantly updated in parallel with changes in the data. Standardized, automated documentation not only improves confidence in the data, but also adapts and scales with the growth of the data and the company, eliminating the need for extensive revisions or constant iterations.

But automation alone is not enough. As it has become evident in DevOps, automation must be integrated into a coordinated process that allows different stakeholders to effectively cooperate, otherwise its benefits fail to reach full potential. This is where the DevDocOps approach comes into play. The idea is to integrate documentation (doc) into the DevOps process. The ability to include documentation continuously within CI/CD can bring higher quality to the generated documents. Through daily references integrated into deliveries, it is possible to increase information accuracy, specification integrity and timeline traceability [13].

DevDocOps is an innovative approach, still considered generalist and lacking more concrete definitions, especially in the field of data engineering. There are no precise standards or guidelines that define how the documentation flow should take place during development [13, 19]. In terms of tools, there are examples on the market, but they mostly work on code comments to create reference guides [8, 17]. There are other technical artifacts that can be obtained and converted into high-quality documents that can effectively promote DevDocOps [9, 13, 20].

In this paper, we present the results of an empirical DevDocOps study, conducted in a real industrial setting. The results serve to highlight the importance of DevDocOps to promote quality, consistency, and efficiency in the creation, review, and availability of technical documentation in this specific context. It also led to the definition of some guidelines to introduce DevDocOps into a typical software life cycle.

The remainder of this paper is organized as follows: Section 2 presents related work. Sections 3 and 4 cover planning and execution of the study, followed by the results and discussion (Section 5). The guidelines are presented in Section 6. Threats to validity are discussed in Section 7 and Section 8 presents final remarks and future work.

2 RELATED WORK

The study by Theunissen [18] examines the balance between documentation and communication in Lean, Agile, and DevOps, known collectively as Continuous Software Development (CSD). It highlights challenges in maintaining effective communication with minimal yet sufficient documentation and proposes a framework for documentation tailored to CSD. This framework defines “necessary conditions” as minimal documentation requirements and “sufficient conditions” as ensuring effective communication based on documented information. The research suggests further investigation into correlating information types and documentation practices across industries using CSD, aiming to organize dispersed information into coherent documentation.

Continuous software development uses an automated deployment pipeline, ensuring versions undergo automated testing and compilation, enabling faster, more secure application updates to the production environment [6, 22]. Emphasizing agile principles, it breaks down development into smaller, manageable project elements to expedite execution [3, 21].

DevOps is an approach focused on integrating development (dev) and operations (ops) teams by building pipelines to automate CI/CD processes [6, 7, 24]. The transformation brought about by the application of the approach enables developers to perform code cross-checking, unit testing and constant compilation, allowing them not only to deliver updated code with high frequency, but also to detect errors and resolve problems more quickly [24].

Since the First International Conference on Computational Data Engineering (COMPDEC) in 1984, data engineering has been formally seen as a set of activities involving: the design of logical and physical databases; data management methodologies; computer architectures for knowledge databases; technology, implementation and operation for data management; and specialized tools [10].

According to Reis and Housley [12], although there are currently several database development techniques, the literature still lacks a formal definition and standards for data engineering applications.

The studies by Leite et al. [6], Poniszewska-Marańda et al. [9], Rong et al. [13], Synko and Peleshchyshyn [17], Theunissen [18] show that technical software documentation aims to achieve the following objectives: (i) Describe the requirements that have already been developed and validated; (ii) Record knowledge of problems and solutions encountered during the life cycle; (iii) Ensure the integrity of knowledge and information about the software developed; (iv) Record and track test results of the versions developed; and (v) Facilitate code reuse and software maintenance.

Aghajani et al. [1] investigated software professionals' needs for high-quality automatic documentation, addressing limitations like the narrow scope of previous studies. They used a qualitative approach, analyzing diverse artifacts from software repositories, development emails, forums, and problem reports. Despite reviewing various tools and approaches for automated documentation, their study concludes that inaccurate documentation remains a critical bottleneck, with potential severe consequences beyond time wasted on code reproduction.

According to Rong et al. [13], integrating software documentation into DevOps, termed DevDocOps, helps automate documentation but doesn't fully resolve all issues. The term DevDocOps was first mentioned in 2019 and has been practically applied in private settings, with limited analysis in open source communities. Rong et al. [14] proposed a documentation approach where developers define the information, guidelines, and models for automated documentation delivery.

The documentation automation tools on the market mostly work on code comments, creating reference guides [16, 17], but there are other technical knowledge artifacts that can be obtained and converted into high-quality documents, such as classes, tags, packages, methods, test scenarios and results, among others [9, 13, 20].

Industry also has some efforts that align with DevDocOps. For example, SAP HANA (High-Performance Analytic Appliance) is an in-memory database platform developed by SAP SE. This technology encompasses database, real-time data processing and advanced analytics capabilities in a single tool. As described by Färber et al. [4], Załęski [25], its main features and benefits are: (i) In-Memory Computing, in which data is stored in the server's main memory, providing ultra-fast access to data, significantly speeding up processing and analysis; (ii) Unified UI for BW/HANA[1] modeling tools; (iii) Advanced ETL (Extract, Transform and Load) functionalities that allow the integration of data from various sources, including SAP and non-SAP systems; and (iv) Support for advanced analytics such as natural language processing, machine learning and predictive algorithms.

One of the main data modeling objects in SAP HANA are Calculation Views. Modeled from a graphical interface, they are used to model business rules and logic, and return data sets as output. Because Calculation Views are structured database artifacts capable of complex calculations and massive business rule applications, comprehensive and detailed documentation is particularly important. The work by Załęski [25] presents a detailed description of Calculation Views, providing a comprehensive analysis of their functions and exploring the different types available.

When analyzing the use of SAP HANA as a Database Management System (DBMS), there is a lack of tools external to SAP that are capable of documenting its objects, and even with internal tools, standardizing documentation is a challenge. A common strategy is to use the SAP HANA Studio[2] Auto-Documentation feature, which allows the creation of documentation in PDF format, but these automatic documentations often do not present all the necessary information in a useful way or are adapted to produce specific documents, such as a mapping document.

As it can be seen, research efforts in DevDocOps can be found in the literature and industry, but they are still at an initial stage in terms of definitions, standards and guidelines.

3 PLANNING OF THE EMPIRICAL STUDY

The research aimed to study DevDocOps in a real industrial scenario, in order to increase the body of evidence regarding this approach with feedback from professionals. More specifically, the research aimed to create and evaluate an approach to streamline the production and continuous delivery of data engineering documentation, in line with the DevDocOps philosophy. Figure 1 describes how planning was carried out in five stages.

3.1 Bibliographic survey

The literature review followed a snowballing process based on two papers used as seed: the work of Rong et al. [13], which is one of the first mentions of the term DevDocOps; and the work of Theunissen [18], who reported a study on how to integrate documentation in Continuous Software Development. Forward and back snowballing was applied to these papers, resulting in the studies by Leite et al. [6], Poniszewska-Marańda et al. [9], Rashid et al. [11], Rong et al. [13], Synko and Peleshchyshyn [17], Theunissen [18], Theunissen et al. [19]. As described in Section 2, they helped to identify and

[1]www.sap.com/brazil/products/technology-platform/hana/what-is-sap-hana.html
[2]blogs.sap.com/2012/03/31/auto-documentation-functionality-in-hana-studio/

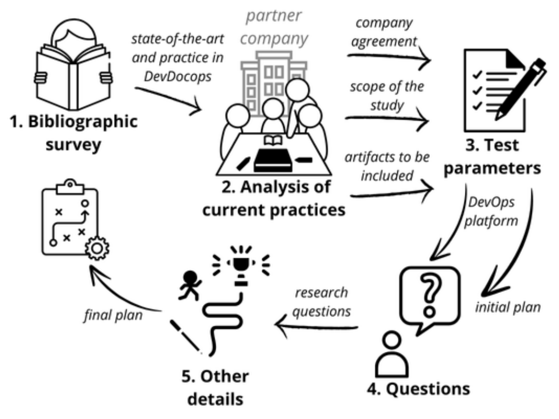


Figure 1: Planning stages

characterize the existence of a bottleneck in the technical documentation of software projects developed and structured using agile methodologies. The result of this analysis is described in Section 2.

3.2 Analysis of current practices

In order to conduct the study in a real industry setting, a partnership with a software development company was established. The company selected as a partner is an entity with a significant impact on the national technology market, standing out in the consulting and software development scene in Brazil. The initial contact was made by the researcher, and the collaboration consisted in an analysis of current documentation practices and the proposal of a DevDocOps approach to be included in the company's software development process.

The planning process involved holding meetings to present the proposal and define a research methodology that would ensure that the organization remained anonymous. These meetings were held in collaboration with a data engineering manager, the key figure responsible for the pillar of data projects involving SAP technologies in the company, in order to ideate and assess the project's objectives and possible practical applications. The company then agreed to make a data engineering consultant available to accompany and support the development and evolution of the project on a voluntary basis, and collaborative dialogues were established.

By monitoring and analyzing the documentation process in two different data engineering development projects, it was possible to map the discrepancies and similarities between the groups, even though they belong to the same technology chain. Both teams share common characteristics, such as the involvement of more than five active developers, an established organizational maturity (existing for more than six months) and experiences with participant exchanges throughout their existence. The documents produced in these projects include elements such as visuals, scripts at a moderate level of description, modeling and evidence of how these functionalities work. However, it was observed that there is no strict standardization regarding the specific phase for preparation or the format to be adopted for these documents, with variability depending on the team, client and management involved in each data engineering project.

After this analysis, the researcher, together with the company's data engineering manager, decided that the most important aspects for the new DevDocOps process were the need for standard, uniform access and constant updates. In this sense, two kinds of documents were selected: *Data Dictionaries*, an essential documentation for data-based projects [11]; and *Calculation Views*, graphical visualization objects that represent partial views of the database, which in the case of the partner company, is SAP HANA.

3.3 Test parameters

In the third stage of the research planning, which focused on the detailed planning of the test parameters, the DevOps platform and the tools that would be used to design and apply the guidelines proposed in this research were defined. Within this context, a comprehensive test plan was drawn up that would be carried out in the organizational environment, with a pre-defined team aligned with the partner company. As a result, the Azure DevOps platform was chosen as the tool for applying the DevDocOps approach. This choice was justified by a number of application factors necessary for the process to work, regardless of the organizational environment applying it. Firstly, the native and comprehensive integration offered by Azure DevOps is capable of centralizing project management, version control, CI/CD pipelines, task tracking and wikis[3], making it possible to achieve efficient synergy, enabling holistic management of the development lifecycle [15].

3.4 Questions

Four questions were defined to establish the improvements that can be achieved with DevDocOps in a more objective way. These questions were based on four important aspects that were defined after the literature review and meetings with the company participants:

- Q1 - Agility: Can DevDocOps reduce the time spent producing documentation?
- Q2 - Quality: Can DevDocOps help to produce more standardized, detailed and correct documentation?
- Q3 - Continuous Integration: Can DevDocOps promote continuous integration of documentation with source code?
- Q4 - Documentation Delivery: Can DevDocOps promote continuous delivery of the produced documentation to the interested stakeholders?

As a means of obtaining answers to these questions, the method chosen for extracting the results was to apply a questionnaire to the participants (developers and leader), which they had to answer after having carried out the documentation tasks. The questions were strategically divided into two domain areas, aimed at the developers and leader involved in the process. This approach made it possible to formulate a comprehensive questionnaire, adapted to the specific perspectives and responsibilities of each group.

To complement the answers from the questionnaire, constant interviews were to be conducted with the developers on a day-to-day basis. For the leader, we conducted one final interview, to complement the responses from the questionnaire with more detailed feedback regarding aspects such as DevDocOps Process Impact

[3]The Wiki in Azure DevOps is a collaboration and documentation tool integrated into the Azure DevOps Services platform

on Documentation Time, Documentation Standardization Level, Documentation Accuracy and Documentation Interest Level.

3.5 Other details

Before starting the project, it was essential to set the study duration, allocate sufficient human resources, identify database objects, and implement strict anonymization measures. Collaboration with a partner company was crucial for ensuring project feasibility and success, highlighting the integration of academic research with business practice. The process included incremental feedback loops aligned with development cycles. Plans were presented, awaiting management approval, followed by scheduling involvement. Development was structured into four sprints, each lasting two weeks, where developers performed tasks potentially impacting project documents. This iterative approach facilitated comprehensive evaluation and incremental adjustments as new study elements were introduced.

4 EXECUTION

The planning stage described in the previous section took around nine months. After this, execution started, in three stages.

4.1 Stage 1: Document generation scripts

This stage aimed to boost agility in the documentation process by developing scripts capable of generating data engineering documentation, in an automated way, from certain input documents, namely Table Structures and Calculation Views. This process resulted in a standardized output document, helping to raise the quality of the documented material. Figure 2 illustrates this step.

To scan the Table structures in order to generate data dictionaries, we used Python with Pandas[4], a library that facilitates data manipulation. This script is designed to receive two CSV files as input – the table structure and some configuration metadata – and produce a data dictionary in *Markdown* language. Figure 3 shows an example of a generated data dictionary.

In order to design the script capable of scanning the Calculation Views, it was necessary to carry out an in-depth analysis of their technical structure in SAP HANA. This analysis involved a detailed understanding of the internal organization of Calculation Views, such as the relationships between tables, join operations, aggregations, and the logic underlying calculations. During this analysis phase, the crucial elements that make up a Calculation Views were identified, highlighting the critical points for extracting relevant information. Understanding the XML structure, which describes the hierarchy and relationships, was essential to creating a script capable of performing a precise and comprehensive scan, and the prior analysis not only underpinned the design of the script, but also made it possible to anticipate potential challenges and develop effective strategies for extracting technical data in a coherent manner.

[4]https://pandas.pydata.org/

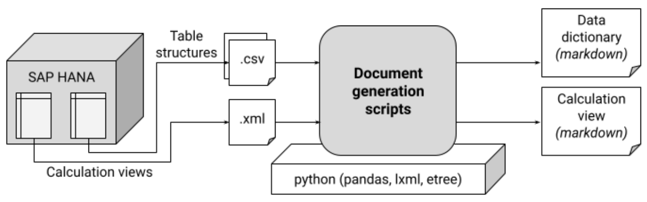
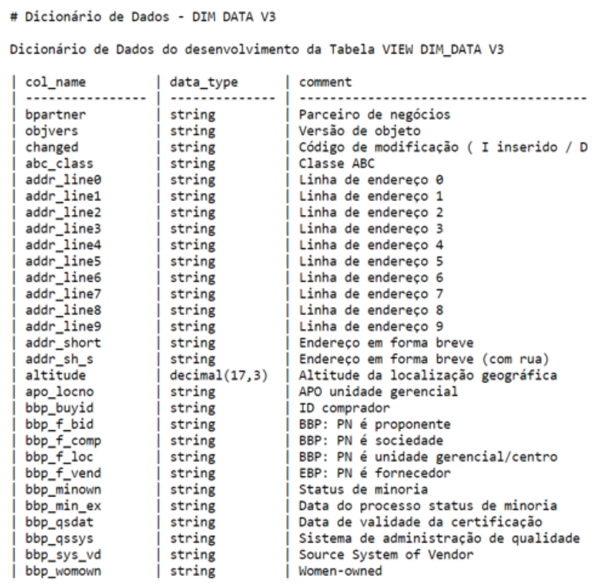


Table structures
SAP HANA
.csv
.xml
Calculation views
Document generation scripts
python (pandas, lxml, etree)
Data dictionary (markdown)
Calculation view (markdown)

Figure 2: First stage: document generation scripts



```
# Dicionário de Dados - DIM DATA V3

Dicionário de Dados do desenvolvimento da Tabela VIEW DIM_DATA V3

| col_name          | data_type      | comment
| ----------------- | -------------- | --------------------------------------
| bpartner          | string         | Parceiro de negócios
| objvers           | string         | Versão de objeto
| changed           | string         | Código de modificação ( I inserido / D
| abc_class         | string         | Classe ABC
| addr_line0        | string         | Linha de endereço 0
| addr_line1        | string         | Linha de endereço 1
| addr_line2        | string         | Linha de endereço 2
| addr_line3        | string         | Linha de endereço 3
| addr_line4        | string         | Linha de endereço 4
| addr_line5        | string         | Linha de endereço 5
| addr_line6        | string         | Linha de endereço 6
| addr_line7        | string         | Linha de endereço 7
| addr_line8        | string         | Linha de endereço 8
| addr_line9        | string         | Linha de endereço 9
| addr_short        | string         | Endereço em forma breve
| addr_sh_s         | string         | Endereço em forma breve (com rua)
| altitude          | decimal(17,3)  | Altitude da localização geográfica
| apo_locno         | string         | APO unidade gerencial
| bbp_buyid         | string         | ID comprador
| bbp_f_bid         | string         | BBP: PN é proponente
| bbp_f_comp        | string         | BBP: PN é sociedade
| bbp_f_loc         | string         | BBP: PN é unidade gerencial/centro
| bbp_f_vend        | string         | EBP: PN é fornecedor
| bbp_minown        | string         | Status de minoria
| bbp_min_ex        | string         | Data do processo status de minoria
| bbp_qsdat         | string         | Data de validade da certificação
| bbp_qssys         | string         | Sistema de administração de qualidade
| bbp_sys_vd        | string         | Source System of Vendor
| bbp_womown        | string         | Women-owned
```

Figure 3: Example of a generated data dictionary

The script developed works with the XML file associated with the Calculation View. Using the lxml[5] and etree[6] libraries, it was possible to effectively map the existing nodes, identifying their characteristics, functions and the chains that permeate the entire structure of the object. This approach not only simplifies the documentation process, but also ensures an accurate and comprehensive representation of the technical nuances of the Calculation View, promoting a detailed understanding for those involved in the process of analyzing and maintaining these complex structures. The resulting documentation represents the view in *Markdown* language. Figure 4 shows an example of a generated calculation view document.

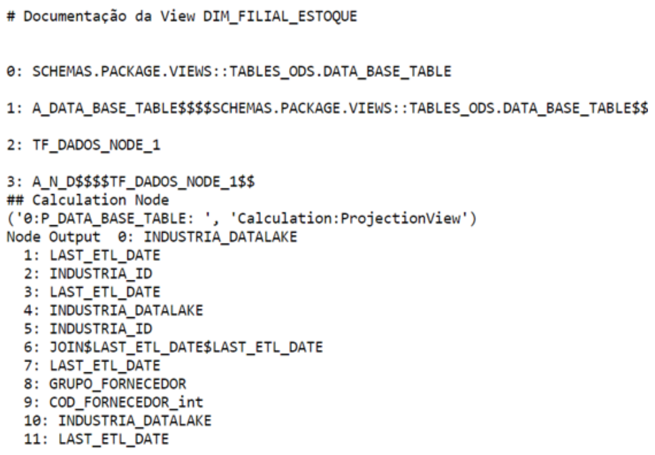
This stage was carried out by a single researcher, and took around one and a half month to be completed.

4.2 Stage 2: DevDocOps configuration

The second stage consisted in configuring the DevDocOps environment (Azure DevOps), including: (i) Azure Repos (Azure's GIT-based source code repositories) to maintain source code, including the aforementioned `.csv` and `.xml` files; (ii) the CI/CD pipeline to automatically run the scripts; (iii) a wiki to hold the generated documents; and (iv) a board to view/manage the development stories. Figure 5 illustrates these elements (bottom side of the figure).

First, some basic configurations were made to enforce some DevOps good practices: the source code repository was configured to prevent commits to be made into the main branch, and to have

[5]https://lxml.de/
[6]https://docs.python.org/3/library/xml.etree.elementtree.html



```
# Documentação da View DIM_FILIAL_ESTOQUE


0: SCHEMAS.PACKAGE.VIEWS::TABLES_ODS.DATA_BASE_TABLE

1: A_DATA_BASE_TABLE$$$$SCHEMAS.PACKAGE.VIEWS::TABLES_ODS.DATA_BASE_TABLE$$

2: TF_DADOS_NODE_1

3: A_N_D$$$$TF_DADOS_NODE_1$$
## Calculation Node
('0:P_DATA_BASE_TABLE: ', 'Calculation:ProjectionView')
Node Output  0: INDUSTRIA_DATALAKE
  1: LAST_ETL_DATE
  2: INDUSTRIA_ID
  3: LAST_ETL_DATE
  4: INDUSTRIA_DATALAKE
  5: INDUSTRIA_ID
  6: JOIN$LAST_ETL_DATE$LAST_ETL_DATE
  7: LAST_ETL_DATE
  8: GRUPO_FORNECEDOR
  9: COD_FORNECEDOR_int
  10: INDUSTRIA_DATALAKE
  11: LAST_ETL_DATE
```

Figure 4: Example of a generated calculation view document

branches always associated with a development story. The CI/CD pipeline was configured to execute some basic code quality scripts.

Regarding the "Doc" part of DevDocOps, the following specific configurations were made. First (Figure 5-1), the document generation scripts described in the previous section were configured to run whenever a pull request (PR) is made (a request to merge some secondary branch into the main branch). As a result, the corresponding documents (Data Dictionary - `dd.md` and Calculation View - `cv.md`) are generated and saved in the environment.

The pull request must be reviewed and approved by someone responsible, such as a tech leader or manager. At this point the documents have already been generated, so he/she can inspect it together with the other software assets that have been produced. Once the pull request is approved (Figure 5-2), another script publishes the generated documentation into the Wiki, where it can be accessed by its stakeholders in an organized way.

The result of this process is that the wiki will always have the most recent approved version of the documentation, at least in regard to these particular documents.

This stage was carried out by a single researcher, and took around two and a half months to be completed.

4.3 Stage 3: DevDocOps in practice

The third stage was carried out with the aim of putting the proposed flow into practice in a real development environment according to the planning. Figure 6 illustrates the process and its main elements: the center contains the eight activities that were executed; the left side defines the duration of each group of activities; the right side

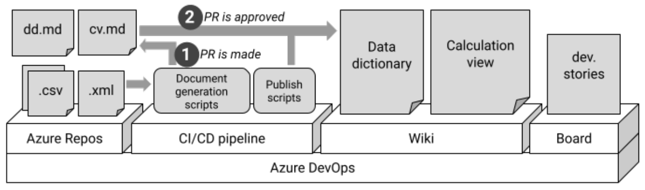


Figure 5: DevDocOps environment configuration

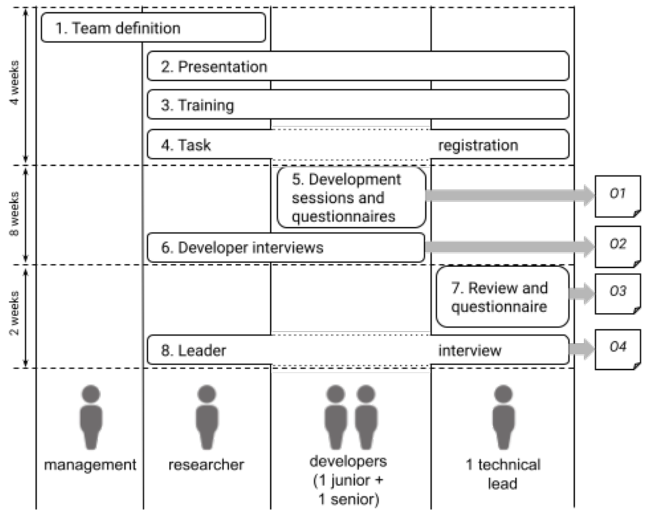


Figure 6: DevDocOps in practice

summarizes the outputs of each activity (more details below); and the bottom shows the roles that participated in each activity.

1. Team definition: This is the only activity that involved company management. Three participants were selected for the team: a junior developer, a senior developer and a technical lead. The diversity of experience and skills was intended to cover different perspectives during the evaluation. Participation was voluntary.

2. Presentation: The team was properly introduced to the project, starting with a group meeting, in order to pass on detailed information about the objectives, methodology and expectations related to the implementation of the DevDocOps approach.

3. Training: The necessary access to the platform was provided, followed by a dedicated training session. The training covered the practical aspects of using the platform, ensuring that all participants were familiar with the features.

4. Task registration: To organize the workflow, the tasks related to the development objects chosen to make up the sample were registered by the researcher and technical lead on the platform board. Seven different tasks were defined, each based on different developed objects, as summarized in Table 1. This diverse approach allowed for a comprehensive analysis and a more holistic understanding of the impact on daily development activities and the production of documentation.

5. Development sessions and questionnaires: Seven development sessions were held with the two developers to carry out the seven tasks within the proposed guidelines. The tasks were split between the two developers, so that the Junior developer performed four tasks and the Senior developer performed three tasks. At the end of each session, the developers were subjected to a questionnaire to collect feedback.

6. Developer interviews: Together with the development sessions, the researcher conducted chat sessions with the developers to gather more detailed impressions about their experience with the platform, areas of satisfaction and any challenges faced.

7. Review and questionnaire: The technical lead reviewed the documentation generated on the platform based on the deliveries

Task	Dev	Description
1	Jr.	Develop a data consumption flow in SAP Hana from a CSV file to a final Stock output table, including data staging, sanitization and processing with a Flowgraph pipeline.
2	Sr.	Add new fields to the portfolio table for branches X and Y migration to SAP, including load job, staging, final table adjustment and Flowgraph pipeline.
3	Jr.	Create a table to extract and share sales data from supplier Y, filtering from sales and anonymizing customer data.
4	Sr.	Add new fields to the sales table for branches X and Y migration to SAP, including load job, staging, final table adjustment, and Flowgraph pipeline .
5	Jr.	Develop a Calculation View to consolidate stock data from all branches.
6	Sr.	Adjust sales calculation view to include branches X and Y, correct the join with the sales dimension, and fix duplication with customers view by considering the valid customer record on the order date.
7	Jr.	Develop a Calculation View to consolidate open orders from all branches.

Table 1: Tasks defined for the study

made by the developers and made available on the wiki. The results were collected by means of a questionnaire, with the aim of gathering impressions about the experience on the platform.

8. Leader interview: An interview session was held with the technical lead to discuss the experience with the platform in comparison with the current documentation process, with the aim of finding the perceived benefits and opportunities for improvement.

Four outputs were produced:

O1. 7 responses to questionnaires by the developers (4 from the junior developer and 3 from the senior developer);
O2. daily interviews with developers;
O3. 1 response to questionnaire by the technical lead; and
O4. interview with the technical lead.

5 RESULTS

Here we discuss the results obtained after analyzing each output described in the previous section.

5.1 Outputs O1 and O2: Developers' Questionnaires and Interviews

Output O1 refer to the questionnaires that were applied after each development session (referred to as "story" in the questionnaire). The following questions were asked:

1. Was documentation produced or updated in this story?
2. Amount of time spent (hours) on documentation (time for automatic generation, reviews)
3. If the DevDocOps process had not been used, how much time would you estimate you would have spent?
4. Does the object under development already have documentation? If so, did you need to consult the previous documentation of this object to perform the task? Was the previous documentation up to date with the same version of the object in production? Was it easy to access or locate the documentation?
5. Was it necessary to develop any documentation manually? If yes, please describe.
6. Did the documentation require adjustments after it was generated? If yes, which ones?
7. How easy was it to use the platform?
8. Were there any issues in using the DevDocOps workflow?
9. With the new process, documentation has become mandatory. How do you evaluate the benefits and challenges of this continuous documentation delivery?
10. What benefits and difficulties have you observed regarding the delivery method (availability, access, and visualization) of documentation on the platform?

Question 1 served mostly to confirm that the stories did involve documentation being produced/updated. The following conclusions were drawn from the other questions:

Document Production Efficiency and Impact of the DevDocOps Process on Documentation Time: Both developers shared the same perception: DevDocOps can reduce the time needed to produce technical documentation. Table 2 summarizes their responses to questions 2 and 3 (amount of time spent on documentation with DevDocOps versus amount of time estimated without DevDocOps). The answers were not precise as they are based on their perception, and not actually measured times. As it can be seen, the time spent with DevDocOps was much smaller than the estimates for the process without DevDocOps. Although these are based on estimates, the consistent results between both developers provide some degree of confidence in their accuracy.

Documentation Accuracy and Conformance between Documentation and Code in Production: The feedback for question 4 and its subquestions revealed that the documentation generated by DevDocOps was aligned with the code in production, with a good accuracy, i.e. the produced documentation accurately reflected the data models and functionalities implemented. Developers also reported that they did not encounter much problem in locating the required documents.

Need for manual effort: Questions 5 and 6 focused on the need to develop or complement the automatically generated documentation. Responses indicate that the developers did not need such effort. In some cases minor adjustments had to be made to improve the visualization and hierarchy for a calculation view, but this did not impact its understanding.

Pros and cons of the new DevDocOps process: The answers to questions 7-10 reported that the new process was easy to use and problem-free. As pros, they highlighted the effort saving potential and standardization of the documents. They also mentioned the

Task	Developer	Time spent w/ DevDocOps	Estimate w/o DevDocOps
1	Jr	10min	45min - 1hour
2	Sr	5min	45min - 1hour
3	Jr	< 10min	2hour
4	Sr	5min	2hour
5	Jr	15 - 25min	2hour - 3hour
6	Sr	20min	2hour
7	Jr	10min	2hour - 3hour

Table 2: Time spent with DevDocOps vs estimate without it

tracing between the source code and the documents up to the wiki, which makes accessing and updating them an easy task. For example, the Junior developer stated the following: "the process has become simpler and more consistent, automatically centralizing and standardizing the documentation (...) allowing the time that would be spent on documentation to be directed to other activities."

Some difficulties were reported in terms of the initial lack of knowledge with the new process. For example, the Senior developer stated: "I believe that the activities are relatively easy to carry out, but they can pose difficulties for those who have never worked with the platform". There were also some effort to understand the formatting of the produced document. But these are only minor problems that were quickly resolved after some time.

Constant interviews (O2) were also being conducted to complement their responses. Much of the content from the interviews only confirmed their responses to the questionnaires, but the following additional observations were made:

Continuous Delivery: The participants noted that the continuous delivery process, following the pattern of a single artifact, significantly facilitated access to the documentation. It was pointed out that in the previous process, there were cases in which the documentation could not be finalized in the current sprint, usually due to time constraints and delivery priorities, highlighting the improved efficiency of the documentation in the automated process, due to the time savings in carrying out such actions.

Mandatory Delivery with Documentation: The developers reported that the obligation to deliver the task together with the documentation did not adversely affect delivery times. On the contrary, the ease of use and speed of the platform were seen as benefits.

Suggested Improvements: Although the process was generally well received, the participants suggested some improvements. For example, an initial follow-up was recommended to facilitate understanding of the platform. In addition, it was suggested to automate the export of task configurations to avoid manual errors, improving the efficiency of the process.

These findings suggest that, from the developers' perspective, DevDocOps had a positive impact on the efficiency of documentation production, the accuracy of documents and agile access to technical information. The results indicate that the goals set for this perspective were successfully achieved, improving the overall effectiveness of the documentation process.

Next we analyze the responses by the leader.

5.2 Outputs O3 and O4: Leader's Questionnaire and Interview

From the leader's perspective, we wanted to assess how the DevDocOps impacted the leader's work and responsibilities. The following questions were asked in the form of a questionnaire (output O3):

1. Was the agility of the documentation positively or negatively impacted by the implementation of this process?
2. What improved and what worsened in terms of the agility of the story development?
3. Did the amount of documentation delivered in the sprint increase?
4. In terms of documentation standardization, was there an improvement?
5. Compared to the previous documentation process, does the documentation delivered with the new process meet the technical requirements?
6. With the new process, documentation became mandatory. How do you evaluate the benefits and issues of this continuous documentation delivery?
7. What benefits and difficulties have you observed regarding the delivery method (availability, access, and visualization) of the documentation on the platform?
8. Is the documentation being continuously delivered? What could be improved?

The answers obtained from the questionnaire filled in by the leader provided the results needed to assess how the process influenced documentation time, adherence to standards, documentation accuracy and the level of interest in technical documentation.

Impact of the DevDocOps process on documentation time: The leader's responses to questions 1 to 3 corroborated the developers' perception and also indicated a significant improvement in documentation time with the *DevDocOps* process. The goal was achieved based on the analysis of the simplified and automated process that was applied, which allowed the development team to produce documentation more quickly and efficiently, freeing up time to focus on other critical project activities.

Quality and adherence to standards: The leader pointed out in question 4 that DevDocOps improved the documentation quality and the team's adherence to established standards. The automation and standardization inherent in the process ensured that the documentation was consistent and followed the predefined standards. Although there were improvements suggested by the developers, the results obtained in the study showed a more organized, clear and easy-to-review documentation, which is fundamental for maintaining quality and consistency in technical information.

Documentation Accuracy: Documentation accuracy was another area that benefited from the implementation of DevDocOps. As reported by the leader in question 5, the automated process reduced human errors and inconsistencies in technical documentation. Documentation became more reliable, more accurately reflecting the current state of the code in production.

Pros and cons of the new DevDocOps process: In questions 6 to 8, the leader reported an increase in the team's level of interest in the technical documentation. The leader noticed that, with DevDocOps, the documentation became more accessible and valuable to the development team. This encouraged team members to get more involved with the documentation, becoming more autonomous in their use of it. The leader put this in the following words: "As a leader, I see documentation as a "living" part of the project cycle, where it should be treated as part of the continuous cycle. If there are new changes and additions, the documentation reflects these changes, allowing us to version the documentation. This helps mitigate possible errors and outdated rules, requirements, and relevant information for the project". The increased interest in documentation can be attributed to the ease of access, quality and usefulness of the documentation produced by the DevDocOps process.

In terms of suggestion for improvement, the leader highlighted: "To improve, I understand that the tool should be adapted to various scenarios of low and high complexity, expanding and consequently adapting it over time. Additionally, for technical leaders, the number of documents to be reviewed increases considerably. I see an opportunity to streamline the review and governance process and automate this process."

After the questionnaire, an individual interview was conducted with the leader (output O4). The following additional comments were made:

Functional and Technical Specification: The leader highlighted the importance of functional and technical specification, emphasizing that this process involves clearly and accurately documenting all the customer's technical requirements. The technical specification covers the detailed technical procedures related to product development, ensuring that all essential information is documented in a comprehensive manner. This "living" approach to documentation, which adapts and evolves throughout the project cycle, has been recognized as a crucial benefit, providing technical visibility and facilitating decision-making. According to the leader, compared to traditional methods, the new documentation process better met technical needs, providing gains in time, standardization and technical detail.

Benefits of Documentation: The leader pointed out several benefits of the detailed specification, including promoting a common understanding among all stakeholders, serving as a reliable reference point during development and identifying gaps in requirements at an early stage. This process contributes to reducing costs and development time, avoiding rework and misunderstandings.

Story Development: Regarding agility, the leader noted an improvement in the rapid generation of artifacts, allowing the technical team to focus on other deliverables. However, a need was pointed out for more rigorous monitoring of the delivery made by the developers and increased governance in artifact reviews.

Project Goals: In relation to the established goals, the leader acknowledged the effectiveness of the process in speeding up the production of documentation, ensuring its quality and standardization, and facilitating continuous delivery. He emphasized that the documentation meets technical needs, fulfilling a strategic role in the project and encouraging an effective documentation culture.

After this detailed analysis of the feedback from the subjects, we can answer the four questions established for this research:

Q1 - Agility: Can DevDocOps reduce the time spent producing documentation? All participants reported that the DevDocOps approach reduced the time spent with documentation.

Q2 - Quality: Can DevDocOps help to produce more standardized, detailed and correct documentation? The subjects reported that quality with DevDocOps increased. The leader, particularly, praised the standardization, while developers noticed the level of details and correctness.

Q3 - Continuous Integration: Can DevDocOps promote continuous integration of documentation with source code? All the subjects noticed that continuous integration allowed a constant synchronization between the documentation and the source, without significant additional effort.

Q4 - Documentation Delivery: Can DevDocOps promote continuous delivery of the produced documentation to the interested stakeholders? Both developers and the leader reported that the adopted approach helped documentation reach their interested stakeholders easily.

6 DEVDOCOPS APPLICATION GUIDELINES

In this section we list some guidelines for DevDocOps:

G1. Is is not just automation. This is the most important lesson learned: DevDocOps should integrate automation into the development life cycle and the review processes, enforcing continuous synchronization with source code and concerns regarding quality and accessibility standards.

G2. Different responsibilities. We observed at least two responsibilities: (i) the developers', who are responsible for creating, maintaining and updating documentation as they progress through the development processes, and focuses on the importance of clear and concise documentation of code and changes, ensuring future maintainability and comprehension; and (ii) the leader's, who has responsibility for overseeing and ensuring that documentation practices and continuous integration workflows are aligned with the overall project objectives and the organization's agile practices, and stresses the need to review and validate documentation to ensure that it meets all the technical requirements and standards of the project. Both roles are structured to complement each other, maximizing the efficiency and quality of the final product delivered in the data engineering environment.

G3. DevDocOps-Driven Leadership. As the main applicator of the practice, it is up to the leader to evaluate and customize the documentation template defined for each project and client. This task includes establishing specific rules for the documentation, such as formatting, depth of information and identification of technical needs. In addition, the leader is responsible for ensuring that the guidelines are adapted to the unique characteristics of each team, starting from the generic script provided, improving and customizing it according to the individual requirements of each project, adapting to the specific demands.

G4. Backlog Management for Efficient Tracking. To ensure effective tracking of the documentation objects developed, it is crucial that the leader structures the backlog in such a way as to group together activities related to the same object change during the sprint. Although the tasks should reflect the smallest possible fragment of documentation, following agile principles, in data engineering it is essential to consider the proper functioning of the object being documented due to its dependencies and impacts on the production environment. In this way, the changes to the object will be reflected more clearly in the final document, guaranteeing a more precise and visible breakdown of the changes made.

G5. Quality Assurance and Safety in Project Delivery. In the proposed flow, it is essential that each delivery includes a detailed review of the documentation objects developed by the programmers, since the deployment now requires the approval of the leader before being carried out in the main branch of the repository. This approach increases the security of the delivered solution, while requiring stricter and more targeted control and monitoring at each delivery phase, reinforcing the quality and reliability of the documentation included in the final product.

G6. Secure Change Management and Deployment in DevDocOps. The developer is responsible for managing changes in such a way as to prevent direct deploys to the main branch of the repository. This practice increases the security of deployments, minimizing risks and impacts on the production environment. It also facilitates the automatic activation of the continuous documentation pipeline, which is triggered once the code has been approved by the leader. This method ensures safe and effective integration of code changes, promoting system stability and reliability.

G7. Centralizing Documentation in DevDocOps: An Access and Security Practice. The automated flow for generating documentation after code validation implies that the documentation created is integrated into the main context, with the option of downloading it. It is vital that the leader promotes the practice of keeping documentation centralized, even with the possibility of transferring documents externally. This facilitates access to historical and current records, consolidating a best practice in DevDocOps to guarantee the accessibility and security of information.

G8. Fine-tuning automated documentation. Although the documentation process is automated, guaranteeing a uniform standard for all objects developed, it is crucial to consider the possibility that, in certain cases, the documentation may require additional adjustments, especially for objects of greater complexity. The developer is responsible for completing the documentation with a detailed review, while the leader has the task of checking that the final document meets the requirements.

G9. Effective Communication in the Life Cycle of Documentation Tasks. Communication is essential and must be reflected in the updates to the tasks defined in the project management framework. The record of all documentation deliveries must be clearly marked in the corresponding tasks. The life cycle of a task concludes with the delivery of the associated documentation, ensuring that each phase of the process is properly documented and communicated, from the start of development to its completion.

G10. Customization and Efficiency in DevDocOps: Using Dedicated Pipelines for Object Documentation. Documentation should be considered an organic element in the project lifecycle, subject to customization according to the specific needs of DevDocOps. It is recommended to implement separate pipelines for each data object, which not only speeds up execution, but also allows individual technical specificities of each element to be addressed. This modular and flexible approach ensures more efficient and adaptable documentation management, in line with the technical requirements and characteristics of each object within the project.

7 THREATS TO VALIDITY

In this section we discuss the threats to validity. We follow the structure suggested by Wohlin et al. [23] combined with the elements suggested by Feldt and Magazinius [5].

Conclusion validity: *"Does the treatment/change we introduced have a statistically significant effect on the outcome we measure?"*

Although it is not uncommon to find development teams with a few members, the small number of participants is a threat. Although we managed to negotiate the participation of three professionals with different levels and background, it is still a small number. Also, we managed to conduct seven development sessions, which also limits the observations. Finally, the outcomes were based on perception and estimates, and not actually measured times.

Internal validity: *"Did the treatment/change we introduced cause the effect on the outcome? Can other factors also have had an effect?"*

Pressure for results is always present in a company, therefore participants might have provided biased positive feedback in order to show that their effort had positive results, specially because they were encouraged by management to participate.

Construct validity: *"Does the treatment correspond to the actual cause we are interested in? Does the outcome correspond to the effect we are interested in?"*

Regarding the positive results in terms of agility, we believe the answer to this question is "yes", because automation has obviously reduced the time needed to produce documentation, together with other benefits perceived equally by all three participants.

External validity, Transferability: *"Is the cause and effect relationship we have shown valid in other situations? Can we generalize our results? Do the results apply in other contexts?"*

The documents chosen have a relatively simple structure, but even so, the lack of documentation prior to the study was evident. Therefore we believe the same effect could be observed in similar situations, with similar documents. However, many of the observed effects may not be perceived if the technical difficulties of implementing the scripts are higher than what was faced here.

Credibility: *"Are we confident that the findings are true? Why?"*

All three participants mostly agreed on the results, even with their different background and expertise levels. Also, the benefits of automation are consistent with the ones seen in other scenarios in the DevOps field.

Dependability: *"Are the findings consistent? Can they be repeated?"*

Responses from the three participants are consistent with each other and the ones from the literature. Also, the differences in the time spent with DevDocOps versus estimates without DevDocOps are consistent among the developers and what was expected.

Confirmability: *"Are the findings shaped by the respondents and not by the researcher?"*

Like the pressure they faced from management for positive results, the participants may have responded in a biased manner to the researcher because this study was approved by management. We tried to mitigate this threat by stating that this study was not meant to evaluate them, but the process being implemented.

8 CONCLUSION

This research explored the integration of documentation into the agile software development process, with a focus on the DevDocOps approach. The study presented a detailed overview of current practices and challenges faced in software documentation, particularly in data engineering, and proposed a series of guidelines to help others find their way in this process. The guidelines were based on the results of practical utilization in a real industrial scenario. A significant reduction in the time required for technical documentation and an improvement in the overall quality of documentation were observed. This time optimization, coupled with the automation of various documentation tasks, allows teams to focus on more critical aspects of development, improving efficiency and productivity. These results are important in a field that has not yet been extensively investigated by the research community.

Even with the growing interest in agile practices and the integration of documentation into the software development cycle, little has been studied about how the specific goals of projects like this one align with and impact data engineering environments. This gap in the literature suggests a significant opportunity for future research.

Here we identify four promising areas for future research: (i) to evaluate the proposed guidelines in different scenarios; (ii) to expand automation to more types of data engineering objects, such as API documentation and other types of documents. It would be interesting to see what kinds of benefits a DevDocOps approach could bring to these scenarios; (iii) to study, more in depth, the effect of standardized templates in improving documentation. This was one of the important learned lessons in our research, and it is important to evaluate how automatic template-based generation could improve documentation quality; (iv) to further investigate important cost-related questions that have arisen in our research. How much does it cost to create documentation? And, perhaps more importantly, how much would it cost NOT to create documentation continuously? Maybe the developers would spend more time trying to understand things that are not documented than the time needed to properly maintain correct and updated documentation.

AVAILABILITY OF ARTIFACTS

The artifacts developed in this project, necessary for the application of DevDocOps, are the documentation generation scripts, environment configuration files, and pipeline configuration scripts, which were made available under the Creative Commons License[7].

ACKNOWLEDGMENTS

This work was supported by CAPES - Finance Code 001.

REFERENCES

[1] Emad Aghajani, Csaba Nagy, Olga Lucero Vega-Marquez, Mario Linares-Vasquez, Laura Moreno, Gabriele Bavota, and Michele Lanza. 2019. Software Documentation Issues Unveiled. *Proceedings - International Conference on Software Engineering* 2019-May, 1199–1210. https://doi.org/10.1109/ICSE.2019.00122

[2] Len Bass. 2017. The Software Architect and DevOps. *IEEE Software* 35 (2017), 8–10. Issue 1. https://doi.org/10.1109/MS.2017.4541051

[3] Hajer Berhouma. 2020. A Generic Model for Software Documentation and its Application in Embedded Systems Developed with Scrum. *ACM International Conference Proceeding Series*, 33–36. https://doi.org/10.1145/3436829.3436858

[4] Franz Färber, Sang Kyun Cha, Jürgen Primsch, Christof Bornhövd, Stefan Sigg, and Wolfgang Lehner. 2012. SAP HANA database. *ACM SIGMOD Record* 40 (1 2012), 45–51. Issue 4. https://doi.org/10.1145/2094114.2094126

[5] Robert Feldt and Ana Magazinius. 2010. Validity threats in empirical software engineering research - An initial survey. In *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering*. 374–379.

[6] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2019. A Survey of DevOps Concepts and Challenges. *ACM Comput. Surv.* 52, 6, Article 127 (nov 2019), 35 pages. https://doi.org/10.1145/3359981

[7] Mirna Muñoz and Mario Negrete Rodríguez. 2021. A guidance to implement or reinforce a DevOps approach in organizations: A case study. *Journal of Software: Evolution and Process* (2021), e2342. https://doi.org/10.1002/smr.2342 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2342

[8] Danilo Pianini and Alessandro Neri. 2021. Breaking down monoliths with Microservices and DevOps: an industrial experience report. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 505–514. https://doi.org/10.1109/ICSME52107.2021.00051

[9] Aneta Poniszewska-Marańda, Arkadiusz Zieliski, and Witold Marańda. 2020. Towards project documentation in agile software development methods. *Lecture Notes on Data Engineering and Communications Technologies* 30 (2020), 1–18. https://doi.org/10.1007/978-3-030-19069-9_1

[10] C.V. Ramamoorthy, P. Bruce Berra, Barry Boehm, Peter c.c. Wang, Wesley Chu, and Gio Wiederhold. 1984. 1984 IEEE First International Conference on Data Engineering, IEEE Computer Society Press (Ed.). *1984 IEEE First International Conference on Data Engineering*.

[11] Sabbir M. Rashid, James P. McCusker, Paulo Pinheiro, Marcello P. Bax, Henrique O. Santos, Jeanette A. Stingone, Amar K. Das, and Deborah L. McGuinness. 2020. The semantic data dictionary – an approach for describing and annotating data. *Data Intelligence* 2 (10 2020), 443–486. Issue 4. https://doi.org/10.1162/dint_a_00058

[12] Joe Reis and Matt Housley. 2023. *Fundamentos de Engenharia de Dados*. Novatec, São Paulo - SP.

[13] Guoping Rong, Zefeng Jin, He Zhang, Youwen Zhang, Wenhua Ye, and Dong Shao. 2019. DevDocOps: Towards Automated Documentation for DevOps. *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019* (2019), 243–252. https://doi.org/10.1109/ICSE-SEIP.2019.00034

[14] Guoping Rong, Zefeng Jin, He Zhang, Youwen Zhang, Wenhua Ye, and Dong Shao. 2020. DevDocOps: Enabling continuous documentation in alignment with DevOps. *Software: Practice and Experience* 50, 3 (2020), 210–226. https://doi.org/10.1002/spe.2770 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2770

[15] Joachim Rossberg. 2019. *An Overview of Azure DevOpsAzure DevOps*. Apress, Berkeley, CA, 37–66. https://doi.org/10.1007/978-1-4842-4483-8_2

[16] Qiwei Song, Xianglong Kong, Lulu Wang, and Bixin Li. 2020. An Empirical Investigation into the Effects of Code Comments on Issue Resolution. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. 921–930. https://doi.org/10.1109/COMPSAC48688.2020.0-150

[17] A. Synko and A. Peleshchyshyn. 2020. Software development documenting – documentation types and standards. *Scientific journal of the Ternopil national technical university* 98 (2020), 120–128. Issue 2. https://doi.org/10.33108/visnyk_tntu2020.02.120

[18] Theo Theunissen. 2020. Identifying Conditions for Effective Communication with Just Enough Documentation in Continuous Software Development.. In *CAiSE (Doctoral Consortium)*. 11–20.

[19] Theo Theunissen, Stijn Hoppenbrouwers, and Sietse Overbeek. 2022. Approaches for Documentation in Continuous Software Development. *Complex Systems Informatics and Modeling Quarterly* (10 2022), 1–27. Issue 32. https://doi.org/10.7250/csimq.2022-32.01

[20] Theo Theunissen, Uwe van Heesch, and Paris Avgeriou. 2022. A mapping study on documentation in Continuous Software Development. *Information and Software Technology* 142 (2022), 106733. https://doi.org/10.1016/j.infsof.2021.106733

[21] Mark Underwood. 2023. Continuous Metadata in Continuous Integration, Stream Processing and Enterprise DataOps. *Data Intelligence* 5 (12 2023), 275–288. Issue 1. https://doi.org/10.1162/dint_a_00193

[22] Ram Mohan Vadavalasa. 2020. End to end CI/CD pipeline for Machine Learning. *International Journal of Advance Research, Ideas and Innovations in Technology* 6, 3 (06 2020).

[23] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. 2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA.

[24] Ravi Teja Yarlagadda. 2021. DevOps and Its Practices. *International Journal of Creative Research Thoughts (IJCRT)* 9 (2021), 111–119. Issue 3. https://ssrn.com/abstract=3798877

[25] Konrad Załęski. 2021. Modeling Concepts. In *Data Modeling with SAP BW/4HANA 2.0*. Springer, 67–96.

[7]https://doi.org/10.5281/zenodo.12706244