DOI: 10.15514/ISPRAS-2024-36(6)-3

Domain-Driven Design in Microservices Architecture

J. Sangabriel-Alarcón, ORCID: 0009-0002-2682-502X <josusangabriel@uv.mx>
J. O. Ocharán-Hernández, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>
X. Limón, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
M. K. Cortés-Verdín, ORCID: 0000-0002-6453-180X <kcortes@uv.mx>

School of Statistics and Informatics, Universidad Veracruzana,
Xalapa, Veracruz, México.

Abstract. With the increment in software development complexity, approaches such as Domain-Driven Design (DDD) are needed to tackle contemporary business domains. DDD is already being used in various software projects with different architectural styles. Although some studies have explored the decomposition of business domains or legacy monolithic systems into microservices, there is a lack of concrete information regarding the practical implementation of DDD in this architectural style. The paper systematizes findings on the purpose of using DDD, its patterns, associated technologies, and techniques to increase the clarity about the use of DDD in microservices-based systems development. A systematic literature review of 35 articles was conducted. Thematic analysis was employed to identify five high-order themes and 11 themes. Based on our analysis, we have concluded that microservice identification emerges as the primary motivation behind developers' adoption of DDD, but not the only usage of DDD reported in the literature. Finally, our analysis found benefits and challenges in the use of DDD in Microservices Architecture which are translated to opportunity areas for future works.

Keywords: domain-driven design; microservices architecture; systematic literature review; thematic analysis.

For citation: Sangabriel-Alarcón J., Ocharán-Hernández J. O., Limón X., Cortés-Verdín M. K. Domain-Driven Design in Microservices Architecture. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 6, 2024. pp. 39-58. DOI: 10.15514/ISPRAS-2024-36(6)-3.

Acknowledgements. This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

Предметно-ориентированное проектирование в микросервисной архитектуре

Х. Сангабриэль-Аларкон, ORCID: 0009-0002-2682-502X <josusangabriel@uv.mx>
Х. О. Очаран-Эрнандес, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>
К. Лимон, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
М. К. Кортес-Вердин, ORCID: 0000-0002-6453-180X <kcortes@uv.mx>

Школа статистики и информатики, Университет Веракруса,
Халапа, Веракрус, Мексика.

Аннотация. С увеличением сложности разработки программного обеспечения для решения современных бизнес-задач необходимы такие подходы, как предметно-ориентированное проектирование (Domain-Driven Design, DDD). DDD уже используется в различных программных проектах с разными архитектурными стилями. Хотя в некоторых исследованиях изучалось разложение бизнес-доменов или унаследованных монолитных систем на микросервисы, пока отсутствует конкретная информация относительно практической реализации DDD в этом архитектурном стиле. Для повышения ясности в отношении использования DDD в разработке систем на основе микросервисов в нашей статье систематизированы выводы о целях использования DDD, его моделях, связанных технологиях и методах. Нами был проведен систематический обзор литературы из 35 статей. Тематический анализ помог выявить 11 тем и пять тем более высокого порядка. Основываясь на проделанном анализе, мы пришли к выводу, что идентификация микросервисов становится основной мотивацией принятия разработчиками DDD, но при этом вовсе не является единственной причиной использования DDD, о которой сообщается в литературе. Наконец, наш анализ выявил преимущества и проблемы в использовании DDD в архитектуре микросервисов, которые будут учитываться при проведении работ в будущем.

Ключевые слова: предметно-ориентированное проектирование; микросервисная архитектура; систематический литературный обзор; тематический анализ.

Для цитирования: Сангабриэль-Аларкон Х., Очаран-Эрнандес Х. О., Лимон К., Кортес-Вердин М. К. Предметно-ориентированное проектирование в микросервисной архитектуре. Труды ИСП РАН, том 36, вып. 6, 2024 г., стр. 39–58 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(6)–3.

Благодарности. Работа была поддержана текущим институциональным финансированием. Никаких дополнительных грантов для проведения или руководства этим конкретным исследованием получено не было.

1. Introduction

This paper is an extension of work initially presented at the 11th International Conference in Software Engineering Research and Innovation (CONISOFT 2023) [1]. The original study is a systematic mapping study on Domain-Driven Design (DDDS) for Microservices Architecture Systems Development. In this paper, we conducted a comprehensive systematic literature review and employed thematic synthesis to identify and analyze patterns in the uses of DDD in this context. This study synthesizes the findings from a broader range of primary studies and search strategies.

Since the release of Eric Evans' book "The Blue Book" in 2004 [2], a community of practitioners has emerged who explore the use of DDD and patterns in different software development projects. DDD can be understood as an approach that addresses the complexities of a business by emphasizing the team's focus on domain knowledge [2]. Some authors [3-5] have proposed patterns and techniques to analyze business domains and incorporate that knowledge into software projects.

DDD patterns can be classified as strategic and tactical designs, which are the key elements of this approach. Strategic design involves domain analysis and decomposition. On the other hand, tactical design translates the knowledge acquired from strategic design into actual lines of code [4].

When applying MSA in practice, developers have encountered a range of challenges in achieving the desired properties of this architectural style [6-9]. Based on the challenges highlighted by various authors concerning MSA, several studies aim to reduce the complexity in the development of microservices-based systems. While some proposals have been put forth to address these challenges, no definitive solution has emerged. However, the use of DDD has been featured prominently in these proposals. This connection between DDD and MSA can be traced back to the 2014 definition of MSA, and the principles of DDD have frequently been referenced within microservices.

Despite the theoretical discussion [10-11] of the relationship between these approaches, uncertainties have surrounded their practical application. Additionally, the challenges associated with MSA [7, 9], combined with those of DDD [11], have given rise to new challenges in the practical implementation of microservices development with DDD. This lack of clarity regarding DDD for microservices development has created a gap between theory and practice. The issue concerning the knowledge gap between the theory and practice of DDD is directly tied to the practical advantages it offers in designing microservices-based systems. In some studies, [12-13], a set of strategies and techniques to design APIs have been reported to deal important decisions about microservices, where the granularity definition of microservices has been one of the most important themes around mentioned architecture. However, various ideas, patterns, and techniques within DDD have often been cited concerning achieving mentioned benefits related to well-defined granularity for microservices. Moreover, incorporating DDD principles suggests enhanced efficiency in stakeholder collaborative work [13]. The mentioned benefits of DDD are examples of potential solutions for microservices challenges, which are hindered by the lack of knowledge about the use of DDD principles and patterns in a practical context.

This lack of clarity can be mitigated by analyzing practical cases documented in the literature regarding the utilization of DDD in the development of microservices-based systems. While other studies have addressed this issue, none provide specific reasons for authors' decisions to employ DDD in microservices development and the corresponding outcomes. Among these unexplored aspects, it is crucial to determine the DDD techniques employed, identify the patterns utilized in microservices design, and explore other pertinent details that shed light on the limitations and areas of opportunity in using DDD for microservices-based development.

To better understand the practical use of DDD in developing microservices-based systems, this study extends our previously systematic mapping study [1]. We conducted a systematic literature review complemented with a thematic analysis. The objective was to investigate the current state of the art in microservices development with DDD in practical scenarios. We compiled and analyzed a collection of diverse studies that reported the use of DDD in their microservices projects from 2014 to 2023. The findings obtained from this evidence can assist developers in identifying the practical applications and adaptations made by their peers when utilizing DDD. Furthermore, this research can also help uncover DDD's limitations and identify areas of opportunity where this approach can effectively address the main challenges associated with microservices development.

This study is organized as follows: Section 2 provides an overview of the related work in this field. Section 3 outlines the research method chosen for conducting the systematic literature review. The execution of this research method is described in Section 4. The results obtained from the research method are presented in Section 5, followed by a discussion in Section 6. Section 7 addresses the potential threats to the validity of this study. Finally, Section 8 presents the conclusions drawn from the evidence collected in this research.

2. Related work

This section provides an overview of the research work associated with the objective of this study. In a study by Singjai et al. 2021 [14], the authors investigated Architectural Design Decisions (ADD) associated with API design and DDD patterns using a grounded theory research method. The APIs developed using DDD served as the foundation for modeling microservices. Specifically, Singjai et al. identified six ADDs and 27 decision options about utilizing the DDD domain model and strategic patterns for delineating API specifics. It is important to note that this study was limited to gray literature and did not analyze white literature sources. Singjai et al. acknowledged the risk of generalizing their findings, underscoring the potential for additional resources in different data sources relevant to the research topic.

Schmidt et al. also conducted a relevant study in 2020 [15], which entailed a systematic literature review focused on microservices identification proposals. This study examined two distinct development approaches: Model-Driven Development (MDD) and DDD. The authors collected a set of primary studies from 2013 to 2019, of which 27 were considered for review. Among these 27 primary studies, only four included DDD patterns specifically for microservices identification. While the study primarily focused on white literature, it did not specifically address the examination of DDD and MSA practices as its main objective. Given the time frame covered by Schmidt et al. and the recent surge in research highlighting the relationship between DDD and MSA, as mentioned in various studies [16-18], there arises a clear need for a dedicated study that concentrates on the utilization of DDD within the context of developing microservices-based systems.

This study will solely encompass white literature to narrow the research scope and delve into previously unexplored evidence of the integration of DDD and MSA. By concentrating solely on white literature, we aim to complement existing related work and provide an analysis of DDD's application in developing microservices-based systems.

3. Research method and conduction

This section describes the method followed for our systematic literature review. Firstly, we followed the Kitchenham proposal [19] for evidence-based research on software engineering. In addition, other methods were selected to complement some phases and activities of the research. The methods used to complement the systematic literature review were: (I) Automatic search with Zhang et al. proposal [20], (II) Snowballing process proposed by Wohlin [21], (III) Narrative synthesis from Popay et al. proposal [22], and (IV) Thematic synthesis from the proposal of Cruzes & Dyba [23]. To complement the analysis with thematic synthesis, some guidelines of the Thematic analysis method proposed by Clark & Braun [24] were performed in this study.

3.1 Search process

We started following Phase 1 of the systematic literature review method proposed by Kitchenham. Following the mentioned method, we defined and refined a set of research questions (RQ) during the research process. These RQs were documented in a systematic literature review protocol [25] and uploaded in Zenodo [25].

These RQs were guidelines for the research process and the key criteria for discarding or selecting papers during the search process. Through a manual search, relevant studies were identified, and they were the basis for performing an automatic search. We chose the proposal from Zhang et al. [20] to create a search string that facilitates the identification of primary studies on different engines. The automatic search method of Zhang et al. is closely related to systematic literature review studies [19], and the proposal of strict metrics to evaluate the quality of a search string (Recall and Precision) reduces the likelihood of missing relevant studies.

Following the automatic search proposal by Zhang et al., the relevant studies found from manual search formed the Quasi-Gold Standard (QGS) [20]. The relevant studies identified after manual search were published in the following databases: IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink. With 18 studies that conformed to the QGS and IEEE Xplore selected as the evaluation engine, we evaluated different versions of the search string with the Recall and Precision metrics following the recommendations of Zhang et al. [20], where the search string was only approved when its recall was at least 80%. Several iterations of search string evaluation were performed.

3.2 Selection process

Once the search string was built, we established inclusion and exclusion criteria based on the characteristics of primary studies that formed the QGS. For the filter process, the selection criteria were grouped into four stages. Stage 1 was performed through the year and type filters of engines selected. Stage 2 grouped exclusion criteria related to the access of papers and the duplicated studies between engines. This duplication was identified mainly between ACM Digital Library and IEEE Xplore, where four duplicated studies were found during the manual search. Stage 3 involved the reading of the title and abstract of each paper. Lastly, in Stage 4, the papers were downloaded and read to confirm that the content answered at least one RQ.

As a result of the selection process, some papers were included and excluded through different stages. A sum of 624 studies was collected from the execution of search strings in all engines. After Stage 1, 357 studies were filtered. In Stage 2, 155 studies were discarded. After Stage 3, 79 studies were discarded, obtaining 123 relevant studies. As a result of Stage 4, 31 primary studies were identified.

3.3 Snowballing process

After the selection process, 31 primary studies were identified. However, some primary studies could have been omitted during the selection process, so we decided to perform a snowballing process. We chose the process proposed by Wohlim [21]. This method proposes a systematic selection based on the relationship between studies through their references, which allows the division of the entire process into backward and forward. Firstly, we performed a backward snowballing, followed by a foreward snowballing. At the end of the snowballing process, 35 primary studies were identified as sum of the primary studies of automatic search and the four primary studies found in snowballing [26-60].

3.4 Data extraction process

Following the recommendations of Kitchenham for a systematic literature review, we performed a preliminary synthesis based on the proposal of Popay et al. [22] to identify the answers to the RQs. We performed only some steps of narrative synthesis to confirm that each primary study answered at least one RQ. This preliminary synthesis also allowed us to familiarize ourselves with the content of primary studies. This familiarization phase is one of the first steps of thematic synthesis [23]. We also performed a thematic synthesis, where the data was combined, and grouped into Themes to express higher-order ideas such as Cruzes and Dyba expressed in their proposal [23].

3.5 Data synthesis

As part of Phase 2 of Kitchenham's method is the Synthesis of research. This process is a crucial part of the analysis of evidence. Through an interpretative and systematic process, new knowledge is generated based on a set of data. Thematic synthesis allows us to combine, compare, and explore the patterns in the data. These meaning patterns are helpful in generating new conclusions (generalizations) to achieve the aim of this study and complete Kitchenham's method. The thematic synthesis method was based on the thematic analysis proposed by Braun and Clark [24], providing guidelines to explore the evidence and cover the step "Data synthesis" of the Kitchenham method. The first step of thematic synthesis corresponds to familiarization with primary studies. The mentioned preliminary synthesis was also used to cover the first step. The second is identifying text segments from primary studies that answer the RQs. This second step inspired the second data extraction for thematic synthesis mentioned in Section 3.4. The third step was the label of text segments. This step was performed through coding in MaxQDA[1] 2020, where the codes were filled

[1] https://www.maxqda.com/

out in the tabular formats described in Section 3.4. The code labels were refined through consensus among the authors of this study.

The fourth step was the identification of themes as a set of closely related codes. Each code represents a relevant and single-faceted concept, while each theme represents a multi-faceted idea [24]. Therefore, we grouped codes that explain the meaning of the same background idea. After identifying the themes, they were grouped into higher-level taxonomies (Higher-order themes). These higher-order themes related to a set of themes show the high-level overview of the data from the evidence collected.

4. Results

The products obtained from the method conduction are shown in this section. These results encompass answers for RQs and a thematic map that synthesizes all data collected by this study. The use of DDD in microservices-based systems development has increased in recent years, where 2023 represents the year with the most significant number of primary studies published. The distribution graphic with the publication years of primary studies is shown in Fig. 1.

Regarding the engines where primary studies were published, the IEEE Xplore was the engine where the major number of primary studies were found, with 25 primary studies published. ACM Digital Library was the second, with five primary studies published, ScienceDirect with three studies, and SpringerLink with two studies. The graphic with the number of primary studies found per engine is shown in Fig. 2.

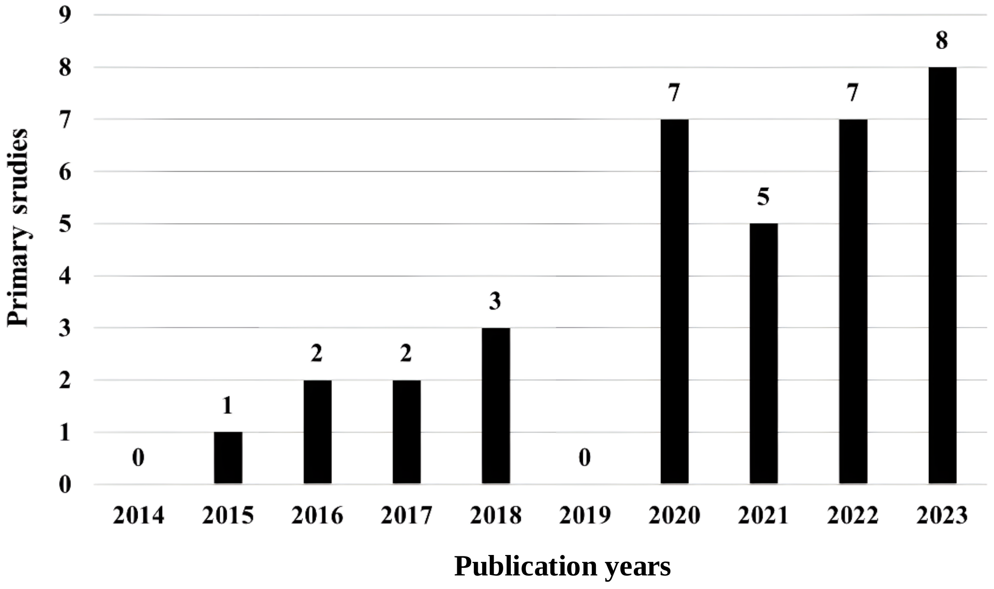

Fig. 1. Primary studies by publication years.

4.1 Answers to research questions

The RQs were the guidelines of this study, and the findings obtained for each RQ enabled us to understand the state of research on the use of DDD for microservices-based systems. This section answers the research questions mainly with quantitative data and some qualitative details. However, the product of qualitative analysis is shown in Section 5.2.

(1) RQ-1: What are the purposes of using DDD for microservices-based systems development? In the use of DDD reported by authors, four motivations were identified in microservices-based systems development. These motivations are shown in Fig. 3.

As shown in Fig. 3, almost all authors of the primary studies mentioned having used DDD for microservices identification [26-52, 54-60], which consists of decomposing a business domain or legacy system into partitions corresponding to microservice candidates.

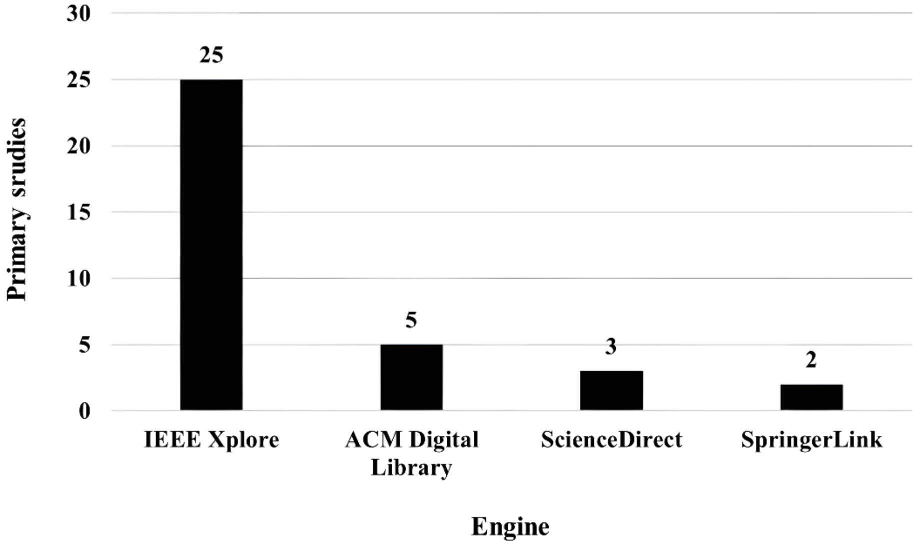

Fig. 2. Primary studies by engine.

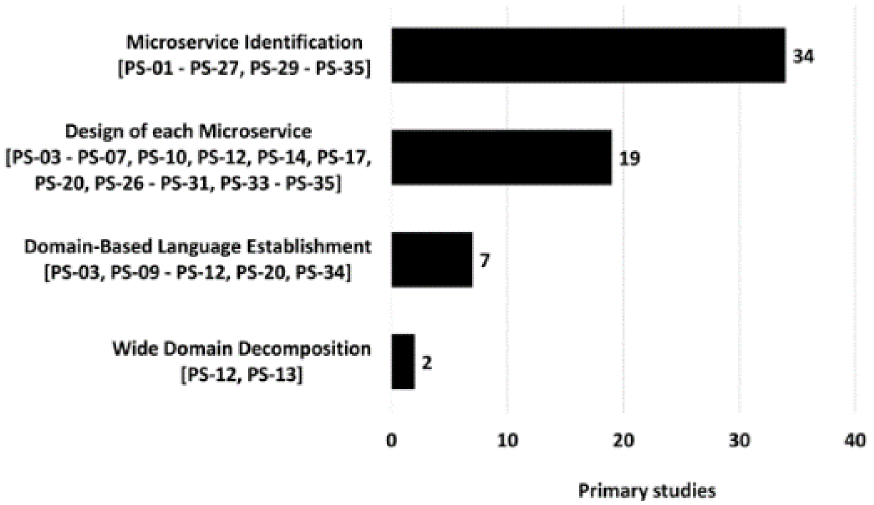

Fig. 3. Purposes of DDD Usage in Microservices-Based Systems Development.

For this purpose, the strategic patterns BC and Subdomain were commonly used. The second most frequent purpose by which authors chose DDD was the design of each microservice [28-32, 35, 37, 39, 42, 45, 51-56, 58-60]. This design took place after the microservices identification, and it refers to the definition of a domain model that reflects business knowledge isolated into each microservice. The third purpose shown in Fig. 3 was using Ubiquitous Language (UL) to cultivate a common language between domain experts and the development team to increase communication effectiveness [28, 34-37, 45, 59]. The fourth purpose was only identified in two primary studies [37, 38]. It uses a Subdomain pattern to split a broad business domain into more manageable partitions. Unlike microservices identification, the use of DDD for wide domain decomposition is about reducing the complexity of the business domain through partitioning, where each part of the domain can be decomposed into several microservices.

(2) RQ-2: What is the evidence about the use of DDD for microservices-based systems development? The first one was software systems, and the second was models. Fig. 4 shows the systems developed by authors with DDD and Microservices Architecture. These systems were classified into two kinds based on the details mentioned by the authors about their development process and the context of the business domain problem.

As shown in Fig. 4, DDD was used to develop 36 microservices-based systems. Of these systems, 56,76% correspond to domains controlled and limited by authors to evaluate a proposal or explore the use of some DDD patterns and principles (Example systems) [27, 29-30, 40, 44, 55-59]. On the other hand, 43,24% of the mentioned systems correspond to real problems where it is necessary to satisfy the necessity of the clients (Industry systems) [26, 28, 34-38, 45-50, 52-53, 58, 60].

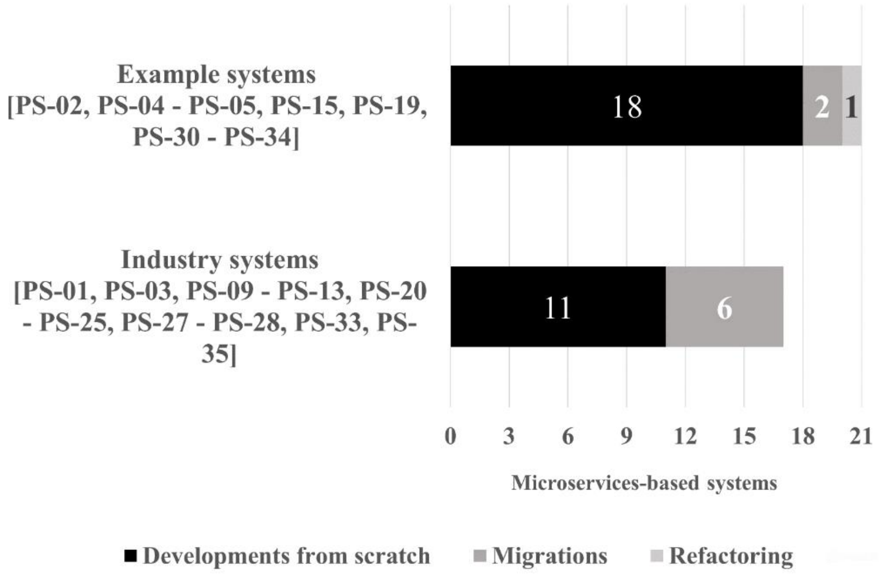

Fig. 4. Microservices-Based Systems developed with DDD.

On the other hand, a set of models involved in the microservices design process were identified. Some of them come from the DDD literature, but others were used to complement the design obtained with DDD, according to the authors of primary studies. In Fig. 5, a set of models is shown, classified by the type of system where the authors mentioned them.

Fig. 5 shows two DDD artifacts used during microservices design: the DDD Domain model and Context map. However, the DDD artifacts were not enough to deal all the specification aspects of microservices-based systems, reason why authors also used UML artifacts to complement the preliminary design obtained with DDD models.

Some models were created by following a notation proposed by authors of primary studies, such as the source model and sketching rough descriptions shown in Fig. 5. These artifacts do not seem to follow a clear standard or notation.

(3) RQ-3: What DDD patterns are used in the microservices-based systems development? A sum of 12 DDD patterns was used by the authors of primary studies in their microservices-based systems design process. These patterns are shown in Fig. 6.

Based on Fig. 6, strategic patterns were mentioned mainly in industry systems, while tactical patterns predominate in example systems. Fig. 6 shows BC as the DDD pattern most frequently mentioned in primary studies [27-30, 34-38, 40, 44-48, 52, 54, 60]. This pattern was treated as a microservice representation, and such as the definition by Evans [2], it delimits the scope of a model. The UL pattern was used by authors of primary studies [28, 34-37, 45-46, 59] to increase the effectiveness of communication between domain experts and the development team, enabling a clear understanding of the problem. In addition, it is one of the patterns (together with Subdomain, ACL, and CS) that was only mentioned by authors who developed an industry system.

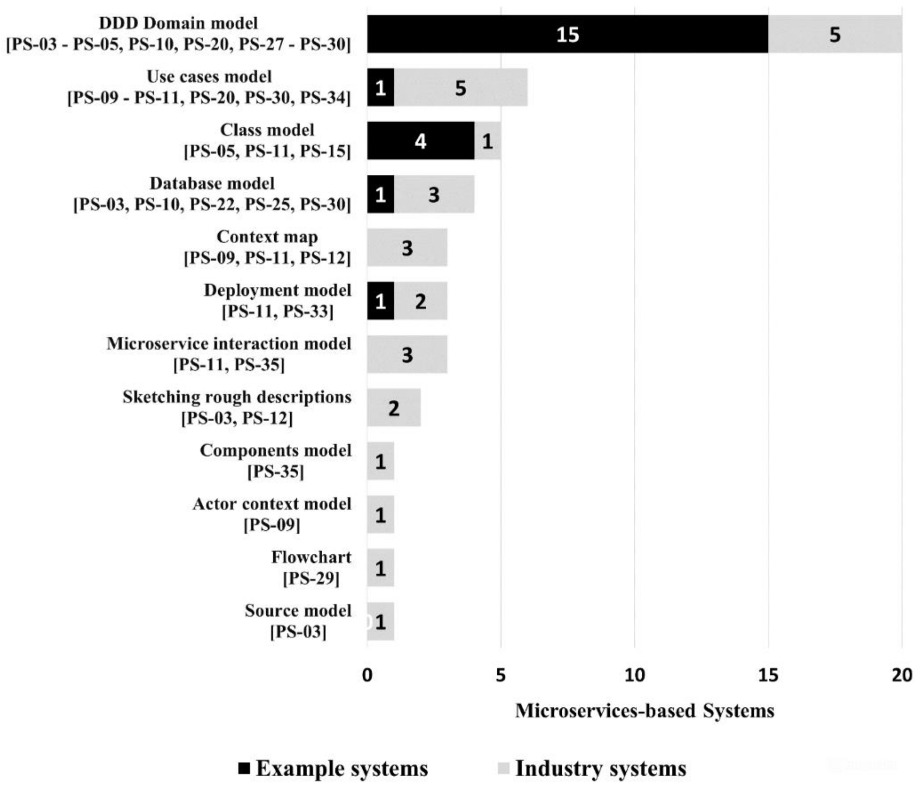

Fig. 5. Models used with DDD for microservices design.

Regarding tactical patterns, they were presented in a DDD domain model to obtain a domain-oriented microservices design. Entity was the tactical pattern most frequently mentioned by authors of primary studies [28-30, 35, 37, 45, 54, 60], and it was not always related to the Aggregate pattern. Unlike Entity, Aggregate is a pattern that requires using Entity and, optionally, other patterns such as Value object, Domain service, Repository, and others. Value object and Domain service were mentioned only as building blocks of the Aggregate pattern. Another pattern used with Aggregate was Repository, which was responsible for manipulating persistent data of an Aggregate through ACID transactions (Atomicity, Consistency, Isolation, and Durability). Event-Sourcing was a pattern mentioned during the DDD design [30], but no details were given about its usage in the microservices design.

(4) RQ-4: What technologies are used with DDD for microservices-based systems development? As reported by the authors, a set of technologies was identified in the microservices-based systems developments with DDD. Most of the technologies were used to implement microservices-based systems, and only three were reported as complements of the design with DDD [30, 34]. They are shown in Table 1.

(5) RQ-05: What techniques are used with DDD in the microservices-based systems development? The techniques of DDD used to complement DDD identified during the microservices-based systems development are shown in Fig. 7.

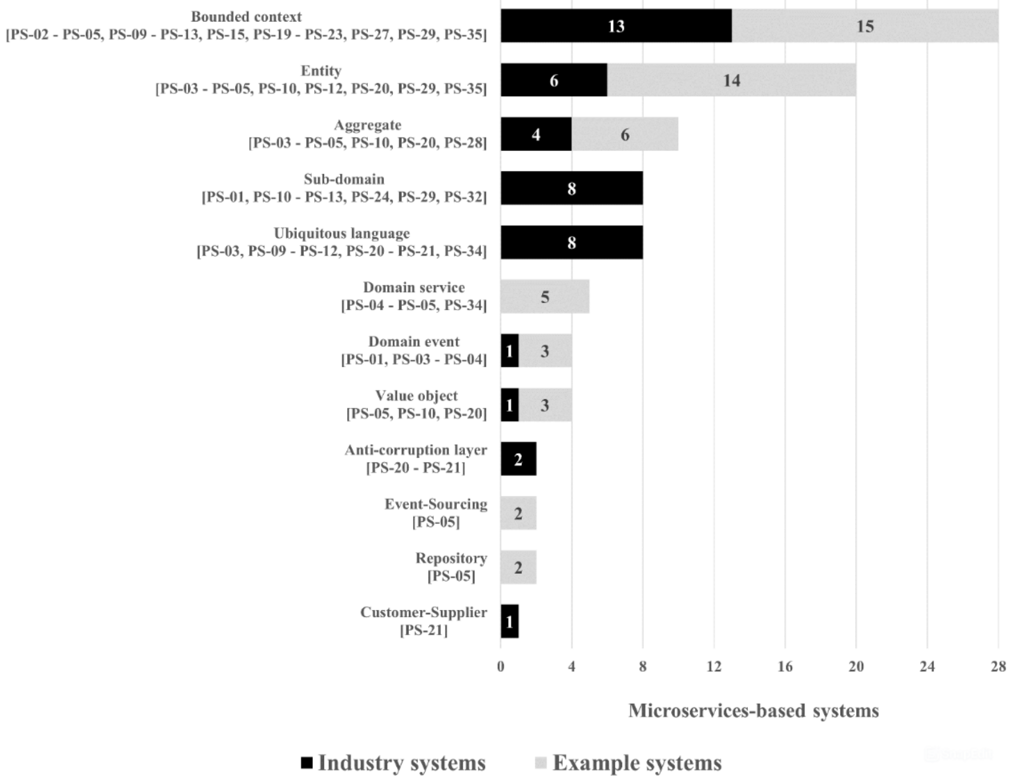

Fig. 6. Domain-Driven Design patterns used in microservices-based systems development.

Table 1. Technologies used with DDD for microservices-based systems development.

Technology name	Description	Primary studies	Official web site
Eclipse Papyrus	Design environment used by authors for code derivation from a DDD domain model made with a UML Profile	PS-05	https://eclipse.dev/papyrus/
ExplorViz	The 3D tool used by authors to identify coupling degrees between BCs (microservices).	PS-09	https://explorviz.dev/
Structure 101	Static code analysis tool used to scan a legacy monolithic project and obtain BC candidates.	PS-09	https://structure101.com/

As shown in Fig. 7, the authors used two kinds of techniques during the microservices design: elicitation techniques and DDD techniques. Context mapping was the DDD technique most frequently used by authors to model microservices candidates as BCs in a context map [34, 36-35, 46, 52]. Event-Storming [4] was a technique related to DDD, as mentioned in PS-01 [26], to identify subdomains, where each subdomain was considered a microservice. Although the authors of PS-01 [26] mentioned the work product obtained after Event-Storming execution (DDD subdomains), no details were mentioned about the procedure followed to perform Event-Storming. Regarding elicitation techniques, these were used together with UL [26, 28, 34, 36-37, 59, 60]. There are some strategies described in DDD literature to cultivate a UL, such as Domain storytelling, Knowledge crunching, and others. However, the authors of primary studies used interviews (mainly), brainstorming, focus groups, and questionnaires to extract the domain knowledge from the interaction with domain experts.

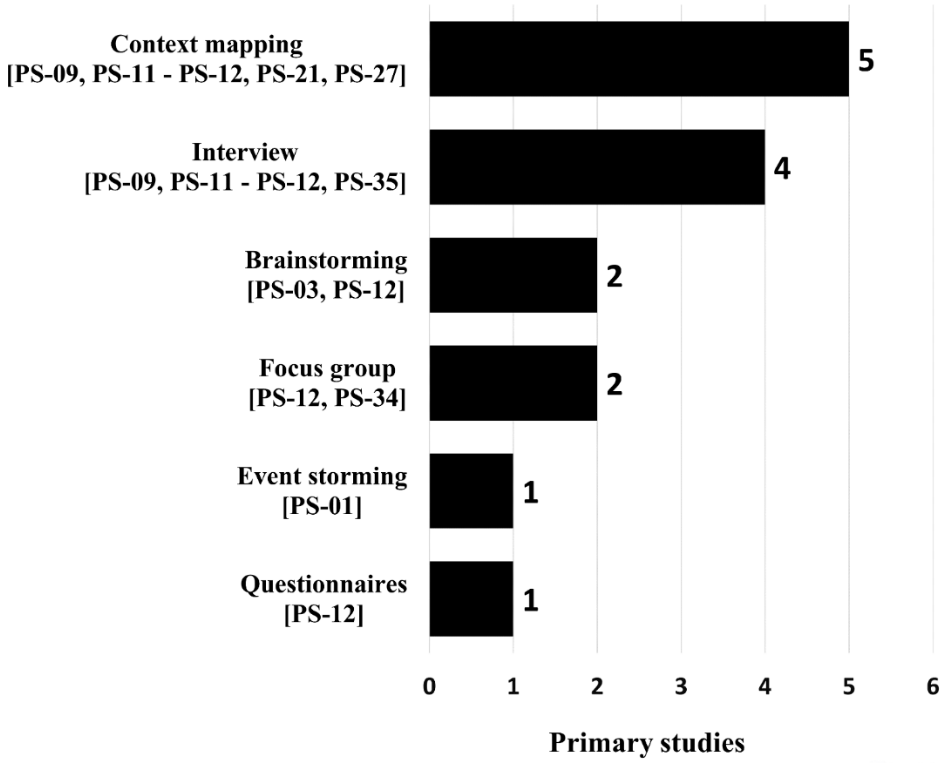

Fig. 7. Techniques used together with DDD.

(6) RQ-06: What challenges are mentioned by authors during the development of microservices-based systems? The authors mentioned 13 challenges when they used DDD in their development processes. These challenges are shown in Fig. 8.

The challenges identified were classified into six categories based on the problems that the authors described in primary studies. As shown in Fig. 10, the authors of primary studies mention two main difficulties. The first one is the procedure that is not defined [27, 30, 45]. It consists of the lack of a strict process to apply DDD techniques and patterns "correctly." The decision of what DDD pattern should be used and how depends on the business domain and the comprehension of a software engineer about the context of the problem. The second main challenge mentioned by authors of primary studies is related to the limitations mentioned by Evans [2]. When technical complexity predominates over the complexity of the business domain, DDD can complicate the solution [30, 42, 56]. The authors mentioned this because they do not recommend using DDD for developments where the most significant complexity is technical.

(7) RQ-7: What proposals exist for the development of microservices-based systems with DDD? Due to the incremental use of DDD in microservices-based systems development, some authors have proposed procedural guidelines to overcome the most frequent challenges of Microservice Architecture with the helplessness of DDD. These proposals are shown in Fig. 9, and they were classified according to the proposal type described by the authors of primary studies where they were extracted.

As shown in Fig. 9, 21 proposals were identified and classified into five categories. These categories come from the denomination authors use to refer to their proposals. For example, the authors named their proposals "Approaches" in six primary studies [35, 40, 43, 55-57]. In five primary studies [30, 33, 37, 39, 58], authors named their proposals as "Methodologies" and so on. Based on Fig. 9, it is also possible to see that 80,95% of proposals were evaluated by authors. In comparison, 19.05%

were not evaluated, postponing their evaluation to future works or delegating the evaluation for interested lectures.

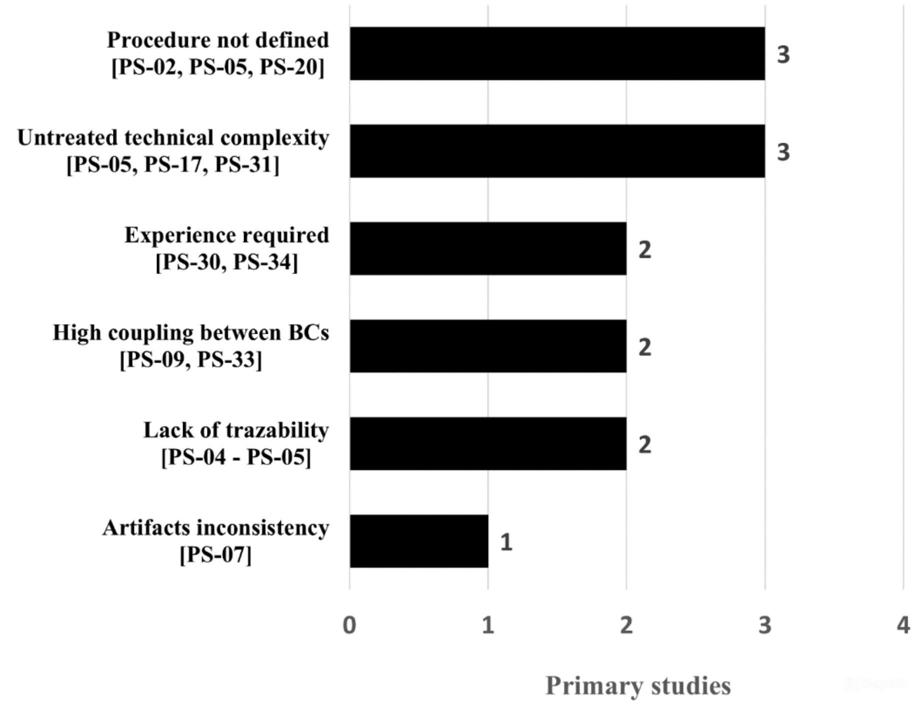

Fig. 8. Challenges faced with DDD in microservices-based systems development.

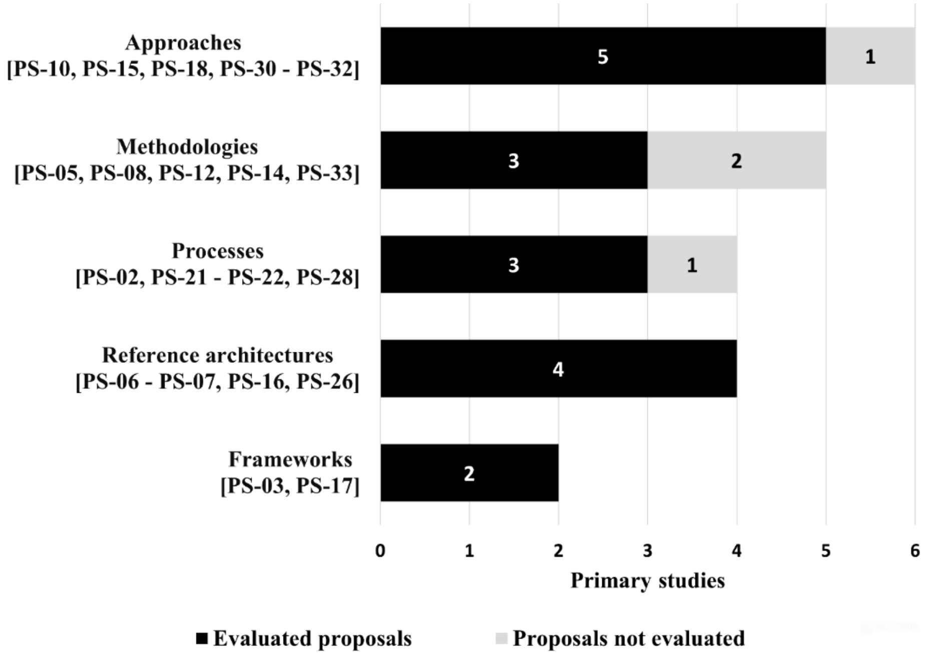

Fig. 9. Proposals for microservices design with DDD.

4.2 Thematic Synthesis Results

As a result of the systematic literature review, a familiarization phase recommended by Braun and Clark [24] was performed. However, a new data extraction was conducted based on RQs and thematic synthesis guidelines [23]. This data collection enabled us to identify meaning patterns among the data. This first level resulted in a set of code concepts seen as building blocks of themes. A sum of 32 codes were identified from the evidence. These codes were transformed into 11 themes that isolate the idea behind a group of codes. In the end, five higher-order themes were identified through theme grouping. Based on the thematic synthesis process, it is possible to describe a particular story of collected data, as mentioned by Braun and Clark [24]. This high-level overview synthesized with thematic synthesis can be seen in Fig. 10 in the high-order themes model.

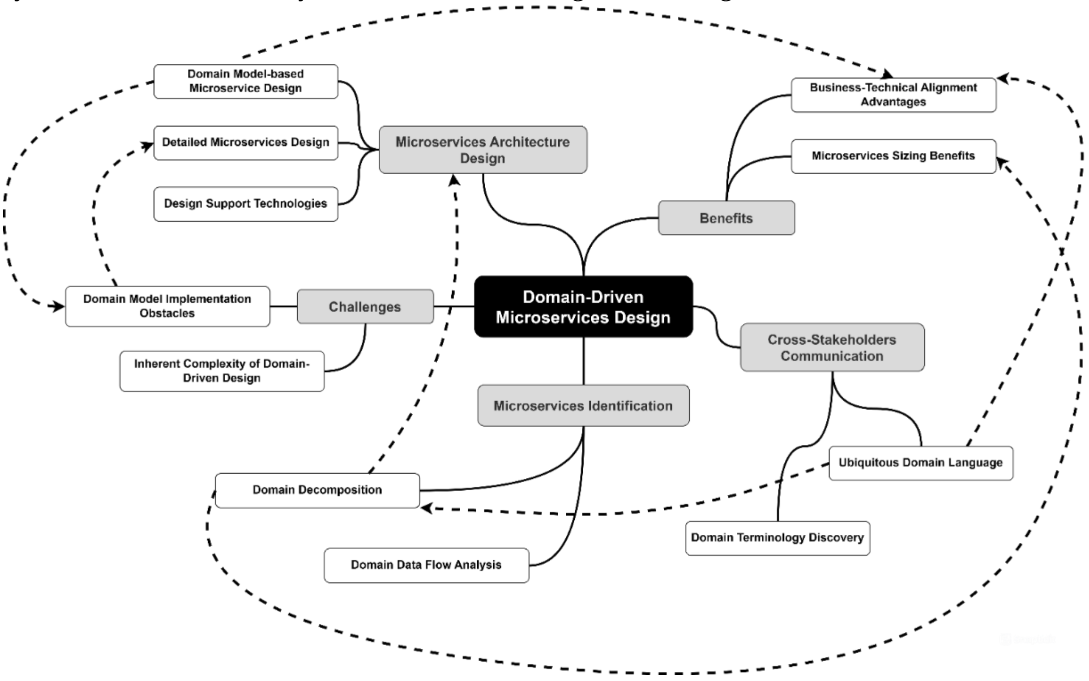
Domain Model-based Microservice Design
Detailed Microservices Design
Design Support Technologies
Microservices Architecture Design
Business-Technical Alignment Advantages
Microservices Sizing Benefits
Benefits
Domain Model Implementation Obstacles
Challenges
Domain-Driven Microservices Design
Inherent Complexity of Domain-Driven Design
Cross-Stakeholders Communication
Microservices Identification
Ubiquitous Domain Language
Domain Decomposition
Domain Terminology Discovery
Domain Data Flow Analysis

Fig. 10. Thematic Map of Domain-Driven Microservices Design.

(1) Cross-Stakeholders Communication. According to Vlad Khononov [4], the central idea of DDD is the communication. The domain knowledge shared between domain experts and developers should be clear and consistent. With this similar purpose, authors of primary studies familiarized themselves with the domain experts' jargon and used it to cultivate a UL free of technical details and ambiguous terms. This approach forms the basis of the theme "Ubiquitous Domain Language", which consists of using UL as a business domain glossary to enrich the domain knowledge exchange between stakeholders. This language is product of a distillation process, which is the idea behind the "Domain Terminology Discovery" theme, and it is related to the use of elicitation techniques mentioned in above sections.

(2) Microservice identification. Cultivating UL enables benefits related to effective communication, but another consequence of its usage is the identification of BCs. Each BC acts as a semantic boundary that delimits the meaning of the terms that conform to an UL. Through BCs, it is possible to decompose a business domain into semantic domain partitions that represent microservices. The theme "Domain Decomposition" encompasses all the activities and strategies (described in the above sections) used by authors of primary studies for business domain partitioning into BCs, subdomains, or Aggregates that represent microservices.

Another strategy was identified using DDD analysis techniques to identify clusters of domain concepts. The authors performed Event-storming to identify microservices candidates. This technique and the use of domain events reflect the "Domain Data Flow Analysis" theme, which involves the analysis of the closely related domain events that allowed authors to identify clusters treated as microservices.

(3) Microservices Architecture Design. As shown in the above sections, all tactical design patterns were translated into using the DDD domain model to design the business domain layer for each microservice identified with strategic design. The domain layer is a crucial part of the architecture of the microservices-based systems developed by authors, which is why the theme "Domain Model-Based Microservices Design" was defined. However, some other technical details were not specified with DDD artifacts.

Mainly in industry systems development, challenges related to the design specification of microservices were mentioned. This lack of technical specification for microservices was why other design patterns such as CQRS, Saga, Strangler Fig, and others were used together with standardized diagrams to describe details related to the implementation of microservices. These design resources used to refine the preliminary design obtained with DDD were defined as the "Detailed Microservices Design" theme. We also notice another design resource to refine the design obtained with DDD for each microservice. This is the use of technologies mentioned in primary studies PS-05 and PS-09. This action to complement the design of microservices was defined as the "Design Support Technologies" theme.

(4) Challenges. In answer to RQ6, challenges mentioned by authors of primary studies were extracted. As a result of thematic analysis, these challenges were classified into two themes that represent the two main difficulties faced by developers during microservices design with DDD. The theme named "Inherent Complexity of Domain-Driven Design” is related to the lack of guidelines, checkpoints, and a strict path to know if a developer is applying DDD correctly. Another challenge was defined as "Domain Model Implementation Obstacles", which comes from the problems faced by authors who tried to implement the DDD domain models. Some authors have made some proposals; however, there are no rules, guidelines, or strict specific ways to generate code from these DDD artifacts.

(5) Benefits. Just as the authors of primary studies have reported challenges in the use of DDD for microservices-based systems development, some authors mentioned the benefits obtained from the execution of some of DDD techniques and the use of its patterns. Some authors of primary studies mentioned benefits related to development complexity. In PS-04, PS-05, and PS-30, authors described the business domain complexity isolated into some DDD patterns such as BCs, Aggregates, or Subdomains. This isolation enabled them to tackle the most significant complexity of their microservices-based projects, the business domain logic. These benefits were grouped into the theme "Business-Technical Alignment Advantages".

Furthermore, other primary study authors mentioned benefits during the microservices size definition. Based on the decomposition process followed by authors, each microservice could sometimes be represented as a BC or an Aggregate. This decomposition proposed by DDD contributes to modifiability. These benefits related to the size of microservices were grouped in the theme of “Microservices Sizing Benefits”.

5. Discussion

In this study, we successfully answered all the research questions by employing the research method conduction. Our efforts involved collecting and synthesizing a wealth of knowledge on the practical use of Domain-Driven Design (DDD) in developing microservices-based systems.

Analyzing the demographic results of the study yielded interesting findings, particularly an increased interest in the adoption of DDD in Microservices Architecture (MSA). It is worth noting that a gap exists between theoretical understanding and practical implementation of certain patterns, such as Subdomain or BC. Consequently, through this research, we have provided evidence-based knowledge on these patterns and their application. Our findings complement the grounded theory study published by Singjai et al. [14] and the systematic review conducted by Schmidt et al. [15], offering valuable insights into the practical use of DDD in microservices system design.

The utilization of Domain-Driven Design (DDD) has emerged as a vital component in the domain analysis phase of microservices-based systems development within the industry. Strategic design, in particular, plays a crucial role in establishing a shared understanding among stakeholders, enabling authors to express ideas unambiguously. Conversely, developers have primarily utilized tactical design to tackle controlled domain problems and serve specific purposes. Additionally, existing literature indicates that DDD has been employed to decompose business domains into microservices candidates in the analysis process. However, it is important to note that the BC pattern is not the only one utilized or emphasized in the literature. Using UL for stakeholder interaction is a common practice in complex domains where developers may not be familiar with the domain. On the other hand, applying Tactical design in industrial projects has been less frequent. Thus, certain DDD patterns, such as Domain Event, Event-Sourcing, and Domain Services, remain underutilized in real-world contexts.

6. Threats to validity and limitations

In the literature reviews, Kitchenham and other authors [19, 22-23] emphasized the importance of reliability. This aspect was carefully considered throughout the research process, from manual search to data synthesis using Cruzes and Dyba's proposal. We implemented a series of mitigation measures to minimize potential biases at various stages of the research.

To ensure the selection of relevant papers was unbiased, we utilized a manual search approach and established inclusion and exclusion criteria based on the Quasi-Gold Standard. These criteria helped us avoid solely relying on one search engine's studies. Once we identified primary studies, we further augmented our research by employing a snowballing technique. This process helped to minimize the possibility of overlooking any relevant studies.

Once the selection process was complete, the chosen primary studies underwent a rigorous evaluation by the authors of this study to ensure their relevance to at least one RQ. Also, the authors continuously reviewed and revised their work during the data extraction process to maintain accuracy. Review questions were developed and regularly evaluated to avoid omissions and confirm that no crucial data had been missed. The same meticulous approach was applied when defining themes and subthemes, with each code being meticulously linked to specific text segments and the themes closely tied to these codes. In the same way, the names assigned to the codes, themes, and higher-order themes were determined through collaborative revisions among the authors of this study.

7. Conclusion

In this study, we adopted the systematic literature review method proposed by Kitchenham [19] to examine the utilization of DDD in developing a microservices-based system. We formulated seven research questions (RQs) to guide our research process and ensure focused research. Our selection process involved both manual and automatic searches to identify relevant studies. Through this process, we identified 31 primary studies. We also employed snowballing techniques to enhance our selection, which led us to four additional studies. We then conducted a preliminary synthesis to familiarize ourselves with the primary studies and address the RQs, mainly focusing on the application of DDD in the development of microservices-based systems. To gather the necessary data, we performed an extraction process. To provide a comprehensive analysis, we further conducted a thematic synthesis utilizing the method proposed by Cruzes and Dyba. To complement this approach, we also incorporated recommendations from the Braun and Clark proposal, ensuring a robust analysis of the collected data.

Throughout our analysis, we have identified specific details regarding the application of DDD that contribute to enhancing effective knowledge sharing between developers and domain experts. These details primarily revolve around the integration of UL with DDD and the utilization of various elicitation techniques. Interestingly, these aspects have not been extensively addressed in related studies, thereby providing fresh insights into the broader scope of DDD beyond its traditional utilization for system decomposition.

Among the different uses we discovered, the most frequently reported one involves decomposing a business domain or legacy system into microservices. However, our analysis captured new and pertinent details about using strategic patterns to define the business scope of microservices, as well as variations and adaptations.

Most authors in the primary studies highlighted the successful implementation of microservices, explicitly noting the absence of coupling issues between microservices. Some authors went so far as to underscore DDD's potential for achieving an optimal scope of microservices based on business capabilities. While the remaining authors did not mention any problems in their DDD-driven microservices systems, they did not specifically address certain characteristic aspects of DDD within the context of MSA.

Despite the overall positive outcomes reported, some challenges persist in the practical application of DDD. These challenges primarily stem from the perceived complexity of implementing DDD, which can be particularly daunting for developers without prior experience analyzing and designing intricate business domains. Additionally, there is an opportunity for future work in refining the implementation of DDD artifacts, such as the domain model, to further enhance its effectiveness and efficiency in microservices development. Finally, we envision future work focused on delving into the creation of DDD patterns that allow the development of code that effectively represents the underlying business logic, with minimal dependencies on specific programming languages based on Object-Oriented Programming.

Conflict of interest

The authors declare that they have no conflicts of interest.

References

[1]. J. Sangabriel-Alarcón, J. O. Ocharán-Hernández, K. Cortés-Verdín, and X. Limón, “Domain-Driven Design for Microservices Architecture Systems Development: A Systematic Mapping Study,” in 2023 11th International Conference in Software Engineering Research and Innovation (CONISOFT), 2023, pp. 25–34.
[2]. E. Evans, “Domain-driven design: tackling complexity in the heart of software,” p. 529, 2004.
[3]. E. Evans, “Domain-Driven Design Reference: Definitions and Pattern Summaries”, 2014.
[4]. V. Khononov and J. Lerman, “Learning domain-driven design: aligning software architecture and business strategy”, p. 312, 2021.
[5]. V. Vernon, “Implementing Domain-Driven Design”, 2013.
[6]. V. Velepucha and P. Flores, “Monoliths to microservices-Migration Problems and Challenges: A SMS,” Proceedings - 2021 2nd International Conference on Information Systems and Software Technologies, ICI2ST 2021, pp. 135–142, Mar. 2021, doi: 10.1109/ICI2ST51859.2021.00027.
[7]. G. Liu, B. Huang, Z. Liang, M. Qin, H. Zhou, and Z. Li, “Microservices: Architecture, container, and challenges,” Proceedings - Companion of the 2020 IEEE 20th International Conference on Software Quality, Reliability, and Security, QRS-C 2020, pp. 629–635, Dec. 2020, doi: 10.1109/QRS-C51114.2020.00107.
[8]. R. Mubashir, J. Ahmed, F. Khakwani, and T. Rana, Microservices Architecture: Challenges and Proposed Conceptual Design. 2019.
[9]. S. Salii, J. Ajdari, and X. Zenuni, “Migrating to a microservice architecture: benefits and challenges,” 2023.
[10]. S. Newman, “Building microservices: Designing fine-grained systems (second edition),” pp. 1–10, 2021.

[11]. F. Rademacher, J. Sorgalla, and S. Sachweh, “Challenges of domain-driven microservice design: A model-driven perspective,” IEEE Softw, vol. 35, no. 3, pp. 36–43, May 2018, doi: 10.1109/MS.2018.2141028.
[12]. M. Tello-Rodríguez, J. O. Ocharán-Hernández, J. C. Pérez-Arriaga, X. Limón, and Á. J. Sánchez-García, “A Design Guide for Usable Web APIs,” Programming and Computer Software, vol. 46, no. 8, pp. 584–593, 2020, doi: 10.1134/S0361768820080241.
[13]. B. Jin, S. Sahni, and A. Shevat, “Designing Web APIs,” 2018.
[14]. A. Singjai, U. Zdun, and O. Zimmermann, “Practitioner Views on the Interrelation of Microservice APIs and Domain-Driven Design: A Grey Literature Study Based on Grounded Theory,” in Proceedings - IEEE 18th International Conference on Software Architecture, ICSA 2021, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 25–35. doi: 10.1109/ICSA51549.2021.00011.
[15]. R. A. Schmidt and M. Thiry, “Microservices identification strategies: A review focused on Model-Driven Engineering and Domain Driven Design approaches,” in 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), IEEE, 2020. Accessed: Nov. 13, 2022. [Online]. Available: https://ieeexplore-ieee-org.ezproxy.uv.mx/document/9141150/.
[16]. A. Macias, E. Navarro, C. Cuesta, and U. Zdun, “Architecting Digital Twins Using a Domain-Driven Design-Based Approach,” International Conference on Software Architecture (ICSA), no. 62, pp. 183–209, 2023, doi: 10.13039/501100011033.
[17]. C. Praschl, S. Bauernfeind, C. Leitner, and G. A. Zwettler, “Domain-Driven Design as Model Contract in Full-Stack Development,” in International Conference on Electrical, Computer, Communications and Mechatronics Engineering, ICECCME 2023, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICECCME57830.2023.10252654.
[18]. F. Rademacher, S. Sachweh, and A. Zündorf, “Towards a UML profile for domain-driven design of microservice architectures,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10729 LNCS, pp. 230–245, 2018, doi: 10.1007/978-3-319-74781-1_17/COVER.
[19]. B. A. Kitchenham, D. Budgen, and P. Brereton, “Evidence-Based Software Engineering and Systematic Reviews,” 2015.
[20]. H. Zhang, M. A. Babar, and P. Tell, “Identifying relevant studies in software engineering,” Inf Softw Technol, vol. 53, no. 6, pp. 625–637, Jun. 2011, doi: 10.1016/j.infsof.2010.12.010.
[21]. C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE ’14, New York, New York, USA: ACM Press, 2014, pp. 1–10. doi: 10.1145/2601248.2601268.
[22]. J. Popay et al., “Guidance on the conduct of narrative synthesis in systematic Reviews. A Product from the ESRC Methods Programme. Version 1,” undefined, 2006, doi: 10.13140/2.1.1018.4643.
[23]. D. S. Cruzes and T. Dybá, “Recommended steps for thematic synthesis in software engineering,” in International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, 2011, pp. 275–284. doi: 10.1109/esem.2011.36.
[24]. V. Clarke and V. Braun, “Thematic analysis: A practical guide,” London: SAGE, 2021, Accessed: Oct. 30, 2023. [Online]. Available: https://uk.sagepub.com/en-gb/eur/thematic-analysis/book248481#description.
[25]. J. Sangabriel-Alarcón, J. O. Ocharán-Hernández, X. Limón, and K. Cortés-Verdín, “Domain-Driven Design in Microservices-Based Systems Development: A Systematic Literature Review and Thematic Analysis [Dataset].” [Online]. Available: https://zenodo.org/records/13294975.
[26]. PS-01. G.-C. Pan, P. Liu, and J.-J. Wu, “A Cloud-Native Online Judge System,” in 2022 IEEE COMPSAC, 2022, doi: 10.1109/COMPSAC54236.2022.00204.
[27]. PS-02. N. Ivanov and A. Tasheva, “A Hot Decomposition Procedure: Operational Monolith System to Microservices,” in 2021 IEEE ICAI, 2021, doi: 10.1109/ICAI52893.2021.9639494.
[28]. PS-03. M. I. Joselyne, G. Bajpai, and F. Nzanywayingoma, “A Systematic Framework of Application Modernization to Microservice-based Architecture,” in 2021 IEEE ICEET, 2021, doi: 10.1109/ICEET53442.2021.9659783.
[29]. PS-04. A. Singjai and U. Zdun, “Conformance Assessment of Architectural Design Decisions on the Mapping of Domain Model Elements to APIs and API Endpoints,” in 2022 IEEE ICSA-C, 2022, doi: 10.1109/ICSA-C54293.2022.00058.
[30]. PS-05. F. Rademacher, S. Sachweh, and A. Zündorf, “Deriving Microservice Code from Underspecified Domain Models Using DevOps-Enabled Modeling Languages and Model Transformations,” in 2020 IEEE SEAA, 2020, doi: 10.1109/SEAA51224.2020.00047.

[31]. PS-06. A. Steffens, H. Lichter, and J. S. Döring, “Designing a Next-Generation Continuous Software Delivery System: Concepts and Architecture,” in 2018 ACM Conference on Software Engineering and Applications, 2018, doi: 10.1145/3194760.3194768.
[32]. PS-07. Y. Ding et al., “Enterprise Service Application Architecture Based on Domain Driven Model Design,” in 2020 IEEE ITCA, 2020, doi: 10.1109/ITCA52113.2020.00167.
[33]. PS-08. P. Ray and P. Pal, “Extending the SEMAT Kernel for the Practice of Designing and Implementing Microservice-Based Applications using Domain Driven Design,” in 2020 IEEE CSEET, 2020, doi: 10.1109/CSEET49119.2020.9206200.
[34]. PS-09. A. Krause et al., “Microservice Decomposition via Static and Dynamic Analysis of the Monolith,” in 2020 IEEE ICSA-C, 2020, doi: 10.1109/ICSA-C50368.2020.00011.
[35]. PS-10. C.-Y. Li, S.-P. Tseng, and T.-W. Lu, “Microservice Migration Using Strangler Fig Pattern: A Case Study on the Green Button System,” in 2020 IEEE ICS, 2020, doi: 10.1109/ICS51289.2020.00107.
[36]. PS-11. A. Rahmatulloh et al., “Microservices-Based IoT Monitoring Application with a Domain-Driven Design Approach,” in 2021 IEEE ICADEIS, 2021, doi: 10.1109/ICADEIS52521.2021.9701966.
[37]. PS-12. M. I. Josélyne et al., “Partitioning Microservices: A Domain Engineering Approach,” in 2018 ACM Conference on Software Engineering and Applications, 2018, doi: 10.1145/3195528.3195535.
[38]. PS-13. M. Pham and D. B. Hoang, “SDN Applications - The Intent-Based Northbound Interface Realization for Extended Applications,” in 2016 IEEE NetSoft, 2016, doi: 10.1109/NETSOFT.2016.7502469.
[39]. PS-14. E. Cabrera et al., “Towards a Methodology for Creating Internet of Things (IoT) Applications Based on Microservices,” in 2020 IEEE SCC, 2020, doi: 10.1109/SCC49832.2020.00072.
[40]. PS-15. R. Petrasch, “Model-Based Engineering for Microservice Architectures Using Enterprise Integration Patterns for Inter-Service Communication,” in 2017 IEEE JCSSE, 2017, doi: 10.1109/JCSSE.2017.8025912.
[41]. PS-16. J. Dobaj et al., “A Microservice Architecture for the Industrial Internet-Of-Things,” in 2018 ACM Conference on Software Engineering and Applications, 2018, doi: 10.1145/3282308.3282320.
[42]. PS-17. S. Braun, A. Bieniusa, and F. Elberzhager, “Advanced Domain-Driven Design for Consistency in Distributed Data-Intensive Systems,” in 2021 ACM Conference on Software Engineering and Applications, 2021, doi: 10.1145/3447865.3457969.
[43]. PS-18. M. Khemaja, “Domain Driven Design and Provision of Micro-Services to Build Emerging Learning Systems,” in 2016 ACM Conference on Software Engineering and Applications, 2016, doi: 10.1145/3012430.3012643.
[44]. PS-19. Z. Li, “Using Public and Free Platform-as-a-Service (PaaS) Based Lightweight Projects for Software Architecture Education,” in 2020 ACM Conference on Software Engineering and Applications, 2020, doi: 10.1145/3377814.3381704.
[45]. PS-20. P. Oukes et al., “Domain-Driven Design Applied to Land Administration System Development: Lessons from the Netherlands,” in Land Use Policy, vol. 105, 2021, doi: 10.1016/j.landusepol.2021.105379.
[46]. PS-21. C. E. da Silva, Y. de Lima Justino, and E. Adachi, “SPReaD: Service-Oriented Process for Reengineering and DevOps,” in Software: Practice and Experience, 2022, doi: 10.1007/s11761-021-00329-x.
[47]. PS-22. C.-Y. Fan and S.-P. Ma, “Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report,” in 2017 IEEE AIMS, 2017, doi: 10.1109/AIMS.2017.23.
[48]. PS-23. A. Krylovskiy, M. Jahn, and E. Patti, “Designing a Smart City Internet of Things Platform with Microservice Architecture,” in 2015 IEEE FiCloud, 2015, doi: 10.1109/FiCloud.2015.55.
[49]. PS-24. K. Zhang et al., “Design of Domain-Driven Microservices-Based Software Talent Evaluation and Recommendation System,” in 2022 IEEE ICEKIM, 2022, doi: 10.1109/ICEKIM55072.2022.00076.
[50]. PS-25. Q. Li, W. Sun, and R. Ma, “Sharing Platform of Digital Specimen of Wood Canker Based on WebGIS in Xinjiang Province: Architecture, Design and Implementation,” in 2022 IEEE CIPAE, 2022, doi: 10.1109/CIPAE55637.2022.00029.
[51]. PS-26. T. Raffin et al., “A Reference Architecture for the Operationalization of Machine Learning Models in Manufacturing,” in Procedia CIRP, vol. 2022, 2022, doi: 10.1016/j.procir.2022.10.062.
[52]. PS-27. C. Batista et al., “Towards a Multi-Tenant Microservice Architecture: An Industrial Experience,” in 2022 IEEE COMPSAC, 2022, doi: 10.1109/COMPSAC54236.2022.00100.
[53]. PS-28. C. Praschl et al., “Domain-Driven Design as Model Contract in Full-Stack Development,” in 2023 IEEE ICECCME, 2023, doi: 10.1109/ICECCME57830.2023.10252654.

[54]. PS-29. N. Legowo et al., "Designing Service Oriented Architecture Model in Sehatin Application with a Domain-Driven Design Approach," in 2023 IEEE ICIMTech, 2023, doi: 10.1109/ICIMTech59029.2023.10278057.
[55]. PS-30. A. Macías et al., "Architecting Digital Twins Using a Domain-Driven Design-Based Approach," in 2023, doi: 10.13039/501100011033.
[56]. PS-31. I. V. P. and V. P. H., "An Approach to Clean Architecture for Microservices Using Python," in 2023 IEEE CSITSS, 2023, doi: 10.1109/CSITSS60515.2023.10334229.
[57]. PS-32. M. Saidi, A. Tissaoui, and S. Faiz, "A DDD Approach Towards Automatic Migration To Microservices," in 2023 IEEE IC_ASET, 2023, doi: 10.1109/IC_ASET58101.2023.10150522.
[58]. PS-33. M. Camilli et al., "Actor-Driven Decomposition of Microservices through Multi-Level Scalability Assessment," in 2023, doi: 10.1145/3583563.
[59]. PS-34. O. Özkan, Ö. Babur, and M. van den Brand, "Refactoring with Domain-Driven Design in an Industrial Context: An Action Research Report," in Software: Practice and Experience, 2023, doi: 10.1007/s10664-023-10310-1.
[60]. PS-35. E. T. Nordli et al., "Migrating Monoliths to Cloud-Native Microservices for Customizable SaaS," in Information and Software Technology, vol. 2023, 2023, doi: 10.1016/j.infsof.2023.107230.

Информация об авторах / Information about authors

Хосуэ САНГАБРИЭЛЬ-АЛАРКОН – инженер-программист, разработчик программного обеспечения Университете Веракруса (Мексика). Сфера научных интересов: архитектура программного обеспечения, проектирование программного обеспечения, инженерия требований, моделирование данных.

Josué SANGABRIEL-ALARCÓN – Software Engineer. Software Developer at Universidad Veracruzana, Mexico (University of Veracruz). Research interests: software architecture, software design, requirements engineering, data modeling.

Хорхе Октавио ОЧАРАН-ЭРНАНДЕС имеет степень PhD по программированию, доцент факультета статистики и информатики Университета Веракруса (Мексика). Сфера научных интересов: архитектура программного обеспечения, инженерия требований, программная инженерия, разработка прикладных интерфейсов.

Jorge Octavio OCHARÁN-HERNÁNDEZ – PhD in Computer Science, Associate Professor at the Facultad de Estadística e Informática, Universidad Veracruzana, Mexico (School of Statistics and Informatics, University of Veracruz) since 2017. Research interests: software architecture, requirements engineering, software engineering, API design.

Ксавьер ЛИМОН имеет степень PhD по искусственному интеллекту, доцент факультета статистики и информатики Университета Веракруса (Мексика). Сфера научных интересов: интеллектуальный анализ данных, мультиагентные и распределенные системы, архитектура программного обеспечения.

Xavier LIMÓN – PhD in Artificial Intelligence, Associate Professor at Facultad de Estadística e Informática, Universidad Veracruzana, Mexico (School of Statistics and Informatics, University of Veracruz). Research interests: data mining, multiagent systems, distributed systems, software architecture.

М. Карен КОРТЕС-ВЕРДИН имеет степень PhD по искусственному интеллекту, профессор факультета статистики и информатики Университета Веракруса (Мексика). Сфера научных интересов: программные продуктовые линии, архитектуры программного обеспечения, аспектно-ориентированное программирование, разработка программного обеспечения, ориентированного на решение конкретных задач, процессы программирования, качество программного обеспечения, моделирование программ.

María Karen CORTÉS-VERDÍN – PhD in Computer Science, Professor at School of Statistics and Informatics, Universidad Veracruzana, Mexico. Research interests: software product lines, software architectures, aspect-oriented software development, concern-oriented software development, software process, software quality, software modeling.