ISSN 2303-4521
Periodicals of Engineering and Natural Sciences *Original Research*
Vol. 12, No. 3, October 2024, pp.541-564

Practices for conducting value stream traceability in DevOps: A systematic literature mapping

Daniel Botero[1], Elizabet Suescún[1], César J. Pardo[2]
[1] EAFIT University. Carrera 49 N.° 7 sur 50, Medellín, Colombia
[2] University of Cauca, GTI Research Group. Calle 5 N.° 4-70, Popayán, Colombia

ABSTRACT

DevOps is an approach that describes a set of practices and cultural values whose main objective is to integrate different teams within an organization at the development and operations level. This approach allows for improving the effectiveness of communication, collaboration, value delivery in software development, and organizational culture. Currently, organizations from different economic sectors find themselves in need of developing software and implementing DevOps practices to improve delivery frequency without sacrificing the quality and stability of the solution. Among the main issues when implementing DevOps is how to perform traceability of added value, henceforth, Value Stream. The Value Stream is presented as a sequence of activities responsible for producing and delivering value to customers and users through a product or service. Given its importance for organizations, this work addresses the most common challenges in organizations to perform Value Stream traceability, the proposals presented in the current literature for this process, and the unexplored research challenges found in the literature on the Value Stream. To do this, a systematic literature mapping is developed, resulting in the identification of various practices. From these, a set of challenges and practices that various organizations could implement to perform DevOps process traceability with a focus on the Value Stream are consolidated and presented.

Keywords: Value Stream, DevOps, Software Development, Software Engineering, Agile Framework, DevOps Practices

Corresponding Author:

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

1. Introduction

The term Software Engineering emerged between the 1950s and 1960s, amidst the growing popularity of software development. In the 1970s, pioneers birthed the first software development methodologies, aiming to offer support, clarity, rigor, and structure to the software development process. In the 1980s, developers automated part of the software lifecycle, resulting in the emergence of what we know as the first generation of CASE tools and the spread of object-oriented programming languages. In 2001, industry stakeholders signed the Agile Manifesto with the aim of simplifying the complexity of methodologies by combining the adaptability of agile developments with the formality and documentation of rigorous approaches [1]. Since the 2010s, the DevOps movement has gained momentum [2]. By integrating the key principles of the Agile Manifesto [3], Lean thinking, and the Toyota Kata movement [2], DevOps can be defined as a set of practices and values that aim to bridge the gap between development and operations, encompassing all elements that facilitate the delivery of high-quality software [4]. This is based on the effective implementation of good communication, collaboration, continuous value delivery in software development, and organizational culture [2], [5], [6]. For this reason, companies developing software for systematizing their own or third-party processes seek to adopt DevOps to enhance their software development process. Organizations that have adopted DevOps practices, such as continuous improvement and collaboration between the development team and operations, have shown significant benefits in software development. Results include reduced software delivery times, a lower error rate

per change, shorter incident recovery times, the establishment of seamless communication between the various areas involved in the software development process, and greater adaptation to unforeseen changes in requirements, among others. On the other hand, organizations that have failed to implement DevOps practices in their projects have had less favorable results, characterized mainly by greater bottlenecks in the development process, lack of coordination between the various areas involved, and software deliveries that do not meet the real needs of the client [7]. Additionally, as mentioned in [6], when implementing DevOps practices in organizations with a more change-resistant culture, difficulties arise in identifying improvements in both organizational culture and the value-delivering process flow. These improvements include organizing teams to collaborate towards a common goal, optimizing the use of automation tools to maximize employee productivity, increasing employee motivation by providing a work environment that fosters wellness, and minimizing excessive work hours. Finally, implementing DevOps practices can reduce the introduction of large batches of urgent work in a short time due to poor planning, among other enhancements. Considering the points discussed, we can define value as the benefit perceived by consumers or stakeholders from a product or service [8]. Implementing traceability of added value, also known as Value Stream, emerges as a crucial practice. This practice can significantly contribute to showcasing the advantages of DevOps implementation in both software projects and organizations. Experts define the Value Stream as the series of steps that an organization executes to create and deliver products and services that generate value for its consumers [9], [10], [11]. Additionally, performing traceability of the Value Stream corresponds to the ability to record the evolution of the results of the implemented practices [9], [10], [11], [12], [13], to evaluate, verify, and validate that these practices improve the delivery of value and organizational culture.

The aim of this work is to identify the proposed practices for evaluating, measuring, and tracing the Value Stream in development cycles where DevOps is used. The literature suggests that organizations can gain a comprehensive overview for decision-making and practice adoption by implementing DevOps. Similarly, organizations equipped with DevOps can analyze the achieved results and identify opportunities for process improvement. Taking these considerations into account, the authors structure this work as follows: Section II outlines the methodological process for conducting systematic literature mapping. Section III presents the results of the analysis of the identified primary works. In Section IV, we present a proposal for tracing the Value Stream based on the analyzed works. Section V discusses the conclusions drawn from this study. Section VI outlines avenues for future research. Finally, Section VII addresses potential threats to the validity of the research.

2. Research protocol

This systematic literature mapping is a method for identifying, assessing, and synthesizing existing information on a research topic [14], [15]. Sources [14], [15] establish the procedure and protocol used to conduct the present systematic mapping. Additionally, to follow the protocol and keep track of the selected works, we used the Parsifal tool (https://parsif.al). The process consisted of three stages: Planning, Execution, and Documentation, which we explain below:

2.1. Planning stage

Figure 1 presents the steps conducted in the Planning Stage, and further explanation follows in the subsequent sections.

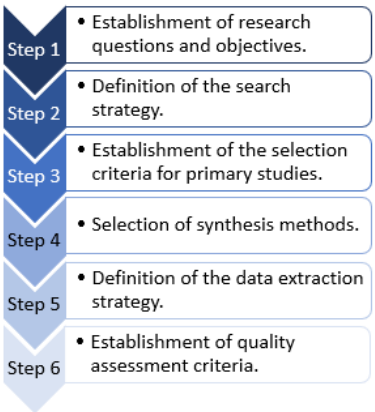

Figure 1. Planning steps undertaken in systematic mapping [15]

2.1.1. Research questions

The main objective of this mapping was to "*establish the proposed practices that allow for tracing the Value Stream in software projects implementing DevOps."* To achieve this goal, we utilized the research questions outlined in *Table 1*. These questions aimed to secure a precise definition for the concept of Value Stream, investigate the prevalent challenges highlighted in the literature concerning the identification of the Value Stream, explore the practices presently employed or suggested to tackle these challenges, and compile information on any unexplored challenges. Subsequently, we generated potential recommendations based on the findings for their practical applicability.

Table 1. Research questions

Research Question	Motivation
Q1. What is the Value Stream in the context of DevOps?	Based on the literature analysis, the objective is to offer an approach to the concept of the Value Stream within the realm of Software Engineering.
Q2. What are the current challenges in tracing the Value Stream in DevOps?	Identifying potential challenges in tracing the Value Stream to concentrate efforts on resolving them.
Q3. Which practices are effective in tracing the Value Stream in DevOps?	Identifying practices that facilitate tracing the Value Stream in DevOps, thereby overcoming challenges prevalent across various sectors of the software development and service industry.
Q4. What unexplored research areas are identified in the literature concerning the verification process of tracing the Value Stream in DevOps within organizational contexts?	Exploring undiscovered research aspects highlighted in the literature regarding the challenges organizations presently encounter when tracing the Value Stream in DevOps.

2.1.2. Search strategy

To conduct the search, we used logical connectors "AND" and/or "OR" with the keywords: *DevOps, DevOps Culture, DevOps Cultural Philosophy, DevOps Framework, Software Engineering, Software Development, Agile Framework, Agile Methodology, Value Stream, Value Traceability, Value Stream Tracking,* and *Value Tracking*.

Based on these keywords, we designed a basic search string: *("Software Development"* ***OR*** *"Software Engineering")* ***AND*** *(("DevOps")* ***OR*** *("Agile Framework"* ***OR*** *"Agile Methodology"))* ***AND*** *("Value Stream"* ***OR*** *"Value Stream Tracking"* ***OR*** *"Value Tracking"* ***OR*** *"Value Traceability").* We adapted this string for each of the following selected databases: ACM Digital Library, Google Scholar, IEEE Digital Library, ISI Web of Science, ScienceDirect, and Scopus. In *Table 2*, we present the adjustments made to the basic string, including the corresponding logical connectors and keywords, to fit each consulted database.

We conducted the search in English, considering information published between 2018 and 2024 (inclusive). We chose this period to include more recent and updated studies.

Table 2. Search strings for each database

#	Database	Search String
1	**ACM Digital Library**	("Software Development" **OR** "Software Engineering") **AND** (("DevOps ") **OR** ("Agile Framework" **OR** "Agile Methodology")) **AND** ("Value Stream" **OR** "Value Stream Tracking" **OR** "Value Tracking" **OR** "Value Traceability")
2	**Google Scholar**	("Software Development" **OR** "Software Engineering") **AND** ("DevOps Culture" **OR** "DevOps Cultural Philosophy" **OR** "DevOps Framework") **AND** ("Agile Framework" **OR** "Agile Methodology") **AND** ("Value Stream" **OR** "Value Stream Tracking" **OR** "Value Tracking" **OR** "Value Traceability")
3	**IEEE Digital Library**	("Software Development" **OR** "Software Engineering") **AND** (("DevOps") **OR** ("Agile Framework" **OR** "Agile Methodology")) **AND** ("Value Stream" **OR** "Value Stream Tracking" **OR** "Value Tracking" **OR** "Value Traceability")

4	**ISI Web of Science**	(“Software Development” **OR** “Software Engineering”) **AND** ((“DevOps”) **OR** (“Agile Framework” **OR** “Agile Methodology”)) **AND** (“Value Stream” **OR** “Value Stream Tracking” **OR** “Value Tracking” **OR** “Value Traceability”)
5	**Science@Direct**	(“Software Development” **OR** “Software Engineering”) **AND** ((“DevOps”) **OR** (“Agile Framework” **OR** “Agile Methodology”)) **AND** (“Value Stream” **OR** “Value Stream Tracking” **OR** “Value Tracking” **OR** “Value Traceability”)
6	**Scopus**	(“software development” **OR** “Software engineering”) **AND** ((“DevOps”) **OR** (“Agile Framework” **OR** “Agile Methodology”)) **AND** (“Value Stream” **OR** “Value Stream Tracking” **OR** “Value Tracking” **OR** “Value Traceability”)

Many companies rely on software to conduct their business activities. While originally conceived in economic sectors outside of software development, these companies have transformed their organizational structures and are now considered software development companies [2]. As this transformation has occurred, the Value Stream has gained greater relevance. The function of the Value Stream is to determine the steps of a process in which improvements can be implemented to deliver value to customers through software in less time and with higher quality [9]. Additionally, it identifies what wastes are present when carrying out a software development process within an organization. In summary, tracing the Value Stream is crucial for ensuring an efficient and high-quality software development process [16].

During the execution of the systematic mapping, we discovered several studies describing various proposals and experiences in implementing processes to trace the Value Stream. The aim is to enhance software development outcomes in organizations. After conducting the search, which involved the strings presented in *Table 2*, we defined a search string for each question. The purpose was to find more targeted results that complemented the initial ones. *Table 3* presents the strings used and their relation to the questions.

Table 3. Search strings for each question

Question	Search String
Q1. What is the Value Stream in the context of DevOps?	((“Value Stream”) AND (“Software”) AND (“DevOps”))
Q2. What challenges currently exist for tracing the Value Stream in DevOps?	((“Value Stream”) AND (“DevOps”) AND (“Software”) AND (“Challenges”))
Q3. What practices can help to trace the Value Stream in DevOps?	((“Value Stream”) AND (“DevOps”) AND (“Software”) AND (“Practices”))
Q4. What unexplored research elements are found in the literature regarding the process of verifying the traceability of the Value Stream in DevOps within an organization?	((“Value Stream”) AND (“DevOps”) AND (“Software”) AND (“further research” OR “Gap”))

2.1.3. Selection criteria for primary studies

The information resulting from the search in each of the different academic databases was evaluated taking into account the following elements: the title, the abstract, and the keywords; this was done to determine whether the study was suitable for inclusion among the relevant studies. Then, the studies were analyzed in detail to select those that met at least one of the criteria in *Table 4*. On the other hand, articles that met any of the exclusion criteria in *Table 5* were not considered.

Table 4. Inclusion criteria

Criterion	Description
I1	Studies detailing practices used in the DevOps process to evaluate, measure, and trace the Value Stream.
I2	Studies that define the Value Stream.
I3	Studies documenting successful DevOps implementations where the Value Stream is evaluated, measured, and/or traced.
I4	Studies describing Value Stream practices in the context of DevOps.

Table 5. Exclusion Criteria. Self-developed

Criterion	Description
E1	Studies that do not provide relevant information to the research on the Value Stream.
E2	Defective, incomplete, or inconsistent studies.
E3	Studies published before 2018.
E4	Duplicate studies.
E5	Studies without full access.

2.1.4. Quality assessment criteria

In addition to ensuring the reliability of the information sources, we measured the quality of the selected studies primarily to utilize them as support for the research and as references for replication or for conducting further studies.

To conduct this quality measurement, we implemented 10 criteria expressed as questions. These criteria, as outlined in [15], address four key aspects: the quality of the report concerning justification, study objectives, and context; the rigor of the research methods employed and the validity of the information; the credibility of the study methods to ensure the authenticity and meaningfulness of the findings; and the significance of the study for the software industry and services. The listed criteria are as follows:

C1. Was the document research-based or is it primarily a "lessons learned" based on expert opinion?
C2. Were the research objectives clearly established?
C3. Is there a sufficient description of the research's contextual background?
C4. Was the research design appropriate for addressing the research objectives?
C5. Was there a control group with which to compare treatments?
C6. Were the data collected in a way that addressed the research problem?
C7. Was the data analysis rigorous enough?
C8. Was researcher and participant bias taken into account in the study?
C9. Is there a clear exposition of the findings?
C10. Is the study valuable for research or practice?

In [15], the GRADE approach notes that it establishes four levels of evidence of study quality according to the criteria (very low, low, moderate, and high). For this case, we established a scale with values of 0.1, 0.4, 0.7, and 1, respectively assigning them. The sum of the score for each question will constitute the final quality score of each study (resulting in a value between 1 and 10). *Table 6* presents the result of the evaluation of the studies according to the quality assessment criteria.

Table 6. Evaluation of studies based on quality assessment criteria

	Quality issues										
Ref.	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**Total**
[2]	1	1	1	1	0.1	1	0.4	0.1	1	1	7.6
[8]	1	1	1	1	1	1	1	1	1	1	10
[9]	0.4	1	0.1	0.1	0.1	0.4	0.7	0.7	1	1	5.5
[10]	1	1	1	1	1	1	1	1	1	1	10
[11]	1	1	1	1	1	1	1	1	1	1	10
[17]	0.4	0.7	0.7	0.7	0.4	0.7	0.7	0.7	1	1	7
[18]	0.7	1	1	1	0.7	1	0.7	1	1	1	9.1
[19]	1	1	1	1	0.4	0.7	0.4	0.4	1	1	7.9
[20]	0.7	1	0.7	1	0.4	0.4	0.4	0.4	1	1	7
[16]	1	1	1	1	0.4	0.7	0.1	0.1	1	1	7.3
[21]	1	1	1	1	0.7	0.4	0.4	0.4	1	1	7.9
[22]	1	1	1	0.7	0.7	0.7	0.7	0.4	1	1	8.2
[23]	1	1	1	1	0.4	1	0.7	0.7	1	1	8.8
[24]	0.4	1	0.7	0.7	0.4	0.4	0.7	0.4	1	1	6.7
[25]	1	1	0.7	1	0.4	0.7	0.7	0.4	1	1	7.9
[26]	1	1	1	0.7	0.7	0.7	1	0.7	1	1	8.8
[27]	1	0.4	0.7	1	0.4	0.7	0.7	0.4	1	1	7.3

Quality issues											
Ref.	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**Total**
[28]	0.7	1	0.7	0.4	1	0.7	0.4	0.7	1	1	7.6
[29]	1	0.7	0.7	0.4	1	0.7	0.7	0.4	1	1	7.6
[30]	1	1	0.4	0.7	1	0.4	0.4	0.7	1	1	7.6
[31]	1	1	0.7	0.4	0.7	0.4	0.7	0.7	1	1	7.6
[32]	1	0.7	0.7	0.4	1	0.4	0.7	1	1	1	7.9
[33]	0.7	1	0.4	0.4	0.7	0.4	0.7	0.7	1	1	7
[34]	1	1	0.7	0.7	0.4	0.7	0.7	0.4	1	1	7.6
[35]	0.4	1	0.7	0.7	0.4	0.4	0.7	0.4	1	1	6.7
[36]	1	1	0.4	0.7	1	0.4	0.4	0.7	1	1	7.6
[37]	1	0.4	0.7	1	0.4	0.7	0.4	0.4	1	1	7
[38]	1	1	0.7	1	0.4	0.7	0.7	0.4	1	1	7.9
[39]	1	0.4	0.7	0.4	1	0.4	0.7	0.4	1	1	7

2.1.5. Data extraction strategy

The strategy aimed to apply consistent data extraction criteria to all selected studies, simplifying their classification through the use of categories. This allowed the information found to be categorized based on the question being answered. The categories used for each question are presented in *Table 7*.

Table 7. Classification scheme

Question	Response Categories
Q1. What is the Value Stream in the context of DevOps?	a. Definitions of the Value Stream. b. Characteristics of the Value Stream. c. Types and classification of the Value Stream.
Q2. What challenges currently exist for tracing the Value Stream in DevOps?	a. Challenges in the use of practices. b. Cultural challenges. c. Management challenges.
Q3. What practices can help to trace the *Value Stream* in DevOps?	a. Practices. b. Success cases.
Q4. What unexplored research elements are found in the literature regarding the process of verifying the traceability of the *Value Stream* in DevOps within an organization?	a. Limitations found. b. Research proposals in consulted studies. c. Elements identified in the consulted literature that can be further explored.

2.1.6. Synthesis methods

We conducted the reading and analysis of the studies. This analysis structured around the following parameters: identification (title, author, year), abstract, and contributions relevant to answering research questions such as: "What is the Value Stream?", "What challenges currently exist for tracing the Value Stream in DevOps?", "What practices can help to trace the Value Stream in DevOps?", and "What unexplored research elements are found in the literature regarding the process of verifying the traceability of the Value Stream in DevOps within an organization?" *Table 8* presents the list of selected studies (totaling 29), along with their contributions to addressing the posed questions.

Table 8. Contribution of primary studies

Research Questions									
Ref.	**Q1**	**Q2**	**Q3**	**Q4**	**Ref.**	**Q1**	**Q2**	**Q3**	**Q4**
[2]	X				[26]	X	X	X	X
[8]	X	X	X	X	[27]	X			
[9]	X				[28]	X		X	
[10]	X				[29]	X	X	X	
[11]	X				[30]	X		X	
[17]	X				[31]	X	X	X	
[18]		X	X		[32]		X		

Research Questions									
Ref.	**Q1**	**Q2**	**Q3**	**Q4**	**Ref.**	**Q1**	**Q2**	**Q3**	**Q4**
[19]	X	X	X	X	[33]	X	X	X	
[20]	X				[34]	X	X		
[16]		X	X		[35]	X		X	
[21]		X	X		[36]	X			
[22]	X				[37]	X			
[23]	X		X		[38]			X	
[24]		X	X	X	[39]	X	X		
[25]		X							
Total	**Q1**		**Q2**		**Q3**		**Q4**		
	22		14		15		4		

2.2. Execution stage

The process began with a context search, conducted in a generic manner without specifying the databases mentioned in item 2 of the research protocol. This was done to gather initial information that could aid in refining the search strategy. Subsequently, we conducted 4 iterations to fine-tune the search strings for each of the databases. *Table 9* presents the studies obtained after executing the search strings in the databases and the number of studies from other sources.

Table 9. Classification scheme

#	Source	EN	RE	DU	RESACC	Total
1	**Other Sources**	8	8	0	0	3
2	**ACM Digital Library**	39	7	4	0	3
3	**Google Scholar**	39	15	1	5	9
4	**IEEE Digital Library**	12	8	6	0	2
5	**ISI Web of Science**	4	2	1	0	1
6	**Science@Direct**	43	5	0	0	5
7	**Scopus**	8	4	3	0	1
8	**General total**	153	49	15	5	29
Acronyms: Found (EN), Relevant (RE), Duplicates (DU), Relevant, but without access (RESACC)						

Figure 2 shows the selection process with the number of studies selected for each stage of the systematic mapping.

Figure 2. Systematic mapping results for each of its stages

3. Results

Below are the results obtained for each of the defined research questions.

3.1. Question 1: What is the Value Stream? (75.86%)

To answer this question, literature defining the concept of the Value Stream was found in *75.86%* of the works ([2], [8], [9], [10], [11], [17], [19], [20], [22], [23], [26], [27], [28], [29], [30], [31], [33], [34], [35], [36], [37], [39]). These definitions are presented below:

The Value Stream is defined as a series of steps that an organization executes to create and deliver products and services that generate value for its consumers [9], [10], [11]. In this context, traceability corresponds to tracking the Value Stream with the aim of finding improvement opportunities, evaluating elements such as quality, waste reduction, and rework, among other aspects. In [30], it is indicated that it also includes activities for reducing effort and optimizing resources related to delivery and/or increasing value as progress is made in the flow. In [10] is described the difference between the Value Stream and processes. Processes are a set of steps to transform inputs into defined and predictable outputs; on the other hand, the Value Stream goes further, monitoring the various activities that generate waste within a process, with the aim of eliminating it and improving the productivity of the value flow, which in turn creates more value and at a faster pace [17], [28], [29], [31], [33].

Furthermore, in [22], the definition of the concept is elaborated upon by outlining four interacting elements within the Value Stream. This reaffirms that the concept extends beyond the organization's software development processes, integrating people, technology, and data. These elements are delineated in Figure 3.

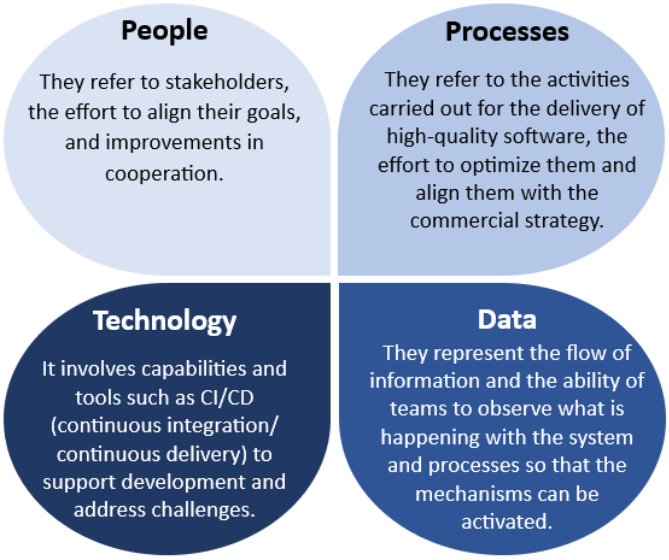

Figure 3. Elements that interact in the value stream [22]

The concept of Value Stream stems from the principle of waste elimination, pioneered by Toyota in the 1940s as part of the TPS (Toyota Production System) [35]. Later, James Womack and Daniel Jones introduced the term Lean in 1990 [37]. Subsequently, in [36], they delineated the principles of Lean Thinking, which involved identifying value and the value chain, albeit in the context of manufacturing tangible products. Finally, in 2003, Mary Poppendieck and Tom Poppendieck introduced Lean Software Development as an adaptation of Lean principles to Software Engineering [35]. From this amalgamation of principles, waste elimination emerged as the cornerstone of Lean thinking in the context of software development. In this regard, DevOps emerged as the convergence of various movements, including Lean, from which it adopted the principle of waste elimination and the concept of Value Stream [2].

Revisiting the concept, the principle of waste elimination dictates that anything not adding value to the customer is considered waste [35], In other words, it signifies negative value [19], reflecting a loss of invested effort. According to [20], activities within the Value Stream are categorized into the following three groups in relation to waste:

a) Activities that add value to the product and are therefore legitimately part of the Value Stream include tasks such as creating code for new features.

b) Activities that do not add value to the product but are necessary to manufacture it. These activities remain in the Value Stream because, otherwise, the product could not be manufactured, such as activities that can be replaced by other improved activities. For example, manual deployments by automatic application deployments.
c) Activities that clearly do not add value to the product or that generate rework must be promptly removed from the Value Stream, as they are non-essential. This process corresponds to waste elimination. An example of such activities would be completing extensive forms for deployment approval.

To classify activities correctly, it is necessary to clearly define their values, prioritizing what generates value over the outcome at the end of the production process [8], [23], [26], [27], [34], [39].

In [34], the author defines two characteristics of value, which we describe below:

Visibility

The concept of visibility encompasses two aspects: visible value and invisible value. Invisible value refers to features that users often take for granted but require significant effort from developers to implement within the system. For example, this could include a new feature for online payments with improved response time. Conversely, visible value encompasses features that users directly perceive as value delivered by the system, such as encrypting information to ensure system privacy.

Type of Delivered Value:

There are two distinct types of value: positive value and negative value. Negative value encompasses unfavorable elements that diminish the benefit for stakeholders; consequently, any element exhibiting this characteristic must be eliminated. For example, a blocking error that prevents users from accessing and utilizing a feature falls under negative value. In contrast, positive value comprises elements that fulfill the needs of the customer and stakeholders. An example of this is the implementation of new technology to optimize processes within the system, which adds positive value.

According to [34], the combination of these two characteristics (visibility and type of delivered value) results in 4 new types of value: new functionalities, defect correction, architecture improvement, and system risk management. These are further explained in *Table 10* below:

Table 10. Types and classification of value [34]

Value	Positive	Negative
Visible	**New functionalities:** New functionalities are incorporated into the system to meet customer needs.	**Defect correction:** Errors in the system that prevent the fulfillment of customer needs are rectified.
Invisible	**Architecture improving:** The system or a portion of it undergoes refactoring to enhance the capability for quicker code modification and maintenance. This facilitates the incorporation of visible value more effectively in the future.	**System risk management:** This involves monitoring and managing various types of risks. While end users may not directly observe or be highly aware of these risks, their materialization results in a significant loss of perceived value for the user.

In summary, the Value Stream comprises a series of steps that an organization undertakes to create and deliver products and services that generate value for its users. Defining the value of the elements within the Value Stream is vital, as this concept is linked to the identification and elimination of waste, along with the optimization of activities necessary to generate value.

3.2. Question 2: What challenges currently exist in tracing the Value Stream in DevOps? (48.27%)

To address this question, we present several challenges associated with tracing the Value Stream, as identified in *48.27%* of the primary studies [8], [16], [18], [19], [21], [24], [25], [26], [28], [29], [31], [32], [33], [34].

To start, as outlined in [18], the absence of information during strategic decision-making can impede the tracing of the Value Stream process. Moreover, as noted in [16], a critical challenge arises from the limited visibility and unreliable measurement of software delivery practices within many organizations, resulting in a dearth of

decision-making information. Furthermore, according to [21], a significant obstacle in measuring the success of DevOps and the Value Stream is the absence of a baseline for metrics, compounded by the lack of a defined Value Stream process, as highlighted in [29], [31], [32].

The initial recommendation, proposed in both [16] and [21], emphasizes the importance of constructing a baseline for the Value Stream to establish a reference point for periodically assessing progress. As explained in [16], while this process may entail a lengthy journey, it is imperative to start in order to gain visibility into improvement opportunities across the Value Stream.

Another challenge lies in the current implementations of Value Stream Mapping, which primarily concentrate on delineating the steps and coordinating software development information. However, as noted in [26], these implementations often lack the capability to model or track the final value of elements traversing through the Value Stream Map. Consequently, organizations may find themselves analyzing the process based on significant assumptions regarding the value of the delivered software product, thereby hindering the optimization of their software development practices towards value.

In [25], the discussion revolves around the challenge of selecting effective metrics for Value Stream traceability. It highlights that popular metrics, such as the number of lines of code written, may not necessarily provide the requisite information for decision-making. Focusing solely on this metric may result in increased code volume, yet it does not ensure enhanced value or quality. Similarly, metrics like code coverage in testing measure the code covered by tests but do not guarantee effectiveness in error prevention. Furthermore, it warns against the overemphasis on metrics, such as velocity metrics, which can potentially lead to a decline in desired performance for the selected metrics.

Another challenge highlighted in the literature is the lack of a clear definition of what flows through the Value Stream. This includes determining what is being developed, such as new features, defects, among other elements. As emphasized in [34], it is imperative to define the type of value associated with each element processed within the organization (refer to *Table 10*) to accurately evaluate the Value Stream.

One of the challenges highlighted in [8] is the need to comprehend the Value Stream beyond the product or finalized functionalities and anticipate future scenarios to ensure a better implementation during the construction of the initial version. This scenario is more prevalent in the software industry than it may appear. For instance, while there is pressure to add value to the product rapidly, either due to market competitiveness or commitments to clients, the emphasis may not be placed on employing the best architecture and development practices during the delivery of specific software functionalities. Instead, there may be reliance on the promise of future refactoring. As explained in [8] in such cases, typically only the amount of effort expended in creating the functionality and the effort invested in its initial version are taken into account, with little consideration given to the future costs involved.

Refactoring, or the expenses associated with maintaining software with subpar architecture and development practices, is crucial to acknowledge. As discussed in [40], this concept is referred to as technical debt, which can be categorized using Fowler's quadrant to delineate the different types of technical debt and their impacts on long-term software development. Conversely, [8] advocates for considering not only the immediate effort but also the future costs of decisions made at present.

Another significant challenge lies in the failure to identify negative value concepts associated with waste, which can lead to underestimating their importance within organizations [19], [33]. A primary challenge for traceability is defining the measurement of value or waste: Key Performance Indicators (KPIs) must be established, but their development requires careful consideration to avoid incurring unnecessary costs. It is crucial to define what is intended to be measured in an organization and why within this context. The findings of this study suggest a focus on measuring the product and the process, but not the people, which can obscure the needs of team members and opportunities for improvement in the human aspect [8], [19], [24]. Additionally, individuals often encounter difficulties in understanding what waste is, how to identify it, how to measure and quantify its impact, and how to reduce waste in general [19].

Next, *Table 11* presents a consolidation of the challenges organizations face when tracing the Value Stream. These challenges fall into two approaches: the first emphasizes the organization's lack of information about its development process; and the second focuses on the challenges of implementation within an organization, exposing weaknesses in this process.

Table 11. Summary of value stream challenges

Challenges of the Value Stream	
Related to information	Insufficient information for making strategic decisions.
	Lack of a measurement baseline or definition of the Value Stream.
	Failure to clearly define the elements that flow through the Value Stream.
	Defining the measurement of KPI value: it is important to specify what an organization intends to measure and the reasons behind it.
Related to implementation	Focusing on measuring the steps rather than tracing the value.
	Neglecting to consider the technical debt incurred as a result of a development decision; this is common when there is a pressing need to deliver value to the customer promptly.
	Neglecting to consider waste and its negative value within the Value Stream.
	Relying on popular metrics that may not necessarily offer pertinent information.
	Not measuring and making the human factor invisible.

3.3. Question 3: What practices can help in tracing the Value Stream in DevOps? (51.72%)

To address the answer to this question, we first present some general recommendations found in the literature, followed by a brief explanation of proposed practices for tracing the Value Stream. These recommendations are derived from *51.72%* of the works identified in the research [8], [16], [18], [19], [21], [23], [24], [26], [28], [29], [30], [31], [33], [35], [38].

In [16], the authors explain the importance of measuring elements such as the error rate per change and the delivery time of a new requirement in DevOps. The first step indicated is to build a baseline: starting measurement practices early allows for the rapid availability of a baseline to compare relative improvements when implementing actions. Capturing data early is essential; therefore, they recommend doing so while developing and refining metrics in the organization.

In [26], the authors recommend classifying all work performed and focusing on the primary objective of each task as a fundamental and essential practice for achieving effective traceability of value. This approach makes it easier to determine if the expected value is being delivered. Additionally, they advise considering waste, categorized as negative value, as tasks that do not contribute to satisfying customer [19], [28], [33].

To categorize waste in the development process, one can utilize the classification proposed by Lean Development, which consists of 7 categories: partially completed work, extra processes, extra features, task switching, delays and waits, lack of transparency, and defects [35]. Below, in *Table 12*, the types of waste described in [35] are explained. These are pertinent to the current proposal, as waste elimination is closely linked to the concept of the Value Stream, as mentioned previously.

Table 12. Waste categories. adapted from [35]

#	**Waste Categories**	
1	**Partially completed work**	Incomplete work that fails to add value, becomes obsolete, and remains unintegrated into the rest of the project: This can create dependencies for future development and pose a financial risk due to the inability to complete it.
2	**Additional processes**	Excessive approval processes and documentation: Documentation artifacts are deemed useful only if someone is awaiting them to perform a task, such as coding, testing, or drafting training manuals; otherwise, they are likely waste. It is important to note that the mere requirement of a process within the organization does not inherently add value.
3	**Additional features**	Adding features that the client did not request can divert the team's efforts from meeting the customer's needs. If something is not needed now, consider it as waste when integrated into the system.

#	Waste Categories	
4	**Task switching**	Assigning people to multiple projects at once constitutes a source of waste: Every time a software developer switches between tasks, they incur significant switch time as they gather their thoughts and get into the flow of the new task. A developer who belongs to multiple teams generally experiences more interruptions and thus more task switches. For instance, the most efficient way to complete two projects with the same team is to focus on one project at a time.
5	**Delays and waiting**	Delays must be minimized. Examples of this include delays due to excessive requirement documentation, delays in reviews and approvals, delays in testing, and delays in implementation, all of which constitute waste. When a critical customer need reaches a development team, the team's ability to respond quickly is directly impacted by the systemic delays in the development cycle.
6	**Lack of transparency**	All team members must be familiar with and understand all relevant elements of the development process. Furthermore, establishing seamless communication within the team and with various stakeholders should be effortless. For example, if a developer has a business-related question, they should have convenient access to the business owner to address it. Hence, it is advisable for teams to comprise members from all relevant areas of interest. However, there should be multiple sources of information. Relying solely on documentation as the primary source of information results in much of it remaining with the creator of the artifact, such as a requirements document or an impediment log.
7	**Defects**	The sum of the defect's impact plus the time it goes unnoticed defines the amount of waste caused by a defect; thus, a critical defect detected in three minutes is not a significant source of waste, while a minor defect that goes undiscovered for weeks represents a much greater waste. The recommendation to reduce the impact of defects is to find them as quickly as possible.

In this section, we briefly explain some of the proposed practices for performing Value Stream traceability obtained from the primary analysis:

3.3.1. Using kanban boards

Using Kanban boards has been recognized as an effective practice for managing identified waste, particularly in representing elements that impede workflow [19]. The creation of a Kanban board serves to visualize workflow, aiming to constrain work in progress (WIP) to the team's capacity. WIP refers to the number of tasks that a team is concurrently managing. It is essential that the WIP does not surpass the established limit. Occasionally, a task in progress may encounter delays due to internal or external impediments. Traditionally, in such scenarios, new tasks are initiated, thereby adding to the WIP. However, when utilizing Kanban, this is discouraged as it violates the WIP limit. Consequently, the impediment hindering the current task must be addressed to complete it and free up capacity for other tasks [38]. Setting a limit on WIP enables users to pinpoint where impediments arise, such as the notorious bottleneck in the Value Stream.

To create Kanban boards, a team divides a board into columns representing the stages of the process. The team typically includes stages such as To Do, In Progress, and Done, but they can adjust them to their way of working by adding, removing, or renaming stages. Then, the team describes the tasks on cards, and they define the WIP limit. The team determines the WIP limit based on their experience, so they start with an initial limit that they can adjust whenever they have a reason to do so based on experience [38]. Figure 4 illustrates an example of a Kanban board where there is a team of two developers and it allows only one task in progress per developer.

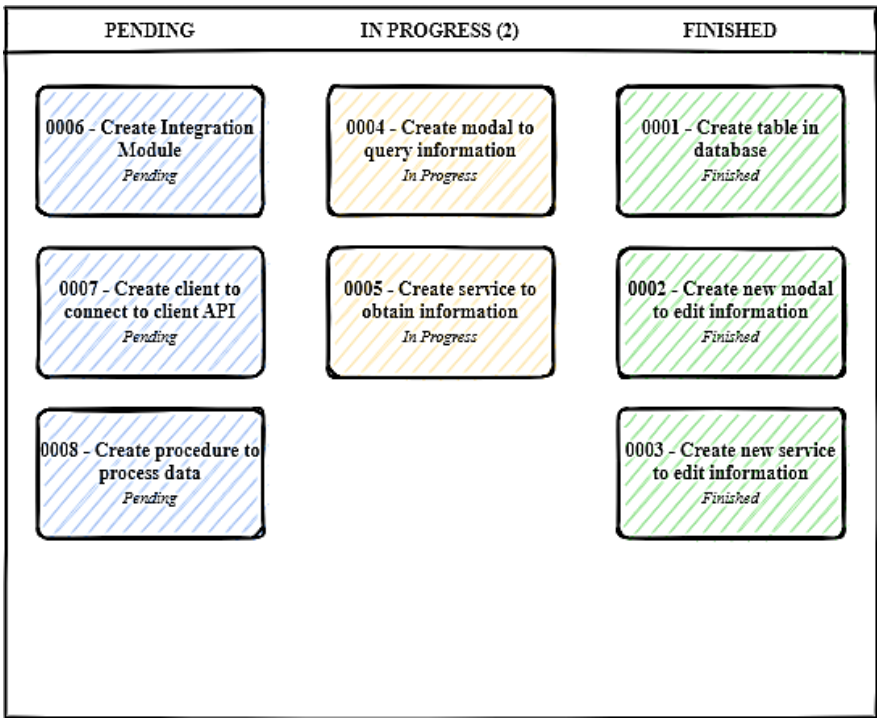

Figure 4. Kanban Board Example [38]

In Figure 4 it is noteworthy that there are currently two tasks in progress. Therefore, if you intend to initiate task 0006 - *Create integration module*, you must first complete one of the tasks in progress to release development capacity for the new task.

In summary, while Kanban proves effective in visualizing and organizing current work, thereby aiding in minimizing non-value-added work, it does not inherently prevent waste from permeating. This work also advocates for Value Stream Mapping (VSM) as a beneficial practice for waste identification [19].

3.3.2. Value Stream Mapping (VSM)

In [23], Value Stream Mapping (VSM) serves as a practice for visualizing waste within processes and their step-by-step progression, as outlined in [29], [30], [31]. With a Value Stream Map of the software development and delivery process, bottlenecks can be identified at various stages of the process. These bottlenecks are key points where improvements should be focused within the Value Stream, preventing efforts from being concentrated in the wrong place. To identify bottlenecks, the Value Stream can be mapped, and a vote can be conducted among stakeholders where they express, from their perspective, where bottlenecks are occurring in the Value Stream and which ones are the most critical. According to [21], consensus among the majority tends to be effective in identifying critical points in the process. When conducting VSM, the aim is to understand how much time it takes to create a product and what proportion of the elapsed time is dedicated to adding value. Seeing the landscape in this way helps identify delays [23].

3.3.3. Combine Metrics Obtained from Systems with Surveys Conducted with Stakeholders

According to [16], many organizations in DevOps utilize two practices for measuring the Value Stream: conducting periodic surveys with stakeholders and collecting data from various systems. However, [16] explains that neither practice alone yields effective results; instead, both should be implemented together to achieve effectiveness. By combining surveying with data collection from systems, a more comprehensive approach is achieved, enabling a broader measurement of both system performance and the work environment [16]. Next, in *Table 13* and T*able 14*, each of the various types of metrics obtained in this combined practice is analyzed, defining their important aspects, advantages, and challenges.

As per [16], while system-based metrics offer utility, synthesizing the organization's software delivery process requires considerable effort. Therefore, it is advisable to complement these metrics with surveys for a more comprehensive understanding. Ultimately, as concluded in [16] metrics based on the system and survey-based

metrics each possess inherent limitations. However, by employing both types of metrics in a complementary measurement program, organizations can achieve a more comprehensive understanding of their Value Stream and the DevOps adoption process. Furthermore, for the current study, *Table 13* and *Table 14 hold* significance as they delineate the characteristics of various metrics.

Table 13. Metrics based on system data. adapted from [16]

Metrics Based on System Data		
Description		Metrics based on the system encompass data derived from various logging systems integrated within the software delivery value stream.
Considerations	**Complete data**	It is essential to ascertain whether the data obtained from the system contains sufficient information for a thorough analysis.
	Integration between systems	Mastering the integration of data from diverse systems is crucial for accurately generating meaningful insights.
	No duplication and correct association	It is worth noting that when records exist across different systems, efforts should be made to track work items effectively, prevent duplication of information, and ensure accurate associations.
Advantages	**Technical accuracy**	The data can be generated with precise time measurements.
	Continuous visibility	The data can be displayed on dashboards that enable continuous monitoring.
	Granularity	The data can be readily collected by component and/or subsystem, facilitating the identification of bottlenecks at specific points. However, it may pose challenges when attempting to generate a comprehensive overview of the system.
	Scalability	An organization can initiate a Proof of Concept (POC) with a project, and upon successful standard implementation, scale it to the rest of the projects within the organization.
Challenges and Obstacles	**Capture behavior outside the system**	This could be considered the most significant limitation, often overlooked, in data derived from the system. For example, the version control system can only capture what resides within it; any work outside of it, such as system configurations and database scripts, will not be accounted for.
	Difficulty in having a 360-degree view	While it is feasible to construct comprehensive measurement models based on system data to offer a holistic perspective, the process of building them is not straightforward technically. It demands a high level of maturity from the human side of agile methodology. Furthermore, integrating measurements of human factors into the system poses considerable challenges.
	Adapting to system changes can be costly	In the event of system changes, failing to update the data collection functions can lead to inaccurate measurements. Adapting the data collection implementation to reflect these changes may require a significant investment of resources.
	Cultural or perceptual measures	Cultural aspects are inherently perceptual and are best measured through surveys. When measured from a system, one may be analyzing lagging data, resulting in delayed reactions to identified events as one is essentially responding to past occurrences. Conversely, survey results enable the measurement of cultural perceptions in real-time, providing current information for timely action.

Table 14. Survey-based metrics. adapted from [16]

Survey-based Metrics		
Description		Survey-based metrics encompass data derived from surveys, capturing insights about both systems and people, including aspects like organizational culture. These metrics are utilized to assess the performance and experiences of individuals within the system.
Considerations	**Cohesion of results**	Survey-based data offers a comprehensive perspective of systems as it can gather information about systems, processes, and culture. It is advisable to conduct system measurements periodically, ideally every four to six months, to ensure a consistent understanding of the system's dynamics.
	Question design	The design and measurement of surveys represent well-understood disciplines that can effectively provide accurate data and insights about systems and culture. By crafting surveys with statistically valid and reliable questions, developed and tested rigorously, organizations can trust in the credibility of their results.
Advantages	**Human perspective**	By posing pertinent questions, surveys can yield precise data concerning systems, processes, and culture, encompassing aspects like task frequency and the assessment of the work environment, among others.
	A comprehensive view	These data facilitate proactive decision-making rather than reactive responses. Surveys prove invaluable for grasping the holistic state of the organization, as they allow for targeted emphasis on various aspects through the questions posed.
	Triangulation with system data	Factors may exist that disrupt system measurements and store outdated data. Conversely, surveys offer an alternative perspective, providing a more current and therefore more accurate depiction compared to the system's viewpoint. In instances of discrepancies between information provided by both sources, it is advisable to review the latest software deliveries from the system before reaching a decision.
	Capture data outside the system	Unlike measurements based solely on system data, surveys have the capability to unveil situations occurring beyond the system's confines. For example, they can shed light on elements of organizational culture, involvement of actors not directly engaged in processes, team knowledge levels, and other pertinent factors.
	Cultural or perceptual measures related to the system	Surveys offer a means to gather data that may be more challenging to ascertain directly from the system. For example, they provide insights into crucial indicators such as the cadence of work pace and factors affecting hiring and staff retention, including organizational culture, job satisfaction, and burnout. Surveys can also uncover issues like a dysfunctional application component, instability in the agile development process within a team, or problems with organizational facilities, among other considerations.

3.3.4. Relating software elements to strategic objectives

In [26] and [18], emphasis is placed on the importance of aligning software elements with strategic objectives. This alignment enables organizations to ascertain whether the developed software contributes to commercial value. By tracking commercial results, organizations can demonstrate the revenue generated through the developed software.

To align software elements with strategic objectives, the process begins by defining development components, including functional features, defect management, risk mitigation, and addressing technical debt. These components can be documented in a work tracking tool [26]. Subsequently, they are linked to the organization's strategic objectives. The goal is to gain a clear understanding of how each development component contributes to achieving specific objectives, with the potential for a single component to contribute to multiple objectives. As illustrated in [18], a hospital in Belgium serves as an example, where a functional feature aids in several strategic objectives simultaneously.

In Figure 5, we observe how a shift in business circumstances gives rise to a strategic opportunity, comprising multiple strategic objectives, each with a distinct contribution from a specific characteristic.

traceability of the Value Stream.

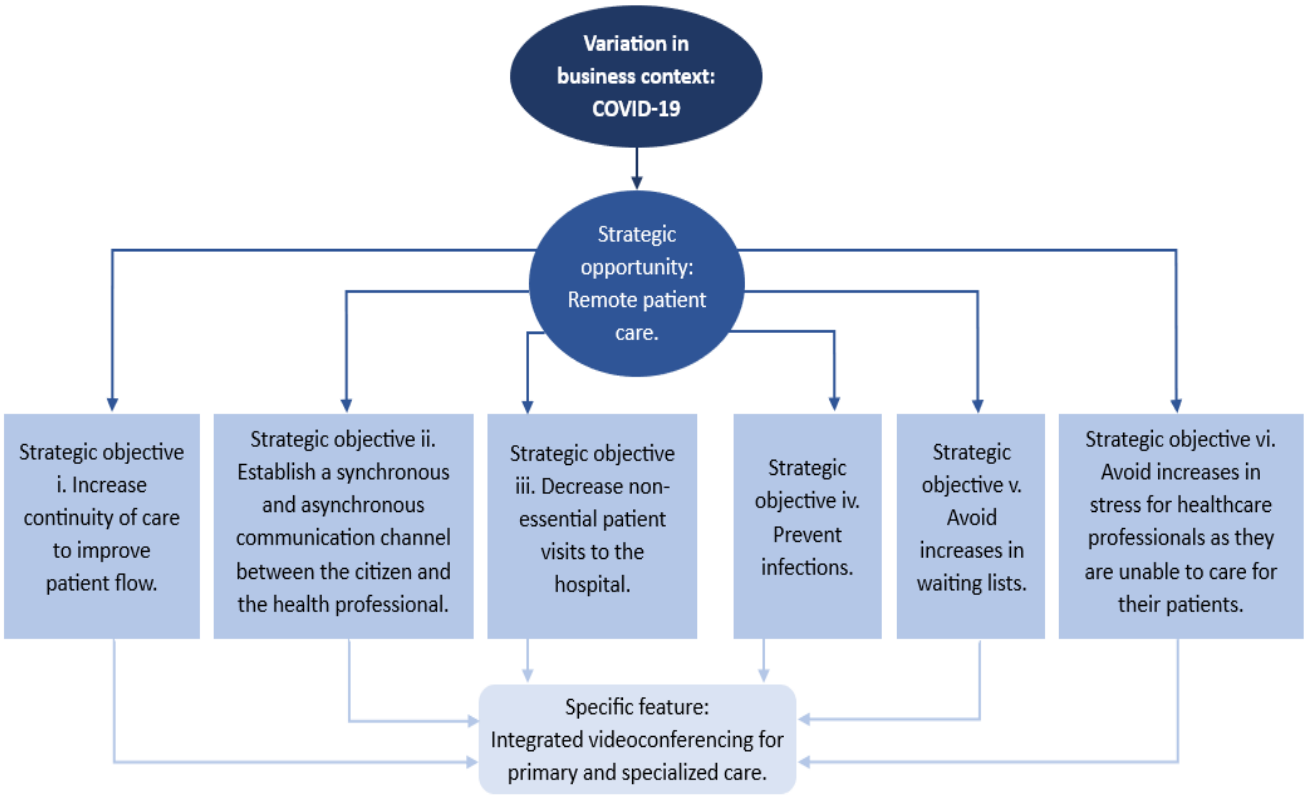

Figure 5. Example of association of software features with strategic objectives [18]

To explain the concept outlined in [18], Figure 5 illustrates the feature as a means to attain a strategic objective. The aim is to monitor the specific functional realization of the particular goal through one or more features [18]. This illustrates how identifying various metrics associated with a feature, such as delivery time or the number of errors, can impact different strategic objectives. It also helps identify whether the features selected for development meet customer needs. For instance, in this approach, if a feature fails to contribute to a strategic objective, it may be considered wasteful, prompting a reassessment of its necessity. If deemed unnecessary, it could be eliminated from the development pipeline. This process refines the list of tasks for the development team, enabling them to focus their capacity on elements that align with customer needs. In essence, this practice validates whether each work item adds value to customer needs and identifies the objectives associated with each work item.

Various proposed practices for tracing the Value Stream were discussed in this response. *Table 15* presents a summary of these practices.

Table 15. Summary of practices that can help to achieve traceability of the value stream in DevOps

Practices that can help to perform the traceability of the Value Stream in DevOps	
Create a baseline of the Value Stream	Establishing a baseline provides a reference point for comparing the improvement in DevOps implementation.
Classify work items	We must classify elements, enabling us to focus their measurement on relation to the type of value they deliver.
Kanban boards	These boards enable visualizing work and restricting work in progress to encourage collaborators to focus on a single task and identify blocks in the development process.
Value Stream Mapping (VSM)	It is a map of the various activities involved in developing a value element, which facilitates the identification of waste in the development process.
Combine metrics obtained from systems with surveys	It is a practice that involves measuring the state of an organization from two perspectives: utilizing system data and conducting surveys with the team and

Practices that can help to perform the traceability of the Value Stream in DevOps	
conducted with stakeholders	stakeholders to gather information on software performance and the work environment.
Associate software elements with strategic objectives	It is a practice that enables associating each work element with a commercial strategic objective to quantify the impact of that element on the business.

These practices can collaboratively measure and trace the Value Stream in a broader and more comprehensive manner. Utilizing Kanban boards aids in waste identification. With Value Stream Mapping (VSM), waste can be quantified and prioritized based on its impact. By associating work items with strategic objectives, it's feasible to comprehend which part of the business is impacted by waste. Lastly, metrics can assist in identifying improvement opportunities in both the system and the development methodology and team environment.

3.4. Question 4: What unexplored research elements are present in the literature regarding the verification process of tracing the DevOps Value Stream within an organization? (13.8%)

Only *13.8%* of the works addressed the posed question directly among the mapped studies: [8], [19], [24], [26]. offer suggestions for future research or explicitly delineate limitations, gaps, or opportunities pertaining to the

According to [24] and [8], many studies highlight the significance of highly skilled personnel and their influence on the ultimate quality of the organization's software product. However, as noted in [8], none of these studies present practical approaches beyond surveys to evaluate or measure compliance with these value considerations in DevOps.

Another unexplored research element uncovered is that, despite the presentation of certain practices in the previous question, such as utilizing Kanban boards and VSM to address waste in tracing the Value Stream, questions such as *"How can waste be measured in a manner that considers all the various influencing factors?"* and *"How can metrics be developed to assist organizations in measuring both the impact of waste and the effectiveness of waste elimination activities?"* necessitate further investigation, as suggested [19].

Furthermore, as explained in [26], the classification of work into various types of value stands out as one of the most critical tasks in achieving Value Stream traceability, demanding significant time and effort. A promising research concept proposed in [26] involves the potential for automating the work classification process using a machine learning system. This innovation could enhance the efficiency and effectiveness of Value Stream traceability, mitigating the need for extensive time and effort investment in this task.

In conclusion, this study has highlighted several unexplored research elements pertinent to Value Stream traceability, shedding light on the challenges encountered in the software development industry. *Table 16* provides a succinct summary of these elements.

Table 16. Unexplored research elements in the literature for conducting value stream traceability in DevOps

Unexplored Research Elements in the Literature for Performing Value Stream Traceability in DevOps.	
Absence of practices to measure the human factor.	Insufficient methodologies beyond surveys for assessing the human factor within organizations.
Measuring waste including its environment.	Insufficiency in methodologies that comprehensively measure waste, encompassing all elements within the process and considering all impacted factors.
Automating classification.	Investigate the feasibility of automating the work labeling process using a machine learning system.

The analysis of various works indicates that further research is necessary to facilitate organizations' adoption of Value Stream traceability practices. Therefore, the following section presents a consolidated proposal based on the findings and literature analysis.

4. Proposal

Next, Figure 6 describes a proposal for performing Value Stream traceability by employing some of the practices mentioned in the research.

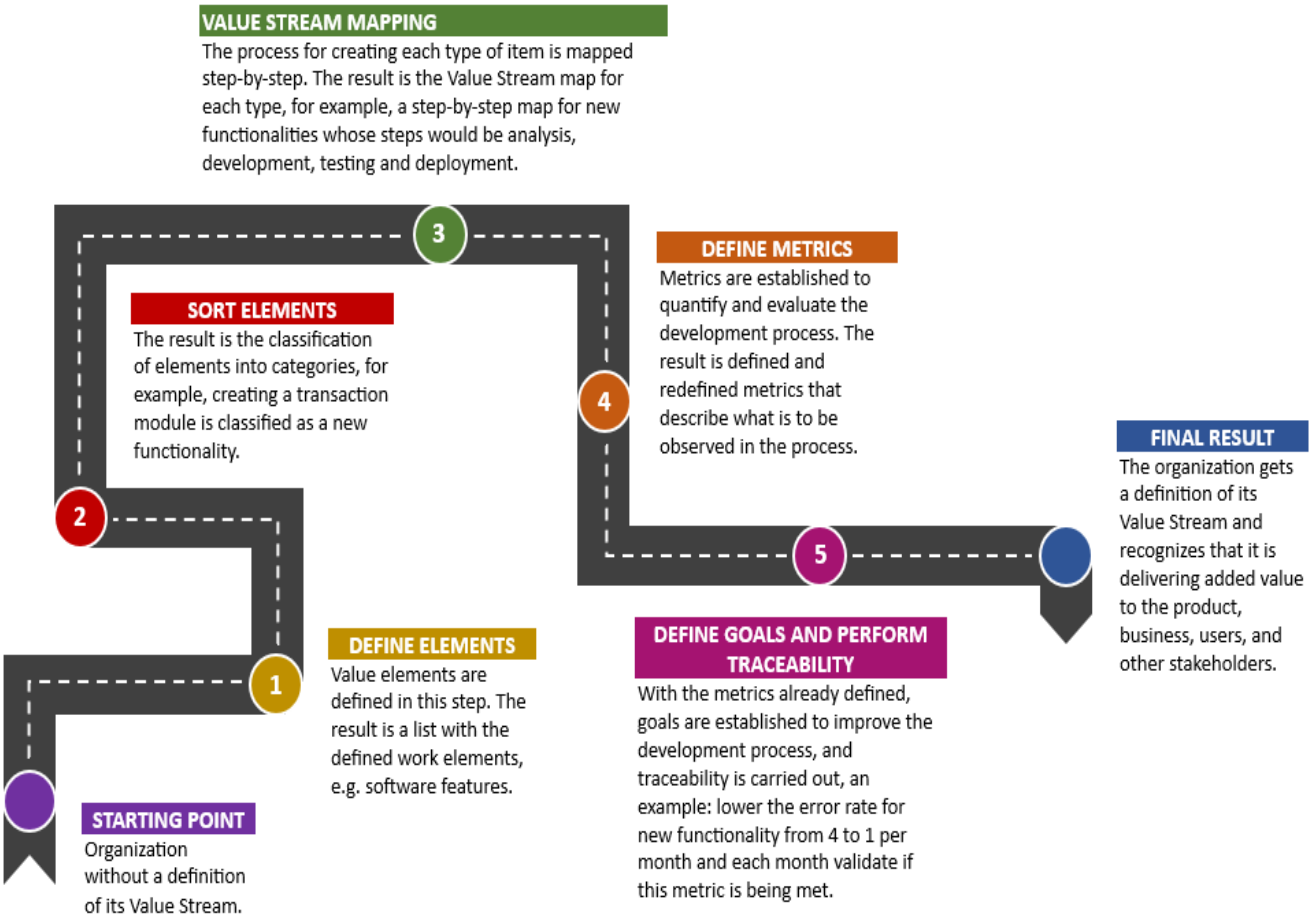

Figure 6. Proposal for roadmap of value stream traceability

Once the proposal is comprehended, we explain each of the steps outlined in Figure 7:

4.1. Define the value elements

As highlighted in [34], it is crucial to have a clear understanding of the value elements being delivered in the current development process of the organization. For instance, within an organization, new features may be developed for an existing application, defects in a legacy application may be rectified, and improvements to an application may be made to mitigate risks. Analyzing the work is essential to identify the segments that contribute value.

4.2. Classify the value types

The value elements identified in step 1 should undergo analysis and classification based on the value they provide [26], [34]. To facilitate this process, *Table 17* can serve as a foundational framework.

Table 17. Classification of values for work elements

Value	**Positive**	**Negative**
Visible	New software functionalities.	Correcting software defects.
Invisible	Improving software architecture.	Managing software system risks

Therefore, an item such as *"Create new registration functionality"* would fall under the classification of *New Functionality*. Conversely, *"Update the version of the security token generation library"* would be categorized under *Manage System Risks*.

4.3. Mapping the value stream of the various value types

After defining and categorizing the value elements, the Value Stream responsible for delivering this value must be identified. In the absence of a formal process for tracing the Value Stream, it is advisable, as suggested in [21], to convene a meeting with stakeholders (Development, Operations, Product, etc.). The objective is to collaboratively create a description of the Value Stream based on actual practices rather than solely relying on internal documentation, which may not accurately reflect the reality of operations.

In [23], a method is outlined as follows: the process commences by segmenting the work into activities. Subsequently, a horizontal timeline is drafted from the customer's perspective, commencing from the moment a need is recognized (or when a request is submitted) and culminating when the product enters production. On this timeline, the significant steps involved in the process are plotted from start to finish, maintaining a level of abstraction without delving into excessive detail. Subsequently, the activities that add value are depicted above the horizontal timeline, while those that do not are placed below. To facilitate this process, a meeting with various stakeholders can be organized, as delineated in the step-by-step procedure illustrated in Figure 7:

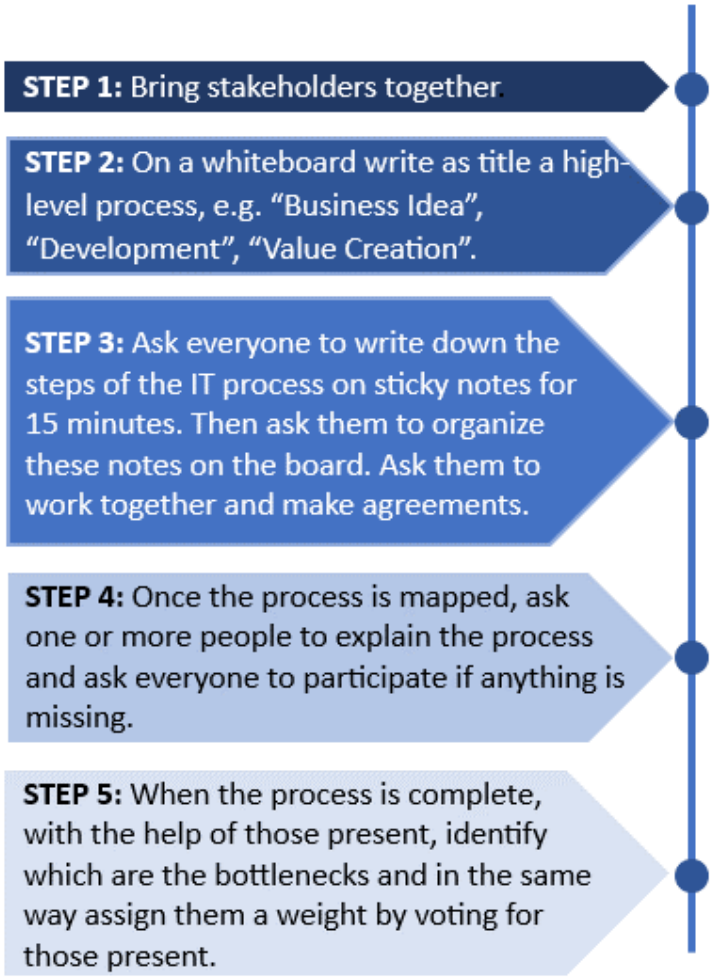

Figure 7. Step-by-step guide to creating a value stream mapping [21]

Upon conclusion of this process, a baseline of the Value Stream is established. Figure 8 illustrates an exemplary Value Stream Mapping (VSM), showcasing the sequence of activities, delineating whether they contribute value or not, and indicating the time taken for each activity. In the instance depicted in Figure 8***Error! Reference source not found.***, activities 1, 3, 4, 6, and 8 are deemed value-adding to the software development process, while activities 2, 5, and 7 are not. Consequently, action plans should prioritize waste elimination by reducing, eliminating, or automating the efforts invested in these non-value-adding activities of the Value Stream. According to [21], organizations should reevaluate the Value Stream every three to six months to identify and address bottlenecks effectively. This approach allows organizations to measure the effectiveness of their actions aimed at improving the Value Stream. Once the Value Stream is defined, organizations can establish metrics to begin evaluating progress in value delivery.

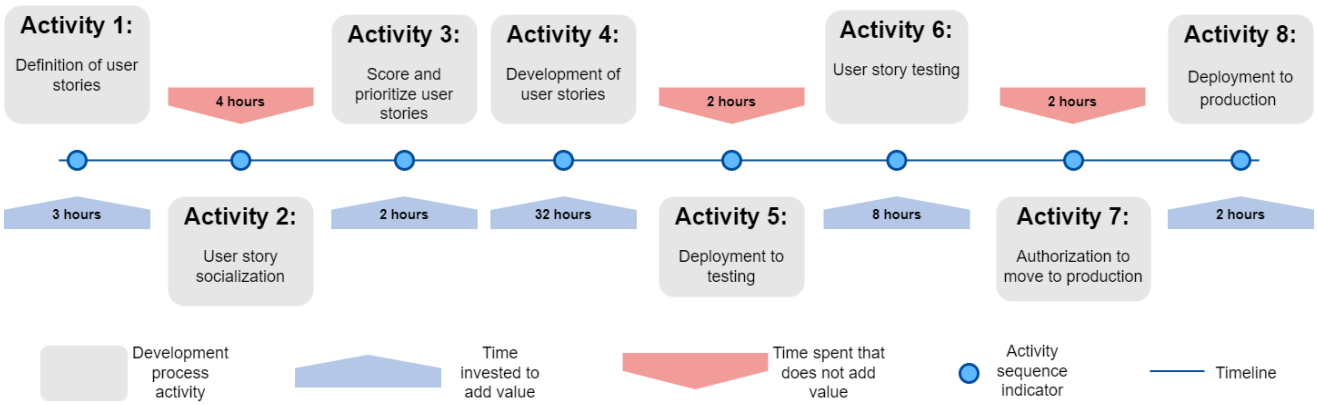

Figure 8. Value stream mapping example [23]

4.4. Define metrics

To enhance this approach, organizations should define technical and cultural metrics, which can be assessed through the practices outlined in [16], including conducting surveys and incorporating records into the developed software. These metrics encompass various aspects and are detailed in *Table 18.*

Table 18. Metric types for value stream in DevOps

Metrics Based on System Data	Metrics Based on Surveys
They are obtained directly from the system.	They are obtained from team members and stakeholders through surveys.
Their strengths are technical accuracy and continuous visibility.	Their strengths lie in providing a comprehensive view of the organization's functioning and capturing data outside the system.

Examples of metrics based on system data include:

a) Response time of the system to a request.
b) Frequency of application crashes.
c) Percentage of memory usage and disk storage capacity utilized by the servers.

Examples of metrics based on surveys include:

a) Administer anonymous surveys utilizing tools like online forms to assess the work environment and individuals' motivation in their current job.
b) Conduct voting sessions during retrospective meetings to gauge the team's sentiments in each development iteration.

4.5. Define goals and perform traceability

Ultimately, to execute the traceability of the Value Stream, it is imperative to establish goals and routinely evaluate the metrics' status in connection with these objectives. This process helps pinpoint bottlenecks and critical points in processes and organizational culture, all of which contribute to the continuous improvement of the organization.

Examples of goals include:

a) Achieve a 10% reduction in overall Value Stream wait time.
b) Reduce server response times from 10 seconds to a maximum of 2 seconds.
c) Lower the error rate per change from 3 errors to a maximum of 1 error.

An organization can periodically assess its metrics, such as monthly, every three months, or every six months, to validate whether goals are being met.

This proposal aims to provide organizations with a foundation to trace the Value Stream and implement improvement actions for their value delivery process through software development.

5. Conclusions

In this study, we addressed several challenges related to achieving Value Stream traceability. We presented practices aimed at overcoming these challenges and proposed methodologies for implementing this process. Additionally, we identified gaps in the existing literature that present opportunities for further research in this previously unexplored topic.

There is a pressing need to broaden research efforts on practices aimed at achieving Value Stream traceability in Software Engineering. For example, beyond surveys, there exists a dearth of data to measure and assess individuals and development teams. Furthermore, it is crucial to underscore the significant potential of existing practices in aiding organizations to enhance their DevOps implementation. An illustrative example is the visibility of waste to senior executives, which is indispensable for driving actions aimed at improving value delivery within organizations.

Similarly, the significance of DevOps practices to organizations extends beyond process optimization. DevOps serves as a catalyst for learning, enabling the identification of gaps in technical understanding among collaborators and fostering improved collaboration and communication. This, in turn, propels organizations forward through the professional development of their staff and a commitment to continuous improvement.

Finally, this study presents a proposal for tracing the Value Stream by integrating various identified practices. This proposal enables the tracing of the Value Stream from multiple perspectives, including business, system, and the work environment.

6. Future projects

In this work, we present a proposal for tracing the Value Stream, which stakeholders can use as the basis for conducting case studies or analyzing it with experts through a focus group to validate its effectiveness. Additionally, stakeholders could update the systematic mapping and expand it by incorporating questions that were not addressed initially, such as “What applications can aid in tracing the Value Stream?” or “What frameworks encompass practices or guidelines for Value Stream tracing?” Another potential avenue for future research could involve focusing on developing the upstream part of the Value Stream, which entails the discovery, analysis, and approval cycle of software development proposals within the organization.

7. Threats to validity

As a potential threat to validity, it is important to acknowledge that this systematic literature review exclusively considers research papers published between 2018 and 2024 (inclusive) and written in English, sourced from multiple consulted databases. However, additional papers from alternative sources were also included. Consequently, there is a possibility that earlier research papers, which could have provided valuable insights to the study, might have been omitted.

Another threat to validity stems from subjective bias in the analysis and the authors’ approaches. Additionally, bias may be present in the selection process of studies. However, we mitigate this by employing a research protocol. The protocol encompasses research questions, inclusion and exclusion criteria, search strings, quality assessment criteria, synthesis method, search strategy, data extraction, and a literature review relevant to the focus of the present work.

Acknowledgements

We thank the anonymous reviewers. Likewise, PhD. Elizabeth Suescún Monsalve and PhD. César Jesús Pardo Calvache acknowledge the contribution of EAFIT University and the University of Cauca, where they serve as full professors, respectively.

References

[1] M. Piattini Velthuis, “Evolución de la Ingeniería del Software y la formación de profesionales”, *Bit & Byte*, vol. 2, No. 4, pp. 15–17, 2016, Consulted: November 10th, 2022. [Online]. Available at: http://sedici.unlp.edu.ar/handle/10915/57358

[2] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland: IT Revolution Press, 2016.

[3] K. Beck *et al.*, “Manifesto for Agile Software Development”.

[4] C. Pardo, C. Orozco, and J. Guerrero, “DevOps Ontology - An ontology to support the understanding of DevOps in the academy and the software industry”, *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 11, No. 2, pp. 207–220, Apr. 2023, doi: 10.21533/pen.v11i2.3474.

[5] A. Dyck, R. Penners, and H. Lichter, “Towards Definitions for Release Engineering and DevOps”, in *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, IEEE, May 2015, p. 3. doi: 10.1109/releng.2015.10.

[6] M. S. Khan, A. W. Khan, F. Khan, M. A. Khan, and T. K. Whangbo, “Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review”, *IEEE Access*, vol. 10, pp. 14339–14349, Jan. 2022.
doi: 10.1109/access.2022.3145970.

[7] D. Smith, D. Villalba, M. Irvine, D. Stanke, and N. Harvey, “Accelerate: State of DevOps 2021”, 2021. Consulted: November 19th, 2022. [Online]. Available at: https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

[8] P. Haindl, “Measuring and Evaluating the Value of Software Features in DevOps”, Doctoral Thesis, Johannes Kepler University Linz, Linz, 2021. Consulted: September 13th, 2022. [Online]. Available at: https://epub.jku.at/download/pdf/6620538.pdf

[9] SAFe STUDIO, “Development Value Streams - Scaled Agile Framework”. Consulted: November 9th, 2022. [Online]. Available at: https://www.scaledagileframework.com/value-streams/

[10] A. K. Kaiser, *Become ITIL® 4 Foundation Certified in 7 Days*, Second Edition. Berkeley, CA: Apress, 2020.
doi: 10.1007/978-1-4842-6361-7.

[11] Richard. Knaster and Dean. Leffingwell, *SAFe 4.5 Distilled: Applying the Scaled Agile Framework for Lean Enterprises*. Boston: Addison-Wesley, 2018.

[12] O. C. Z. Gotel and C. W. Finkelstein, “An analysis of the requirements traceability problem”, in *Proceedings of IEEE International Conference on Requirements Engineering*, Colorado Springs: IEEE Comput. Soc. Press, pp. 94–101.
doi: 10.1109/icre.1994.292398.

[13] P. Rempel and P. Mader, “Preventing Defects: The Impact of Requirements Traceability Completeness on Software Quality”, *IEEE Transactions on Software Engineering*, vol. 43, No. 8, pp. 777–797, Aug. 2017. doi: 10.1109/tse.2016.2622264.

[14] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic Mapping Studies in Software Engineering”, in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, Jun. 2008. doi: 10.14236/ewic/ease2008.8.

[15] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering (2.3)”, Durham, Jul. 2007. Consulted: December 14th, 2022. [Online]. Available at: https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf

[16] N. Forsgren and M. Kersten, “DevOps metrics”, *Commun ACM*, vol. 61, No. 4, pp. 44–48, Mar. 2018. doi: 10.1145/3159169.

[17] M. Kersten, “A Cambrian Explosion of DevOps Tools”, *IEEE Softw*, vol. 35, No. 2, pp. 14–17, Mar. 2018. doi: 10.1109/MS.2018.1661330.

[18] K. Tsilionis and Y. Wautelet, “A model-driven framework to support strategic agility: Value-added perspective”, *Inf Softw Technol*, vol. 141, No. 106734, Jan. 2022. doi: 10.1016/j.infsof.2021.106734.

[19] H. Alahyari, T. Gorschek, and R. Berntsson Svensson, “An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations”, *Inf Softw Technol*, vol. 105, pp. 78–94, Jan. 2019. doi: 10.1016/j.infsof.2018.08.006.

[20] P. Rodríguez, M. Mäntylä, M. Oivo, L. E. Lwakatare, P. Seppänen, and P. Kuvaja, "Advances in Using Agile and Lean Processes for Software Development", in *Advances in Computers*, vol. 113, Oulu, Finland, 2019, pp. 135–224. doi: 10.1016/bs.adcom.2018.03.014.

[21] M. Hering, *DevOps for the modern enterprise: Winning practices to transform legacy IT organizations*, First Edition. Portland, OR: IT Revolution, 2018.

[22] N. Stankovic, "Enhancing enterprise agility with Bizdevops method for Business-It Alignment", Master Thesis, University of Twente, Enschede, 2020. Consulted: December 14th, 2022. [Online]. Available at: http://essay.utwente.nl/85426/

[23] H. Atwal, *Practical DataOps*. Berkeley, CA: Apress, 2020. doi: 10.1007/978-1-4842-5104-1.

[24] E. Reke and D. Powell, "Rethinking value – a means to end the whispering game", *Procedia CIRP*, vol. 104, pp. 1041–1045, 2021. doi: 10.1016/j.procir.2021.11.175.

[25] P. Flewelling, *The Agile Developer's Handbook*, First Edition. Birmingham, UK: Packt Publishing, 2018.

[26] G. C. Murphy and M. Kersten, "Towards Bridging the Value Gap in DevOps", in *DEVOPS 2019: Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, JM. Bruel, M. Mazzara, and B. Meyer, Eds., Springer, Cham, 2020, pp. 181–190. doi: 10.1007/978-3-030-39306-9_13.

[27] G. C. Murphy and M. Kersten, "Task Modularity and the Emergence of Software Value Streams (Impact Award Paper Keynote)", in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, en ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 3. doi: 10.1145/3540250.3569445.

[28] C. Marnewick and A. L. Marnewick, "Benefits realisation in an agile environment", *International Journal of Project Management*, vol. 40, No. 4, pp. 454–465, May 2022. doi: 10.1016/j.ijproman.2022.04.005.

[29] S. Tankhiwale and S. Saraf, "Value Stream Mapping (VSM) Led Approach for Waste and Time to Market Reduction in Software Product Development Process", *Telecom Business Review*, vol. 13, No. 1, pp. 27–35, 2020, Consulted: December 14th, 2022. [Online]. Available at: https://www.proquest.com/openview/8871f7cb371f1809d919367c2dfd4cfd/1?pq-origsite=gscholar&cbl=2043509

[30] S. Kundgol, P. Petkar, and V. N. Gaitonde, "Implementation of value stream mapping (VSM) upgrading process and productivity in aerospace manufacturing industry", *Mater Today Proc*, vol. 46, No. 10, pp. 4640–4646, 2021. doi: 10.1016/j.matpr.2020.10.282.

[31] D. Mudgal, E. Pagone, and K. Salonitis, "Approach to Value Stream Mapping for Make-To-Order Manufacturing", *Procedia CIRP*, vol. 93, pp. 826–831, 2020. doi: 10.1016/j.procir.2020.04.084.

[32] R. Yadav, M. L. Mittal, and R. Jain, "Adoption of lean principles in software development projects", *International Journal of Lean Six Sigma*, vol. 11, No. 2, pp. 285–308, Nov. 2018. doi: 10.1108/IJLSS-03-2018-0031.

[33] D. Musat and P. Rodríguez, "Value Stream Mapping Integration in Software Product Lines", in *Proceedings of the 11th International Conference on Product Focused Software*, in *PROFES '10*. New York, NY, USA: Association for Computing Machinery, 2010, pp. 110–111. doi: 10.1145/1961258.1961285.

[34] M. Kersten, "What Flows through a Software Value Stream?", *IEEE Softw*, vol. 35, No. 4, pp. 8–11, Jul. 2018. doi: 10.1109/ms.2018.2801538.

[35] Mary. Poppendieck and Tom. Poppendieck, Lean Software Development: An Agile Toolkit. in *Agile Software Development Series*. Boston: Addison-Wesley Professional, 2003.

[36] J. P. Womack and D. T. Jones, “Lean thinking—banish waste and create wealth in your corporation”, *Journal of the Operational Research Society*, vol. 48, No. 11, p. 1148, 1997. doi: 10.1038/sj.jors.2600967.

[37] J. P. Womack, D. T. Jones, and D. Roos, *The machine that changed the world*. New York: Rawson Associates, 1990.

[38] Henrik. Kniberg and Mattias. Skarin, Kanban and Scrum - Making the Most of Both, in *Enterprise software development series*. Morrisville, NC: Lulu Press, 2010.

[39] M. Kersten, “Value Stream Architecture”, *IEEE Softw*, vol. 34, No. 5, pp. 10–12, 2017. doi: 10.1109/ms.2017.3571573.

[40] M. Fowler, “Technical Debt Quadrant”. Consulted: August 21st, 2023. [Online]. Available at: https://martinfowler.com/bliki/TechnicalDebtQuadrant.html