Empirical Software Engineering (2024) 29:8
https://doi.org/10.1007/s10664-023-10417-5

Insights into software development approaches: mining Q&A repositories

Arif Ali Khan[1] · Javed Ali Khan[2] · Muhammad Azeem Akbar[3] · Peng Zhou[4] · Mahdi Fahmideh[5]

Accepted: 31 October 2023

Abstract

Context Software practitioners adopt approaches like DevOps, Scrum, and Waterfall for high-quality software development. However, limited research has been conducted on exploring software development approaches concerning practitioners' discussions on Q&A forums.
Objective We conducted an empirical study to analyze developers' discussions on Q&A forums to gain insights into software development approaches in practice.
Method We analyzed 13,903 developers' posts across Stack Overflow (SO), Software Engineering Stack Exchange (SESE), and Project Management Stack Exchange (PMSE) forums. A mixed method approach, consisting of the topic modeling technique (i.e., Latent Dirichlet Allocation (LDA)) and qualitative analysis, is used to identify frequently discussed topics of software development approaches, trends (popular, difficult topics), and the challenges faced by practitioners in adopting different software development approaches.
Findings We identified 15 frequently mentioned software development approaches topics on Q&A sites and observed an increase in trends for the top-3 most difficult topics requiring more attention. Finally, our study identified 49 challenges faced by practitioners while deploying various software development approaches, and we subsequently created a thematic map to represent these findings.
Conclusions The study findings serve as a useful resource for practitioners to overcome challenges, stay informed about current trends, and ultimately improve the quality of software products they develop.

Communicated by: Sebastian Baltes

✉ Arif Ali Khan
arif.khan@oulu.fi

[1] M3S Empirical Software Engineering Research Unit, University of Oulu, Oulu FI-90014, Finland

[2] Department of Computer Science, School of Physics, engineering and computer science, University of Hertfordshire, Hatfield AL10 9AB, UK

[3] Software Engineering Department, Lappeenranta-Lahti University of Technology, Lappeenranta 15210, Finland

[4] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

[5] School of Business, University of Southern Queensland, Springfield QLD 4350, Australia

Published online: 23 November 2023

Keyword Software development approaches · Q&A websites · Software process improvement · Software repositories mining

1 Introduction

To deliver a high-quality software product, it is essential to adopt systematic and quantifiable approaches across all software development lifecycle activities (Al-Saqqa et al. 2020). Careful selection and implementation of appropriate software development approaches can significantly enhance the business value and user satisfaction of software systems (Bano et al. 2017). To ensure that the product's value and quality are maintained, it is imperative that software development activities are conducted in a formal and controlled manner.

Software development is a complex process that involves a variety of approaches aimed at improving the quality and efficiency of software products. In this study, we use the term "development approach" to denote the varied strategies, methodologies, and processes employed in the field of software development (Pressman 2005). This encompasses traditional methodologies such as Waterfall, as well as more recent approaches like Agile, DevOps, hybrid, and beyond. The term also includes the practices and procedures that these methodologies encapsulate. Moreover, in this study, we did not differentiate between industrial and academic contexts; instead, it provides a holistic view of development approaches as they manifest across the software development landscape.

The literature and industry commonly employ various agile and traditional approaches, including Scrum, DevOps, Kanban, Extreme Programming (XP), Lean Development, Waterfall, and V-Model (Kuhrmann et al. 2017, 2021). These approaches are based on different concepts, such as plan-driven, iterative, incremental, or lean development (Kuhrmann et al. 2017). Additionally, a hybrid development approach, defined as any combination of agile and traditional methodologies is utilized to design complex software systems (Kuhrmann et al. 2017, 2021). In the literature, various studies have focused on specific software development approaches. For example, (Kuhrmann et al. 2017) discussed the hybrid software development approaches, (Khan et al. 2021) focused on agile trends in globally distributed software development environments, (Kuhrmann et al. 2019) examined software development approaches in academia, and Riaz (2019) studied the challenges and benefits of Kanban implementation.

However, to the best of our knowledge, there is a paucity of research exploring different aspects of software development approaches using data mining and analysis of relevant information obtained from Q&A online platforms such as Stack Exchange (SE) [1], that includes SO [2], SESE [3], and PMSE [4]. These online platforms offer a valuable source of knowledge for development teams, where practitioners engage in discussions on diverse software development approaches, challenges, and concerns (Ali Khan et al., 2020; Khan et al., 2019).

Recently, researchers have used supervised (deep & machine learning) and unsupervised learning (topic Modelling) techniques to analyze a wide range of software development-related issues encountered by developers on Q&A websites (Vidoni 2022). These studies have proposed automated methods to extract developers' discussion posts on various topics, such as challenges in continuous software engineering (Zahedi et al. 2020), identifying challenges in Docker development (Haque et al. 2020), security vulnerability (Le et al. 2021),

[1] https://stackexchange.com/ accessed on 3/04/2022

[2] https://stackoverflow.com/ accessed on 3/04/2022

[3] https://softwareengineering.stackexchange.com/ accessed on 3/04/2022

[4] https://pm.stackexchange.com/ accessed on 3/04/2022

software bug and rationale detection (Ullah et al. 2023; Zhou et al. 2020), developers' communications and their implications (Brisson et al. 2020), non-functional requirements (Paixão et al. 2017), design patterns (Dwivedi et al. 2018), and software maintenance and evolution (Khan et al. 2022; Sun et al. 2015). Nonetheless, we have not found any research that specifically investigates developers' discussions on Q&A platforms with a focus on software development approaches. Therefore, this study aims to investigate and analyze the developers' discussion posts on core relevant SE platforms (including SO, SESE, and PMSE) in detail, aiming to provide valuable insights into software development approaches.

This study aims to fill this significant gap by critically analyzing and mining the topics, recent trends, and the challenges reported by the software developers across the three core platforms of SE previously mentioned. We assume that this work will help address common challenges developers face. It is harder for practitioners to decide where to focus their efforts without such information. As a result, we have proposed the following research questions (RQs) to accomplish our stated objectives:

- RQ1: How successfully the questions related to software development approaches are answered?
- RQ2: What are the software development approaches discussion topics?
- RQ2.1: What are the most popular and difficult topics?
- RQ3: What challenges arise when practitioners implement software development approaches, given the identified topics?

In our quest to answer the aforementioned RQs, we have conducted an empirical study, collected 13903 software development approaches related to posts and used a mixed method approach that consists of LDA (Blei et al. 2003) topic modeling and qualitative analysis to achieve the study objective. Furthermore, quantitive measurements are used to portray useful information for the software developers, which will help in improving software quality and user satisfaction by promptly incorporating the following research findings of this study:

(1) The number of practitioners' questions related to software development approaches has been on a gradual rise. However, the trend of developers' questions (received accepted answers) in this research domain has been declining since 2014.
(2) Fifteen commonly discussed software development approaches-related topics were identified using the LDA model. In our study, LDA model was utilized purely at the application level. The objective was to explore software development approach-related questions from developers, a unique context, leveraging LDA's capacity to glean insights from large unstructured text data.
(3) Among the fifteen topics mentioned above, the popular and difficult ones were identified, and the top three most difficult topics exhibited an upward trend over time. A correlation analysis using Kendall's Tau correlation test (Kendall 1938) revealed a strong negative correlation between topic popularity and difficulty, indicating a decline in popularity as the difficulty of the topic increases.
(4) We identified 49 challenges practitioners faced when deploying the software development approach, which were organized into 14 sub-themes and 4 higher-level themes.

The insights gained from the study findings can aid researchers in advancing their investigations to improve software development activities. Furthermore, software practitioners can leverage the results to better understand the core aspects of software development approaches. Equipped with this understanding, developers can proactively prepare to establish advanced practices in software development and tackle challenges that may arise throughout the development process.

This paper consists of the following seven sections. The study motivation is discussed in Section 2. The related work to analyze the existing relevant studies is presented in Section 3. Section 4 details the research methodology. The findings of this study are presented in Section 5 and the implications are reported in Section 6. The threats to the validity are highlighted in Section 7, and the conclusion, along with future avenues are presented in Section 8.

2 Motivation

Software companies, teams, and individual developers have been searching for effective and efficient software development approaches for decades. The Waterfall model, established in 1970, was the first mature development approach (Royce 1987), followed by the Spiral model (Boehm 1988), and subsequently, Lean and Agile development approaches (Beck et al. 2001), also including Scrum and present-day DevOps (Kim et al. 2021). As software development activities rapidly evolve, an increasing number of development practices and tools have emerged, drawing from the concepts of the mentioned development approaches. To deliver high-quality products and stay competitive in the market, software practitioners must not only conduct development activities well but also possess knowledge of team management (e.g., the collaboration of developers and operations), time management (e.g., estimation of user stories), and tools usage (e.g., Kanban tools and Docker) (Kim et al. 2021). Selecting the appropriate development approach for an organization, team, or project is challenging, as software development approaches involve a complex set of practices (Kuhrmann et al. 2017). Due to various development environments and complex software products, it is not feasible to have a single software development approach that fits all Kuhrmann et al. (2017, 2021). According to Klünder et al. (2017), software practitioners do not always strictly follow defined development processes and practices. Their study highlighted that practitioners' perspectives on software development approaches differ from academic research and the unpredictable challenges that arise when using formal software development processes in an industrial setting. Marco et al. Kuhrmann et al. (2017) suggested that practitioners use traditional development approaches as a framework and adopt agile methods at specific stages of the software development life cycle, resulting in hybrid development approaches (Kuhrmann et al. 2021).

Therefore, understanding various aspects of software development approaches is crucial for software practitioners to tailor a development approach that aligns with their project's unique requirements. However, there is a lack of comprehensive knowledge and guidance that practitioners and researchers can use to achieve this understanding. This gap prompted us to investigate the different aspects of software development approaches and provide a structured understanding that helps practitioners in enhancing software quality and contribute to the overall project success. In literature, various studies have investigated software development approaches and practices in the industrial domain (Kuhrmann et al. 2017; Zhou et al. 2021; Aymerich et al. 2018; Mushashu and Mtebe 2019). However, research specifically focusing on examining developers' discussions regarding software development approaches across Q&A sites is lacking. Since software practitioners often seek information for resolving challenges from Q&A sites, which contain a vast amount of data, exploring these sites with respect to software development approaches is crucial.

Hypotheticalmotivatingscenario : Prior to rationalising our research questions, we have devised an exemplar motivating scenario. This scenario serves to contextualize our

research problem, validate the necessity of our study, engage our readers, and demonstrate the real-world consequences and potential advantages of our work.

Imagine a startup, *TechInnov*, tasked with developing innovative smart home solutions. The team, a mix of seasoned professionals and recent graduates, brings diverse approaches to software development. With a fast-paced market waiting and deadlines looming, the team faces tension due to differing adherence to Waterfall and Agile methodologies, causing miscommunication and delays.

Sam, the team lead, is caught in a challenging situation, for example:

- How to select a fitting software development approach for his team and project?
- How to balance traditional approaches and Agile practices?
- What tools should the team use for effective workflow?

Sam needs guidance on how software practitioners manage similar situations. He often turns to repositories mining studies on Q&A sites for guidance based on the practitioners insights. However, he realizes that these resources fall short of providing a comprehensive study that examines the various aspects of software development approaches in detail (Storey et al. 2014; Barua et al. 2014). This is where our research comes in and it aims to bridge this knowledge gap, providing a structured understanding of software development approaches and insights from Q&A site discussions, benefiting practitioners like Sam.

3 Related Work

We now summarized the related studies focusing on software development approaches (Section 3.1), and Q&A software repositories mining (Section 3.2).

3.1 Software Development Approaches

Software development approaches consistently evolve because of the rapid, iterative, and complex software development environments and domain problems. Bajec et al. (2007) addresses the issue of software development methods lacking adaptability to project-specific situations. They reveal that developers often avoid using methods that exist only on paper, as they fail to cater to the unique needs of different projects. The authors propose a new approach called “Process Configuration" to create project-specific methods from existing ones, considering the project’s requirements. The proposed approach offers increased flexibility and is easier to implement compared to other existing approaches.

Marco et al. Kuhrmann et al. (2021) conducted a large-scale international survey to investigate the factors that make a software development method agile. They analyze the perceived degree of agility in various project disciplines, development methods, and practices. Findings indicate that most projects exhibit increasing degrees of agility, with the selection of practices having a stronger impact than the methods used. The study concludes that agility cannot be defined solely at the process level, and additional factors must be considered when implementing or improving agility in a software company.

Bustard et al. (2013) conducted an industrial survey to observe the principles and practices of agile development approaches adopted in 2010 and 2012. The research offers insights into the nature and practice of agile development, highlighting key outcomes and trends. The study findings further reveal that agile practices had been widely used before 2012. Since then, there has been a growing tendency to adopt agile methods.

The more recent trend of integrating agile and traditional development approaches (i.e.hybrid development approaches) is becoming more common in practice. Tell et al. (Tell et al., 2021) explores the construction of hybrid software development methods by analyzing 1467 data points from a large-scale practitioner survey. The findings reveal that modern software development consists of eight core approaches and a few practices. The study proposes a systematic construction approach for hybrid methods, characterized by the practices they include. Using an 85% agreement level, the researchers present examples of hybrid methods and introduce an initial construction procedure to define a method frame and incrementally enrich it with ranked sets of practices.

Additionally, the HELENA study conducted by Klünder et al. (2017) gained insights into the distribution of hybrid approaches. Preliminary findings suggest that combining traditional and agile software development approaches provides an opportunity to deliver a software product in a continuous loop, get frequent feedback, and improve overall productivity. This study's findings only cover the German industrial domain.

Khan et al. (2021) conducted an industrial study aims to develop a taxonomy of factors that positively impact the scaling process of agile methods in the Chinese Global Software Development (GSD) industry. Factors are identified through a literature review and an industrial empirical study with Chinese agile and GSD practitioners. The resulting factors are categorized, prioritized, and organized into a taxonomy using the Fuzzy AHP multi-criterion decision-making approach. This taxonomy provides a valuable resource for the GSD industry to assess and improve the scaling process of agile methods.

3.2 Mining Software Issues Across Q&A Repositories

Various studies used an unsupervised learning approach (topic modeling) to mine software development-related issues from different repositories.

For instance, (Barua et al. 2014) used the LDA approach to analyze the textual content of SO discussions and automatically discover the main topics present in developer conversations. This approach differs from prior work, which focused on user activities or social interactions in Q&A websites. By analyzing the discovered topics, their relationships, and trends over time, the authors gained valuable insights into the development community. They observed a wide range of topics of interest to developers, including jobs, version control systems, and C# syntax. Additionally, they found that some questions led to discussions on other topics and that the most popular topics over time were web development (particularly jQuery), mobile applications (especially Android), Git, and MySQL. This analysis helps the software engineering community better understand the thoughts and needs of developers.

In a continuous software engineering (CSE) domain, (Zahedi et al. 2020) used the LDA approach to empirically investigate the practitioners' perspectives by mining Q&A discussions on the SO platform. They used a topic modelling approach to identify dominant topics and conduct qualitative analysis to pinpoint key challenges. The study found that questions are becoming more technology-specific and harder to answer. Among the 32 identified topics, "Error messages in Continuous Integration/Deployment" and "Continuous Integration concepts" were the most dominant. The paper also highlights the most challenging areas in CSE from practitioners' viewpoints.

Haque et al. (2020) conducted a a large-scale empirical investigation on Docker technology by mining 113,922 Docker-related posts from the SO community. Using the LDA approach for topic modeling, the authors identified 30 topics grouped into 13 main categories, with the majority of posts belonging to application development, configuration, and networking

categories. The study found that monitoring status, transferring data, and authenticating users were particularly popular topics among developers. It also revealed challenges faced by developers in areas such as web browser issues, networking errors, and memory management, as well as a lack of experts in the domain. The findings are expected to guide future research on the development of new tools and techniques and help the community focus their efforts on Docker-related topics.

Le et al. (2021) analyse developers' Security Vulnerabilities (SVs) discussions on two major Q&A websites, SO and Security StackExchange (SSE). Using topic modeling, they examined 71,329 SV posts to identify 13 main discussion topics. The study found that these topics did not necessarily align with expert-based security sources like Common Weakness Enumeration (CWE) and Open Web Application Security Project (OWASP). The analysis also revealed that while SV discussions tend to attract more expert answers than other domains, some difficult topics, such as Vulnerability Scanning Tools, receive limited expert support. Furthermore, the authors identified seven key types of answers to SV questions, with SO often providing code and instructions, while SSE offers experience-based advice and explanations. The findings aim to help researchers and practitioners effectively acquire, share, and leverage SV knowledge on Q&A websites.

Vidoni (2022) identified the need for guidelines on conducting systematic Mining Software Repositories (MSR) studies, as existing research often lacks a systematic approach for repository selection and data extraction. They conducted a systematic literature review of MSR studies. The results showed that many MSR studies do not report selection or data extraction protocols and rarely discuss threats to validity due to the selection or data extraction steps. The authors concluded that there is a need for guidelines on conducting systematic MSR studies and proposed new guidelines and a template to consolidate related studies and strategies for systematic literature reviews in the MSR field.

3.3 Comparative Analysis

The comparative analysis of our work against existing related studies is presented in Table 1. The results demonstrate a significant distinction between our findings and those of previous related studies. In contrast to previous studies, our research uniquely bridges the domains of software development approaches and Q&A repositories mining. For instance, the work of Tell et al. (2021) focused primarily on the exploration and construction of hybrid software development approaches, analyzing a large volume of data from a practitioner survey. While this research significantly contributed to the evolution of software development approaches, it did not consider the rich insights that can be extracted from Q&A platforms.

Similarly, Marco et al. Kuhrmann et al. (2021) conducted a large-scale international survey to investigate the factors that make a software development method agile. Their work, which looked into various project disciplines, development methods, and practices, indeed advanced our understanding of agility in software development. However, like Tell et al. (2021) they did not tap into the extensive real-world data available on Q&A platforms, which could provide further empirical insights and understanding of agile practices in various practical contexts.

On the other hand, studies like that of Zahedi et al. (2020) and Le et al. (2021) brilliantly harnessed topic modeling techniques to mine Q&A platforms for insights into continuous software engineering challenges and security vulnerabilities, respectively. These works, while mining Q&A repositories, did not specifically target the examination of software development approaches as their primary research objective.

As indicated in Table 1, our research stands uniquely at the intersection of two crucial areas: software development approaches and Q&A repositories mining, filling a significant gap in the literature. By using mining techniques to scrutinize software development approaches as reflected in Q&A discussions, we aim to unearth practical insights about these approaches that are directly tied to developers' preferences, challenges, and interactions. This nuanced perspective has the potential to guide future academic research and to inform practical enhancements in software development methodologies.

4 Research Methodology

4.1 Research Questions

This study investigated four research questions to identify common software development approaches topics, trends, and challenges to help software practitioners improve their development activities. To answer these RQs, we retrieved 13903 software development approaches related to posts from SO, SESE, and SEPM using the approach described in Section 4.2.

RQ1: How successfully the questions related to software development approaches are answered?

Rationale : Taking inspiration from existing relevant studies (Zahedi et al. 2020; Pinto et al. 2014; Treude et al. 2011), RQ1 examines the nature of responses to questions related to software development approaches. The investigation of successful, ordinary, and unsuccessful questions is a strategic approach to understand the dynamics within the software development domain. While we value the depth and quality of expert answers, however, our

Table 1 Comparative analysis of related works and our study. Note: (✓: included, X: not included, +: simple overview)

Studies	Software Development Approaches	Repositories Mining	Topics popularity & difficulty	Challenges	Approaches via Q&A
Bajec et al. (2007)	✓	X	X	X	X
Kuhrmann et al. (2021)	✓	X	X	✓ (+)	X
Bustard et al. (2013)	✓	X	X	X	X
Tell et al. (2021)	✓	X	X	X	X
Klünder et al. (2017)	✓	X	X	X	X
Khan et al. (2021)	✓	X	X	X	X
Barua et al. (2014)	X	✓	X	X	X
Zahedi et al. (2020)	X	✓	✓	✓	X
Haque et al. (2020)	X	✓	✓	✓ (+)	X
Le et al. (2021)	X	✓	✓	X	X
Vidoni (2022)	X	✓	X	X	X
Our Study	✓	✓	✓	✓	✓

focus is more to gauge overall community engagement. The number of answers can hint at the community's interest and involvement. By categorizing questions as successful, ordinary, or unsuccessful, we aim to identify trends and areas needing more expert insight. Our rationale combines both quantity and quality to capture the essence of interactions in the software development Q&A domain.

Method : Initially, we computed the average number of answers for questions related to software development approaches and compared it to the respective value, taking inspiration from Zahedi et al. (2020). Additionally, we analyzed the question that received the most answers to gain a more comprehensive understanding of the success rate of addressing questions in the software development approaches domain.

Drawing on previous research (Zahedi et al. 2020; Pinto et al. 2014; Treude et al. 2011), we classified questions related to software development approaches into three categories:

- Successful questions: Resolved questions
- Ordinary questions: those that received answers but no accepted answer
- Unsuccessful questions: Unresolved questions

Following this classification, we performed frequency analysis and counted questions for each category. To further understand the evolving trends in the data, we applied a third-order polynomial regression (Hastie et al. 2009). This combined approach allowed us to provide both a clear overview and detailed insights into the dataset's dynamics. We then calculated the distribution of these categories and illustrated the growth trends for all questions- successful questions, ordinary questions, and unsuccessful questions within this domain.

RQ2: What are the software development approaches discussion topics?

Rationale : Identifying the key discussion topics in Q&A repositories provides an overview of the current software development landscape, keeping abreast of evolving trends and practices. This understanding can guide practitioners in choosing and implementing relevant software development approaches tailored to their projects, fostering efficiency and success. Thus, RQ2 not only illuminates the current state of software development discussions but also paves the way for more informed decision-making in both research and practice.

Method : Building upon the work of Mansooreh et al. (Zahedi et al. 2020), we employed the LDA (Blei et al. 2003) topic modeling technique (Section 4.3) to identify most common discussion topics related to software development approaches. Topic modeling is a Natural Language Processing (NLP) technique that automatically extracts structured information from a collection of documents (Chen et al. 2016). For the LDA model, we considered a question's title, body, and corresponding answers as a single input document, and the output consisted of the most frequently occurring topics identified in the text corpus. Moreover, in our study, LDA model has been utilized purely at the application level. The objective was to explore questions related to software development approaches raised by developers using the potential of LDA to extract insights from the large unstructured text data.

RQ2.1: What are the most popular and difficult topics?

Rationale : This RQ is developed with the aim of understanding the types of identified software development approaches related to topics that are popular (attractive) to practitioners. Moreover, we also look at the topics that the practitioners considered challenging (difficult) to answer. We investigated the topic's popularity and difficulty based on similar previously conducted studies (Peruma et al. 2022; Bagherzadeh and Khatchadourian 2019; Abdellatif et al. 2020; Zahedi et al. 2020).

The *popularity* of topics highlights the methodologies or practices that are currently in high demand or present notable challenges to the developer community. These topics can guide the allocation of resources, such as targeted training or tool development, to address

the complexities associated with these popular approaches. These topics can also steer the direction of future research and development in software engineering, ensuring that the focus remains on improving and innovating the most relevant development approaches.

The aim of examining topic *difficulty* is to delve into the complexity of responding to posts within each topic. By discerning if certain topics pose more difficulties in generating responses than others, we can pinpoint the areas that require increased community engagement. Furthermore, this investigation enables us to underscore the topics that necessitate the development of improved frameworks or tools, thereby aiding developers in effectively tackling the challenges associated with these topics.

Method : Drawing from the work of Ahmed and Bagherzadeh (2018); Rosen and Shihab (2016); Le et al. (2021), we utilized average values of (P1) views, (P2) score, (P3) favorite count, and (P4) comments to gauge the popularity of developers' topics. Concurrently, we computed three metrics-(D1) percentage of accepted answers, (D2) median duration (minutes) to receive an accepted answer since creation, and (D3) average percentage of answers to views-to assess topic difficulty. Intuitively, a more popular developers' topic would exhibit higher average views, scores, favorite count, and comments, and vice versa. In contrast, a more challenging topic would have a higher D2 value but lower D1 and D3 difficulty metric values.

To generate a more uniform and representative value across various developers' topics, we calculated the reciprocals of D1 and D3 and then determined the geometric mean (Equation 6) of popularity (1) and difficulty (2) metrics. We opted for the geometric mean over the arithmetic mean due to the different metric units. To enhance the comparison of all topics' popularity and difficulty, we normalized the popularity (4) and difficulty values (5) using Min-max normalization (5).

$$Popularity_i = G(P1_i, P2_i, P3_i, P4_i) \tag{1}$$

$$Difficulty_i = G(1/D1_i, D2_i, 1/D3_i) \tag{2}$$

$$Normalized\_Popularity_i = N(Popularity_i) \tag{3}$$

$$Normalized\_Difficulty_i = N(Difficulty_i) \tag{4}$$

$$N(x_i) = \frac{x_i - min(x)}{max(x) - min(x)} \tag{5}$$

$$G(x_1, x_2, \cdots, x_n) = (\prod_{i=1}^{n} x_i)^{\frac{1}{n}} \tag{6}$$

RQ3: What challenges arise when practitioners implement software development approaches, given the identified topics?

Rationale : RQ3 identified the critical challenges faced by practitioners concerning various activities within software development approaches. By being aware of the challenges, practitioners can make informed choices when selecting and adopting software development approaches, considering potential obstacles, and planning for contingencies to ensure project success.

Method : We firstly adopted the Accumulated Post Score (AMS) formula (7) proposed by Bajaj et al. (2014) to rank the collected developer's posts. AMS is a metric often used in studies related to online Q&A forums, like SO. It is generally utilized to measure the rank or significance of a post, which is often determined by user interactions like upvotes, downvotes, comments, or answer count.

$$AMS_i = 3U_i - 25D_i + 10C_i + A_i + F_i \tag{7}$$

Where U_i, D_i, C_i, A_i, and F_i represent $developersPost_i$ upvotes, downvotes, comment count, answer account, and favorite count, respectively. The equation considers the above factors to rank the developer's posts and identify their importance. By computing AMS for all collected developer posts in the data set, we selected the top 200 questions with the highest score for manually analyzing and evaluating. We used the thematic analysis approach proposed by Braun and Clarke (2006) for qualitative data analysis. The thematic synthesis in software engineering research is considered flexible and can be used to produce an insightful synthesis (Cruzes and Dyba 2011).

We employed the AMS metric proposed in Bajaj et al. (2014) and identified the top 200 posts based on their accumulated post score. This selection criterion was chosen as these posts, by virtue of their high scores, likely represent the most impactful or resonant posts. After examining these 200 posts, we realised recurring insights, indicating that this subset was rich in information and representative of the selected dataset. Furthermore, considering our resource constraints, analyzing 200 posts offered an optimal balance between analytical depth and coverage breadth. Thus, the selected posts reflect a comprehensive and representative sample to analyze the critical challenges practitioners encounter while implementing software development approaches.

Furthermore, rigorous process involving all authors was implemented to uphold the integrity and quality of data coding in the thematic process. Primarily, the first two authors used the MAXQDA[5] tool for data analysis. However, to evaluate the interpersonal bias and degree of agreement, the third and fourth authors were also invited and asked to randomly select and analyzed 20 posts from the dataset. We then used Cohen's Kappa test (Cohen 1960) to measure the agreement level between the authors and ensure there were not significant differences in their data coding approach. Cohen's kappa coefficient (k) is simply "the proportion of chance-expected disagreements which do not occur, or alternatively, it is the proportion of agreement after chance the agreement is removed from consideration" (Cohen 1960). The (k) coefficient assesses the agreement level among a group of raters that evaluate N-objects into (c) mutually exclusive categories (Cohen 1960). In simple terms, a (k) value of 0 means the agreement could simply be by chance, while a positive score suggests a greater level of agreement than chance and a negative value means less agreement than what chance would predict. Perfect agreement level is indicated by a (k) value between 0.81 to +1.00., as proposed by Landis and Koch (1977) in their seminal study on observer agreement. We wrote Python code to conduct the test and run using Google "Colab"[6] platform. The sample code is provided in the replication package (Khan et al. 2022). We obtained the Cohen's Kappa coefficient value (k=0.77) which is positive and substantial agreement based on the divisions provided by Landis and Koch (1977). Thus, we can confidently say that no significant level of personal bias exists between the authors for the data coding process.

Once the initial codes were generated, all the authors were invited to participate, with the objective of critically analyzing and discussing the revised set of codes. The intention was to foster collective knowledge exchange and contribute to the robustness of the coded data. Throughout the meeting, any disagreements or conflicts related to the final code generation were addressed through mutual discussion and brainstorming. This served as a platform for the authors to engage in constructive dialogue, which enhanced collective understanding and ensured the validity of the final codes. This iterative process, emphasizing the value of collaboration and open discussion, underpinned the creation of a high-quality thematic analysis. We finally identified 49 codes, and then these codes were further mapped into

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

[6] https://colab.google/

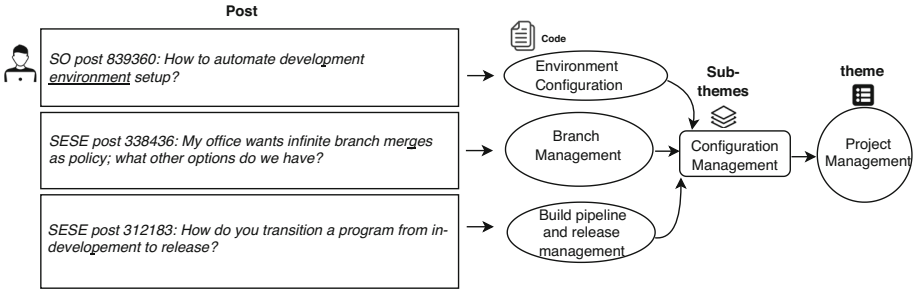

Fig. 1 Translation of codes into themes

14 sub-themes and 4 higher-order themes. The sample analysis of the proposed thematic synthesis process is depicted in Fig. 1.

4.2 Data Collection

To examine the support provided by Q&A sites for discussions on software development approaches, we gathered research data from three popular software Q&A repositories: SO, SESE, and PMSE. SO is the best-known online platform for developers to ask questions, learn, and share their programming knowledge. SESE is an online community for academics and developers to ask questions related to the software development activities. Additionally, PMSE is a public platform for project managers to ask questions and receive answers. Many developers post software development-related queries on these platforms to obtain recommendations and frequent assistance from experts. Such information can help software practitioners improve the existing software development process and enhance product quality.

To collect data from the selected Q&A platforms, we embarked on a systematic approach to identify the relevant tags. The second and fourth authors took the lead in the initial tag selection process, drawing the commonly recognized software development approaches (e.g., Scrum, Domain-Driven Design, and DevOps) from relevant literature studies such as Zhou et al. (2021); Klünder et al. (2017); Kuhrmann et al. (2017, 2019); Khan et al. (2021); Kuhrmann et al. (2017); Al-Saqqa et al. (2020); Bajec et al. (2007). To enhance the breadth of our search, we also incorporated pivotal concepts integral to the software development realm, including terms like the development process, agile methodologies, and the software development life cycle. Recognizing the diverse nomenclature in the industry, we expanded our tag pool by adding commonly used aliases and abbreviations, drawing from various other online sources using google search. With this comprehensive list, we scoured the selected Q&A platforms and focused on those tags which had at least one question to ensure that they were actively relevant within the community. Each tag was reviewed carefully, where its description was scrutinized to ensure it covers our research objectives. Tags that deviated from the expected context were excluded. The first and third authors then analyzed the tags selection process performed by the second and fourth authors, confirming them based on their expertise and understanding. At the end of this rigorous process, 28 relevant tags were identified across the selected Q&A platforms as shown in Table 2. The column "selected tags" represents the software tags against which we collected developers' questions from SO, SESE, and SEPM. The columns "SO", "SESE" and "SEPM questions" display the number of developer queries collected against each tag. The column "Total" indicates the total number of developer questions across the three Q&A platforms. We excluded developers'

Table 2 Selected Tags and Question Numbers (exclude questions that contain code snippet)

No.	Selected Tags	SO questions	SESE questions	PMSE questions	Total
1	domain-driven-design	3507	835	N/A	4497
2	agile	965	1093	1351	3410
3	devops	3156	73	21	3342
4	Scrum	729	717	1534	2981
5	development-process	31	676	104	813
6	kanban	173	47	309	532
7	waterfall	72	46	68	191
8	sdlc	64	88	11	163
9	extreme-programming	45	55	17	117
10	agile-project-management	77	N/A	N/A	77
11	prince2	N/A	3	56	59
12	agile-processes	53	N/A	N/A	53
13	mda	47	N/A	N/A	47
14	kanban-board	N/A	N/A	42	42
15	model-driven-development	39	N/A	N/A	39
16	rational-unified-process	18	13	N/A	31
17	rup	23	N/A	5	28
18	safe	N/A	N/A	24	24
19	iterative-development	N/A	22	N/A	22
20	Lean	N/A	22	N/A	22
21	scrumban	N/A	N/A	17	17
22	dsdm	N/A	N/A	12	12
23	scaled-agile-framework	N/A	9	N/A	9
24	personal-software-process	7	2	N/A	9
25	nexus	N/A	N/A	7	7
26	feature-driven	3	N/A	N/A	3
27	large-scale-scrum	N/A	3	N/A	3
28	dsdm-atern	2	N/A	N/A	2
	Total (Remove Duplicates)	8486	2978	2439	13903

posts containing code snippets in their bodies to prevent data from leaning towards a specific technology and code language directions. Furthermore, we removed repeated posts with multiple selected tags. In summary, we extracted 13,903 developers' questions along with their corresponding answers using the software tags mentioned in Table 2 from the Stack Exchange Data Dump [7] downloaded from the "Internet Archive." The data was retrieved on July 6, 2022.

4.3 Topic Modelling

Topic Modeling is an approach used in NLP that autonomously pulls out structured data, like themes, from a collection of documents (Chen et al. 2016). A set of semantically related

[7] https://archive.org/details/stackexchange_20211206

words that frequently co-occur in textual data can be considered a representation of one topic (Chen et al. 2016; Blei et al. 2003). Using a topic modeling framework to analyze a corpus of documents allows researchers to efficiently organize and index documents based on their semantic structure (Chen et al. 2016). LDA (Blei et al. 2003) is a prominent unsupervised topic modeling algorithm used for identifying underlying topics within a text corpus (Barua et al. 2014). It operates by assuming each document is a mixture of topics, and each topic is a distribution of words, thus revealing the hidden thematic structure within the text. It has been used in different other domains (Jacobi et al. 2016), such as continuous software engineering (Zahedi et al. 2020), Docker development (Haque et al. 2020), requirements engineering (Khan et al. 2018), and security vulnerability (Le et al. 2021). Following the work of Mansooreh et al. (Zahedi et al. 2020), we define a question's title, body, and corresponding answers as one input document for the LDA model and output the number of frequently occurring topics identified in the text corpus.

4.3.1 Preprocessing of Collected Posts

Before applying LDA to the dataset curated from developers' queries on the SO and SE platforms, it's essential to preprocess the collected textual documents to eliminate noise. First, we removed HTML tags, code snippets, punctuation, and stop words using the NLTK [8] stopwords corpus. For this, we identify and remove code snippets enclosed within the `<code>` tags. Specifically, we employed the pattern '`<code>.* < /code >`' to target and replace such segments with an empty string. Subsequently, to ensure a comprehensive cleanup, we further applied another regular expression, `<.*?>`, to remove any residual HTML tags from the content. We acknowledge that users occasionally overlook proper code formatting, which might lead to some code snippets being embedded within the main text. To mitigate this, both first and fourth authors conducted a manual review of a subset of the data to ensure the accuracy of our preprocessing. Our combined automated and manual efforts aimed to minimize the presence of unintended code snippets, thereby reducing potential noise in our dataset. After these steps, we converted all document texts to lowercase. We subsequently performed lemmatization (keeping only noun, adj, vb, and adv) and word stemming (converting words to their root form) to remove multi-form and irrelevant words. Finally, we adopted the approach proposed by Yang et al. (2016) that made the word frequency statistics of the data corpus and excluded the words which occurred less than 10 times.

4.3.2 Topic Modelling with LDA

After preprocessing the dataset, we applied the LDA algorithm to the collected textual documents of developers' discussions on the Q&A platforms. The number of topics "K" extracted from the training corpus is one of the most important input parameters in implementing LDA; if K is too small, the recovered topics may overlap, which is difficult to generalize. On the contrary, LDA may create excessively fine-grained topics in case of K is too large. In this regard, we observed a consecutive range of K from 2 to 50 with an increment of 1. Similar to Barua et al. (2014); Ahmed and Bagherzadeh (2018); Rosen and Shihab (2016); Le et al. (2021), alongside K, we also set hyper-parameters (α and β) values with an inclusive range from 0.01 to 1 in the step of 0.2, and an additional value "symmetric" ($1.0/numTopics$) and "asymmetric" ($1.0/(topicIndex + \sqrt{numTopics})$) for α, "symmetric" for β. α determines

[8] https://github.com/nltk/nltk

the sparsity of document-topic distribution, and β controls the sparsity of topic-word distribution. As proposed in other research studies (Rosen and Shihab 2016; Abdellatif et al. 2020), we measured the coherence score to choose the optimal number of identified topics because it highly correlates with human comprehensibility (Röder et al. 2015). Topic coherence represents the semantic correlation between high-scoring words that appeared in the topic. Hence, we trained the LDA model with each tuple of (K, α, β) and chose the top 5 highest coherence values (with different K values) with corresponding results for comparison. To validate the results, the first author manually checked the topic words and most related developers' posts for 5 candidate K values to assure the optimal K value was selected. After these steps, we found that a tuple value of (15, 0.41, 0.81) provides relatively granular topics for collected software development processes-related developers discussion in the Q&A platforms (SO and SE). Significantly, we filtered out topics with a probability less than 0.1 in a document to exclude unimportant topics adopted from Barua et al. (2014). To assign descriptive labels to each discussion topic from the Q&A platforms, we thoroughly examined the top 10 most recurrent terms. This ensured that the labels were representative of the core essence of the discussions. Furthermore, to gain a comprehensive understanding and ensure the accuracy of our interpretations, we carefully reviewed the top 15 developer discussion posts that were most closely associated with each topic. These posts were selected based on their relevance (score) to each topic, as reported in Ahmed and Bagherzadeh (2018); Le et al. (2021); Zahedi et al. (2020). Additionally, we offer a comprehensive list of topics, the top 10 words, the top 15 posts, and assigned labels in the data replication package, which can be found in the our study Notion repository (Khan et al. 2022).

5 Results

In this section, we summarize the results of the data analysis, which aims to address the core research questions of this study comprehansively.

5.1 RQ1: Successfully Answered Questions

We investigated how frequently the developers responded to questions related to software development approaches. Upon performing statistical analysis of practitioners questions that received answers, we observed that most questions in the software development approaches domain received fewer than three answers (mean=2.41). However, several developers' questions attracted considerable attention from practitioners, with a high number of responses. The developer question and its accepted answer which received a maximum of 37 responses from practitioners are presented in Fig. 2. We can observe that this question was published in 2009 on SO and pertained to project management regarding deadlines in software projects. The accepted answer to this question received many upvotes and stated, "Software is done when it is done, no sooner and no later." Interestingly, this question was closed in 2012 as it did not fit the specific topics on SO.

We categorized software development approach-related questions into three types: successful, ordinary, and unsuccessful questions, as outlined in Section 4.1 (RQ1). The distribution of these categories can be found in Table 3. Interestingly, we can conclude that most of the questions, i.e., 52% in the software development approaches domain are identified as successful. Also, only 6% developers' questions are categorized as unsuccessful and have not received any answer from the community on the SO and SE platforms. In contrast,

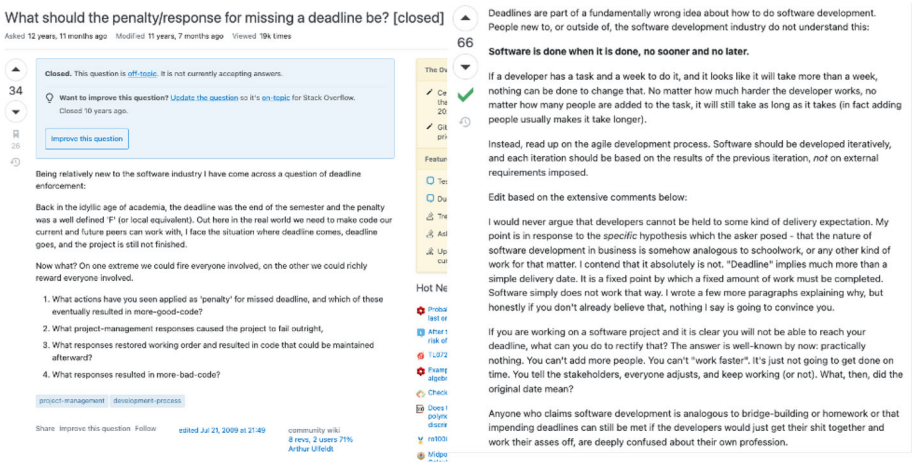

Fig. 2 The question that received maximum answers

41% developers' posts are classified as ordinary questions, where they received responses from developers but not a successful answer (See Table 3). Furthermore, we depicted the trends of the developers' questions (successful, ordinary, and unsuccessful) in Fig. 3.

Our results point out that we need more expert involvement, as some relevant questions remain unanswered. One way to tackle this is to extend the functionalities of the existing developers discussion forums like Stack Overflow by employing an automated approach that identifies topics and their concerned categories that remain unanswered and target them to the volunteers for possible solutions. Gamification-based approaches can be utilized to provide incentives, rewards, and social recognition to the frequently contributing developers on social platforms on challenging and emerging topics or issues (Khan et al. 2019). Furthermore, Machine and deep learning can be employed to automatically group the software development discussion in the Q&A forums into useful and meaningful categories, such as DevOps, streamlining the user experience for those seeking information on trending software methodologies.

In addition, by understanding the difference between questions that get answered (successful), those that get responses but not accepted answer (ordinary), and those that get no responses (unsuccessful), professionals can better identify what topics are popular, which areas need more exploration, and where the tough challenges lie. Recognizing these patterns not only helps in solving problems but also keeps everyone updated with the latest trends. Overall, these steps and insights promote a stronger exchange of knowledge in the software development community.

Table 3 Distribution of Questions (Successful, Ordinary, Unsuccessful)

	Successful	Ordinary	Unsuccessful	Total
#Posts	7299	5733	871	13903
%Posts	52%	41%	6%	100%

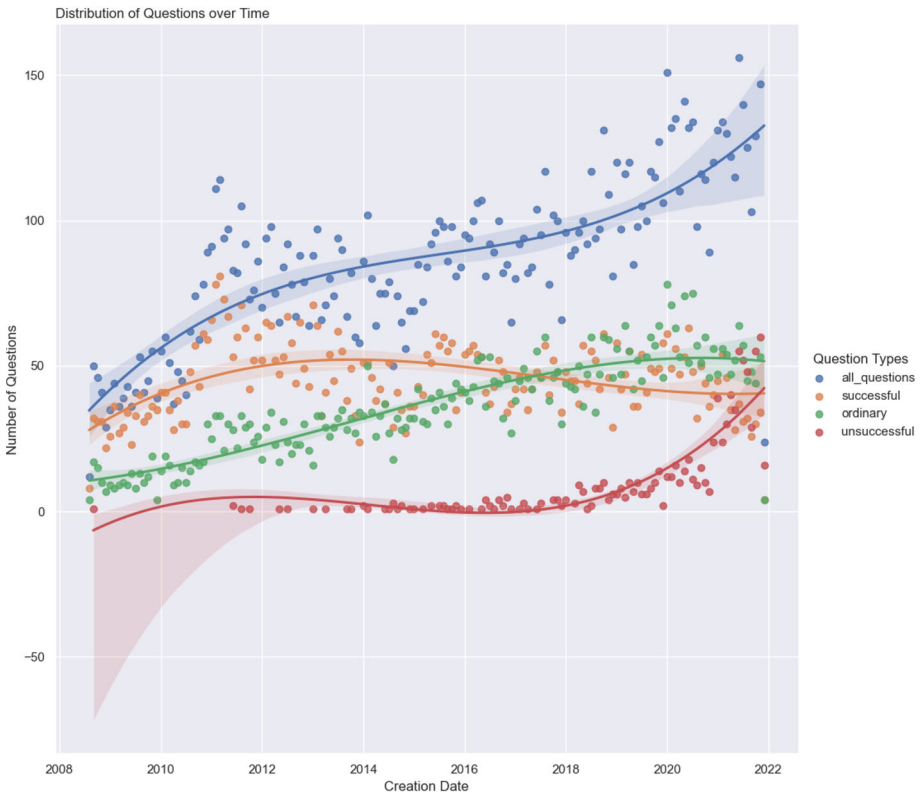

Fig. 3 Trend of Questions

Key Findings of RQ1

Finding 1: In the software development approaches domain, 52% of questions were successfully answered on Q&A platforms. However, 6% were deemed unsuccessful as they received no responses from the community. **Finding 2**: The success rate of software development approach questions has declined since 2014, indicating a need for more expert input on Q&A platforms and a potential rise in question complexity.

5.2 RQ2: Software Development Approaches Topics

To address RQ2, we used topic modeling LDA approach (see Section 4.3) and identified the optimal topics number (K = 15) and parameters (α = 0.41, β = 0.81) by comparing the coherence score, manually checking the topics keywords, and corresponding documents. After training the LDA model, we analyzed and inspected the top 10 keywords identified by the LDA algorithm and read through the top 15 developers' posts with the highest relevance to each topic. The relevant developers topics were manually selected based on the identified keywords using the LDA algorithm. After performing the above steps, we gave each developer topic a descriptive label, as shown in Table 4. The complete list of developers topics, top 10

Table 4 Identified topics using LDA

Topic	Topic Name	Number (Proportion) of questions	Trend
T1	Continues integration, build and deployment	2587 (19%)	
T2	Events, bounded contexts and Microservices in DDD	1819 (13%)	
T3	Domain model, design patterns and layers in DDD	3740 (27%)	
T4	DevOps automation tools	2809 (20%)	
T5	Project, team and time management	7933 (57%)	
T6	Team role and responsibilities in Scrum	2462 (18%)	
T7	Project managers' responsibility and contract management	905 (7%)	
T8	Entities, value Objects, and aggregates in DDD	2889 (21%)	
T9	Tools and plugins in agile software development	2833 (20%)	
T10	Software development methodology concepts	1836 (13%)	
T11	Meetings in agile team	499 (4%)	

Table 4 continued

Topic	Topic Name	Number (Proportion) of questions	Trend
T12	Task management in Kanban Board	2935 (21%)	
T13	Software testing	1479 (11%)	
T14	Story estimation in Scrum sprint	2964 (21%)	
T15	Software design and requirements	3561 (26%)	

words identified using the LDA algorithm, top 15 practitioners posts' Id and assigned labels could be found in the data replication package at Khan et al. (2022). To filter the unimportant topics, we considered a valid topic in a document whose probability related to the document was at least 10% (Barua et al. 2014). Table 4 presents a comprehensive list of the identified topics, while Fig. 4 illustrates the frequency of occurrence of each topic. Our analysis revealed that "T5: Project, team and Time Management", "T3: Domain model, design patterns and layers in DDD" and "T15: Software design and requirement" are respectively the top 3 highly asked topics in software development approaches related posts.

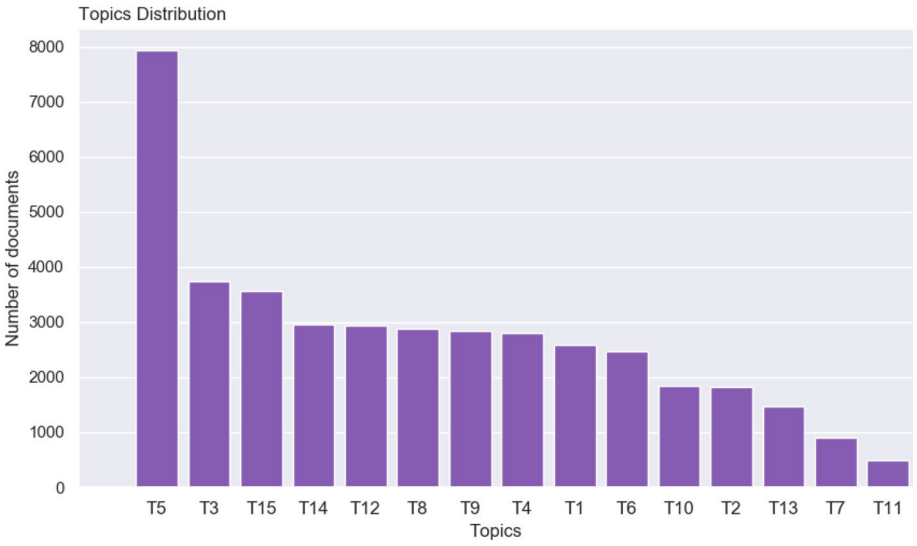

Fig. 4 Distribution of Dominant Topics

Each identified topic with sample developer's posts is elaborated as follows- aimed at facilitating comprehension of their significance.

Continues integration, build, and deployment (T1) We labeled developer topics about continuous integration (CI), continuous deployment (CD), and automated pipeline building with tools like Jenkins, Azure DevOps, and GitHub Actions as "T1". CI and CD are essential components of DevOps development and are favored by agile teams. When examining developer posts on SO and SE, we found that many practitioners and software developers struggle to adopt CI, CD, and other automated practices (*e.g. VSTS Branching Strategy & CI/CD pipelines- SO post#50694511)*), managing pull requests in workflows like *(how should we handle pull requests into hotfix/* branches?- SO post #39021924)* and controlling CI pipelines *(e.g. Run pipeline stages only after the previous stage is completed - GitLab CI- SO post #62919867)*.

Events, bounded contexts and Microservices in DDD (T2) The Code "T2" is assigned to the developers' topic in the Q&A websites that discuss the implementation issues of domain events, integration events, bounded contexts, and microservices in Domain-Driven Design (DDD). Domain events are useful for integrating multiple microservices or bounded contexts by providing a messaging-style communication channel. For example, developers asked questions like *(How to fund transfer between bank accounts DDD style with EventStore?- SO post #54967846)*, *(CQRS + Microservices Handling event rollback- SO post #46190467)* and *(Choreography Sagas in DDD - Chain of Integration Events?- SO post #63597403)*.

Domain model, design patterns, and layers in DDD (T3) The code "T3" is assigned to developers' topics that focus on the domain model, design patterns, and different layers such as the user interface, application layer, domain layer, and infrastructure layer in DDD. Some example developers' posts of T3 are *(DDD - How to reuse code in application layer?- SO post #65114332)*, *(Placing entity framework models in the infrastructure layer in onion architecture- SO post #65474540)* and *(Could REST API be considered a presentation layer in DDD?- SESE post #318186)*.

DevOps automation tools (T4) The code "T4" relates to developers discussing technical and configuration matters around DevOps automation tools. Analyzing these discussions, Docker and Kubernetes are often mentioned as applications that aid in automating software development for DevOps teams. Examples include questions like: *(How to run docker containers on different machines- SO post #34938674)* and *(RancherOS + K8s On a single physical machine with multiple nodes- SO post #58235277)*.

Project, team, and time management (T5) The code "T5" refers to developers' discussions about project management, teamwork, and time management. Common issues on SO and SE platforms include work in progress (WIP), such as *(Team consistently exceeding their working in progress (WIP) limit- SEPM post #14049)*, setting development standards *(e.g. Sending out a 'Request for Comment' when establishing a new guideline- SESE post #56753)* and adopting new technologies in organizations *(e.g. Approach to encouraging organizational adoption of new web dev tools- SESE post #279820)* .

Team role and responsibilities in Scrum (T6) The code "T6" pertains to developers' discussions about the roles and responsibilities of Scrum team members, such as the Scrum Master, Product Owner, and development team. Analyzing topics on SO and SE platforms, it is evident that there is confusion regarding the relationship between the Scrum Master and Product Owner, as seen in questions: *(e.g. In Scrum, is a Scrum Master position higher than a Product Owner?- SEPM post #15731)* and *(Can Product Owner + Scrum Master*

(Team/Kanban Lead) provide combined leadership for a team?- SEPM post #17287). Other posts explore the Product Owner's role, for example, *(Product Owner role in Scrum- SEPM post #11613)*, or address development teams in different locations, such as *(A Scrum Master is working with a development team that has members in different physical locations- SEPM post #26223)*.

Project managers' responsibility and contract management (T7) The code "T7" refers to developers' discussions about the responsibilities of management roles and the significance of contracts in project assurance. Analyzing topics on SO and SE platforms, it is clear that many practitioners talk about project managers' duties in posts such as *(Agile management roles- SEPM post #6769)* and *(Financial indicators of a project and a project manager's efficiency- SEPM post #30733)*. Additionally, developers often discuss contract management in posts *(e.g. How can a PM manage the contract aspects for Scrum projects with outcome-based pricing? SEPM post #29636)* and *(How is a software development contract concluded? SEPM post #30284)*.

Entities, Value Objects, and aggregates in DDD (T8) In DDD, entities have a unique identity and combine data and behavior, like users or products. Value objects describe characteristics without a unique identity. Aggregates are collections of related entities and value objects, simplifying the management of complex DDD systems. Each aggregate has an entity, known as the aggregate root, which controls access rights for objects outside the aggregate. The code "T8" is given to developers' posts on SO and SE platforms that discuss aggregates and related concepts. During the analysis, it is noted that developers frequently ask about concepts *(e.g. DDD, Aggregate roots and Entities- SO post #46661591)* and practices such as *(DDD: is it correct for a root aggregate to hold a reference to another root aggregate?- SESE post #328571)*.

Tools and plugins in agile software development (T9) The code "T9" is given to posts focusing on commonly used tools and plugins in agile software development approaches. For instance, developers ask about suitable IDEs *(e.g. Javascript IDE for agile development- SO post #7090013)*, PHP open-source tools such as *(PHP Open Source Tools for Agile Development- SESE post #86883)*, or cross-platform project management tools like *(Any agile free cross-platform project management/ALM tools with Mylyn integration out there?- SO post #3863050)*.

Software development methodology concepts (T10) The code "T10" is given to developers' discussions on SO and SE platforms that focus on concepts of software development methodologies, including Extreme Programming, Waterfall, and Scrum. For example, developers ask questions like *(What's the relationship between SDLC and methodologies like XP, RAD, Scrum, etc.?- SEPM post #15378)* and *(Software development methodology difference- SO post #6147478)*.

Meetings in agile team (T11) The code "T11" is given to posts focusing on common meetings agile teams hold during product development. These meetings include but are not limited to, daily stand-ups *(e.g. Should we cancel the daily stand-Up if we have another meeting during the day?- SEPM post #20796)* and *(Can a daily scrum meeting be replaced by a status email?- SESE post #210111)*, sprint reviews, and sprint retrospectives *(e.g. What's the ideal length of the sprint review/retrospective based on the length of the iteration?- SESE post #71926)*.

Task management in Kanban board (T12) Kanban board offers a clearer view of project workflow by visually displaying the work of all team members in software development. As a result, we assigned the code "T12" to practitioners' discussion topics that address problems or issues related to Kanban Board. Developers need to efficiently organize and manage tasks on the Kanban board, as seen in questions like *(Why in JIRA my field Resolution is labeled as*

Unresolved when the status is Resolved?- SEPM post #22987) and *(Why are items for other users on the TFS current iteration task board grey?- SO post #18450086)*.

Software testing (T13) Software testing helps detect bugs or errors early and offers solutions before delivery, improving software application quality and increasing user satisfaction. Incorporating software testing in development approaches ensures security, reliability, and high performance. We assigned the code "T13" to developers' posts focusing on software testing-related issues, including unit testing, integration testing, acceptance testing, and test-driven development (TDD). Examples of developer discussions include: *(Unit testing - Practicality: Should it pass or fail all or most of the time?- SO post #33205413)* and *(test-driven development - Who should write the tests?- SESE post #35610)*.

Story estimation in Scrum sprint (T14) Story estimation helps the Product Owner gauge the effort required for each user story to complete the software project on time and within budget. Each team member provides their perspective on the work during story estimation. We assigned the code "T14" to practitioners' discussion posts on selected Q&A websites that address issues or problems related to story estimation. Common story estimation questions include when to estimate, such as *(Should a scrum team estimate time for the user stories during Sprint Planning, or before it?- SESE post #270689)* and *(When is it time to do poker planning - during Story Time or during Sprint Planning?- SEPM post #15545)*, and how to use story points for estimation, e.g. *(Measuring scale for story points in Scrum framework-SO post #9017428)*.

Software design and requirements (T15) Detailed, clear, and accurate software requirements help designers, developers, and testers better understand product functionalities and develop software applications that lead to higher user satisfaction. Improved software requirements elicitation, specification, and validation can also streamline the design phase, resulting in higher-quality software applications. We assigned the code "T15" to developers' discussion posts on SO and SE platforms where practitioners discuss challenges related to requirements, design, or both. Numerous posts cover software design and requirements, such as *(Requirements with units in Software Design Document- SESE post #332967)*. Unified Modeling Language (UML) is also frequently mentioned in the design and requirement analysis phases, e.g. *(Include Use Case Diagram (UML)- SO post #31759144)*.

Our analysis of software development approaches related topics on selected Q&A repositories provides valuable insights into the most frequently discussed areas. The identified key discussion topics bear several interesting implications. With the highest frequency topics being "T5: Project, team and time management", "T3: Domain model, design patterns and layers in DDD", and "T15: Software design and requirements", these areas stand out as notable focal points of interest in software development. The development of more effective project management methodologies could improve team coordination and productivity (Itzik and Roy 2023; Berntzen et al. 2022) Recognizing these topics can significantly benefit the Q&A repositories. For instance, understanding the most discussed topics allows Q&A platforms to prioritize and highlight content that addresses these areas. If "T5: Project, team and time management" is a frequently discussed topic, repositories can prioritize content around this theme, ensuring that users get immediate access to the most relevant information.

Similarly, standard frameworks for implementing Domain-Driven Design could simplify complex processes, reducing development time and enhancing software quality (Berntzen et al. 2022). Further, these findings can help steer the development of new tools and technologies. With an understanding of what is being most discussed, tool developers can cater more specifically to the needs of software developers. For example, the frequency of discussions around "T4: DevOps automation tools" and "T9: Tools and plugins in agile software development" suggest these are areas where new or improved tools could be beneficial. The

discussions highlighted can also guide educational efforts by shaping the curricula of training programs and university courses to better prepare future practitioners. By focusing on these identified areas, education providers can ensure they are equipping learners with the most relevant knowledge and skills (Kuhrmann et al. 2019).

The overall findings indicate that an intensified focus on these topics could drive advancements in understanding, strategy, and methodology, leading to more effective software development. Consequently, these insights not only shape the focus of future software development research but also guide the development of practices and tools in the industry. In summary, these findings offer guidance for the software development field, directing attention to key areas where greater focus could yield substantial improvements in practice, tools, and education.

Key Findings of RQ2

Finding 3: The top three highly discussed topics are "T5: Project, team, and time management," "T3: Domain model, design patterns, and layers in DDD," and "T15: Software design and requirement", emphasizing the importance of these aspects in addressing developers' concerns and challenges. **Finding 4**: Practitioners often discuss CI/CD, DevOps tools, and Agile methodologies like Scrum, Kanban, and Extreme Programming. This suggests that developers actively engage with these methods and need guidance to effectively incorporate them into their workflows. **Finding 5**: Other notable topics include software testing, story estimation in Scrum, and Kanban task management- highlighting the demand for effective strategies to ensure quality, precise estimates, and improved task handling in software projects.

5.3 RQ2.1: Popular and Difficult Topics

We further analyzed the topics discussed in Section 5.2 (RQ2) to identify the most popular and difficult topics to address. As depicted in Fig. 5, we utilized the min-max normalization technique (see Section 4.1, RQ2.1 (method)) to calibrate the values of both popularity and difficulty metrics. Comprehensive statistical measures for these metrics are made accessible in the data replication package (Khan et al. 2022). The analysis led us to identify the following popular and challenging topics:

Topic popularity The top three most popular topics on the SO and SE platforms have been identified using min-max normalization. These topics are Software development methodology concepts (T10), Software design and requirements (T15), and Software testing (T13) (See Fig. 5). Although T10 is the most popular topic, it ranks 11th in the frequency distribution of all topics, as depicted in Fig. 4. We used Kendall's Tau correlation test (Kendall 1970) to evaluate the relationship between topic popularity and frequency distribution. The resulting Tau value is 0.028, with a probability of tau = 0 (no association) and a p-value of 0.923. This suggests no significant correlation exists between the topic's popularity (Fig. 5) and its frequency distribution (Fig. 4).

It demonstrates that the frequency of a topic does not accurately represent its popularity- the number of questions on a topic cannot fully capture user activity on a Q&A website. By understanding these findings, software practitioners can gain insight into the prevailing topics in the field of software development approaches. This information can prove valuable when selecting the most suitable software development approach.

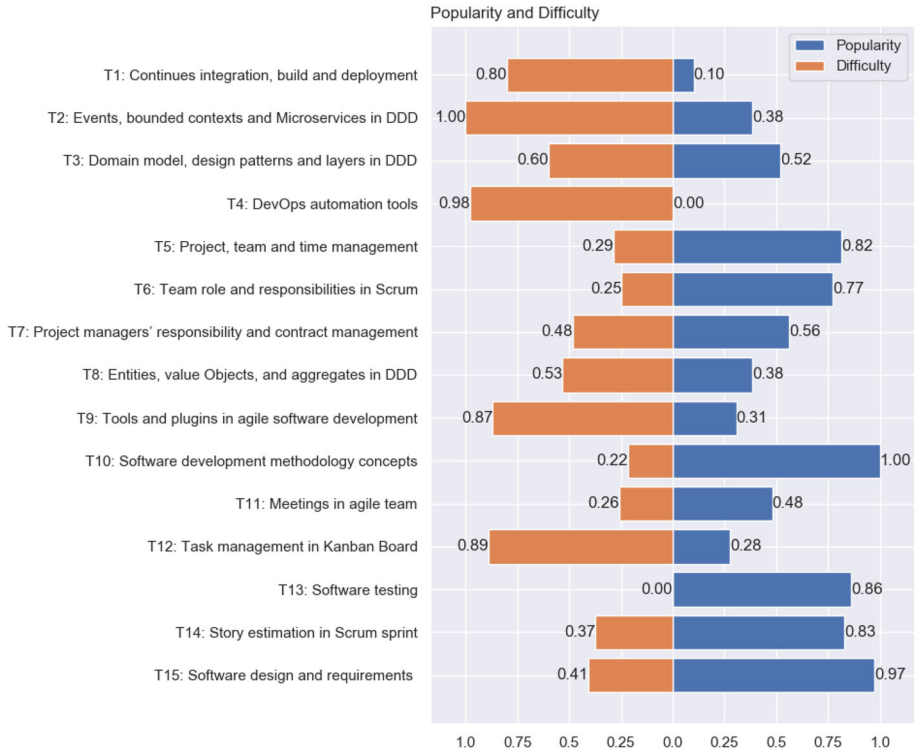

Fig. 5 Popularity and difficulty of topics

Topic difficulty In the dataset, the top three most difficult topics for developers were identified as Events, bounded contexts, and Microservices in DDD (T2), DevOps automation tools (T4), and Task management in Kanban Board (T12) (see Fig. 5). From Fig. 7, it is evident that the number of posts on any of the top three most difficult topics exhibited an upward trend. For instance, for DevOps automation tools (T4), the developers' questions increased significantly (See Fig. 7). This information is valuable for software practitioners to continuously monitor developers' posts on Q&A websites and provide timely solutions or remedies for questions regarding emerging trends. In contrast, the number of questions for the top three most popular developers' topics did not increase, with a downward trend visible in Fig. 6. Using Kendall's Tau correlation test (Kendall 1970), we discovered a strong correlation (tau = -0.600, p-value = 0.001) between topic popularity and difficulty (popularity decreased as difficulty increased).

The analysis from RQ2.1, which highlights the most popular topics in software development, carries profound implications for the trajectory of the software industry. The prominence of topics such as Software development methodology concepts (T10), Software design and requirements (T15), and Software testing (T13) indicates a collective emphasis on refining methodologies, enhancing software design practices, and ensuring robust testing procedures. Given this trend, we can anticipate an increased investment in methodology research, potentially leading to the emergence of new methodologies or the refinement of existing ones. This focus suggests that the industry is gearing towards a more structured and

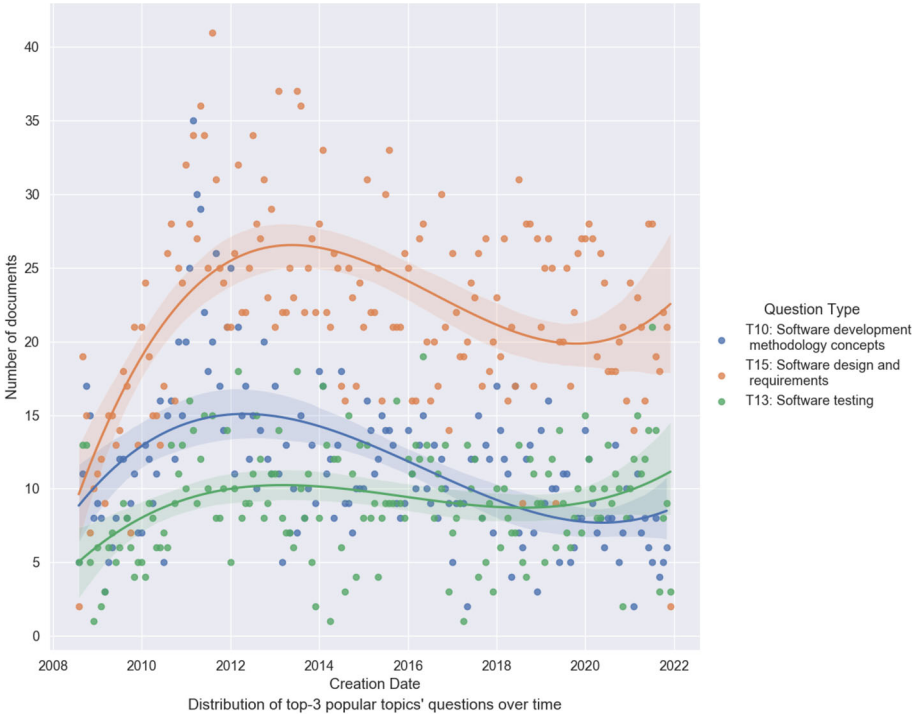

Fig. 6 Distribution of top-3 popular topics' questions over time

systematic approach to software development, aiming for higher quality outputs and more efficient processes.

Concurrently, the recognition of challenging topics such as Events, bounded contexts, and Microservices in DDD (T2), DevOps automation tools (T4), and Task management in Kanban Board (T12) highlights the complexities inherent in these areas. The increasing discussions around these subjects signify a pressing demand for specialized knowledge and solutions. Especially in emerging domains like DevOps automation tools (T4), there is a clear indication of the industry's evolving needs. For software professionals, this presents a strategic avenue to specialize and offer expertise, potentially positioning themselves as thought leaders or solution providers for these intricate topics, thereby contributing to the advancement of the field.

Furthermore, the observed correlation between a topic's popularity and its perceived difficulty reveals an interesting dynamic. As topics become more complex, their popularity seems to decrease, suggesting a potential knowledge gap. This emphasizes the need for industry experts to focus on these challenging areas, offering solutions and insights. By addressing these topics, there is potential for significant advancements in software quality and development practices. In summary, RQ2.1 findings serve as a guide, highlighting areas in software development that require attention, innovation, and growth.

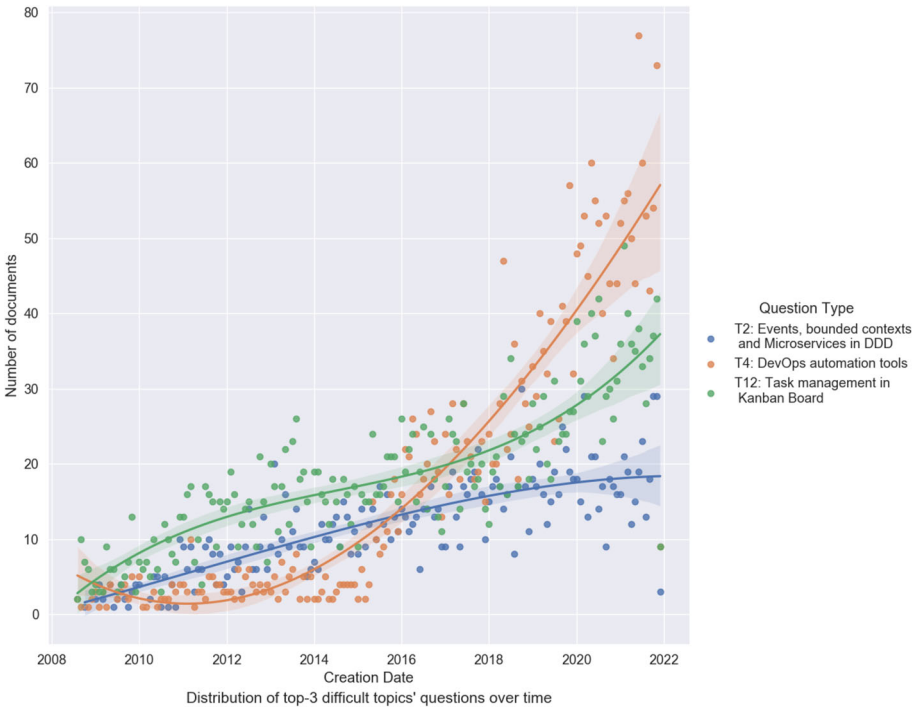

Fig. 7 Distribution of top-3 difficult topics' questions over time

Key Findings of RQ2.1

Finding 6: The top three most popular topics are Software development methodology concepts (T10), Software design and requirements (T15), and Software testing (T13). No strong correlation exists between frequency distribution (Fig. 4) and popularity (Fig. 5), indicating that question count doesn't fully represent user activity on Q&A websites. **Finding 7**: The top three most difficult topics for developers are Events, bounded contexts, and Microservices in DDD (T2), DevOps automation tools (T4), and Task management in Kanban Board (T12). A strong negative correlation between popularity and difficulty indicates that difficult topics are less popular. **Finding 8**: Many questions on difficult topics go unanswered, indicating practitioners should focus on addressing them. Providing solutions for these areas can enhance software quality and progress development approaches.

5.4 RQ3: Software Development Approaches Challenges

Through an exhaustive qualitative analysis, we identified 49 particular challenges that practitioners face in software development methodologies (for detailed methodology, see Section 4.1, RQ3). These identified challenges have been classified and mapped into 14 sub-themes and 4 high-level themes (categories), as demonstrated in Fig. 8.

The summary of the high-level themes (categories) is as follows:

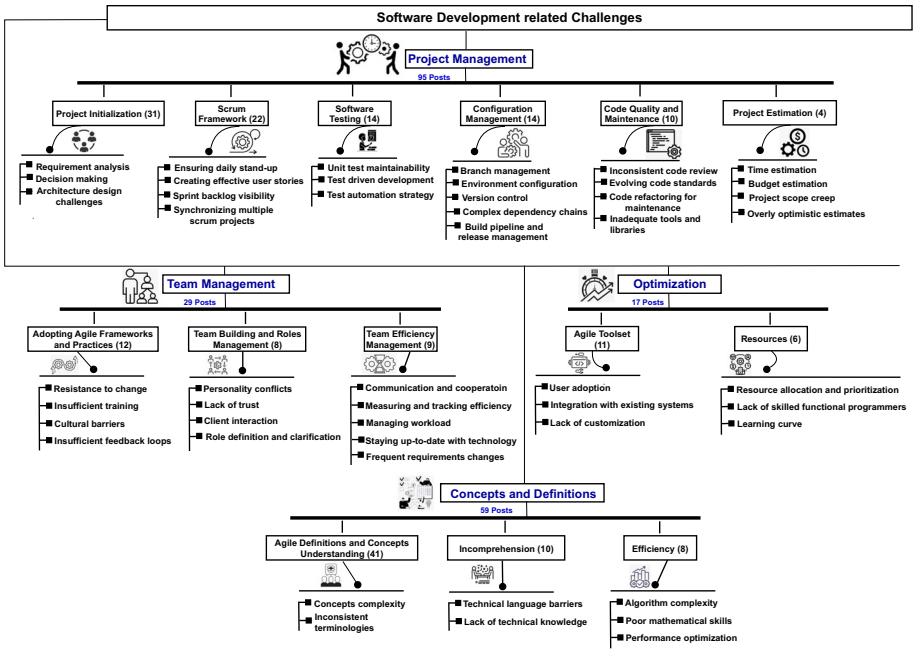

Fig. 8 Mapping of identified challenges

5.4.1 Project Management Challenges

We analyzed developers' discussions on selected Q&A platforms, focusing on the project management challenges they encounter when adopting software development approaches and practices. This category comprises the most significant portion of the dataset, including 95 out of 200 top-ranked developers' posts. These posts highlight practitioners' questions for effective and efficient solutions in software project management. Examining the data from these 95 posts, we identified 22 project management relevant challenges classified across six sub-themes (See Fig. 8).

In project management, practitioners face challenges related to project initiation activities such as requirement analysis, decision-making, and architecture design. For example, *(How can I get things right at the beginning of a software project?- SESE post #324082)*. These initial steps provide a fundamental structure; if they go wrong, they can lead to project failure. Software practitioners should provide guidelines with examples to help in initiating and executing project activities more efficiently.

A common sub-theme of developers' questions in this category relates to challenges when adopting the Scrum framework, such as daily stand-up, user stories, sprint backlog, and project synchronization. For example, *(Developers wonder if bug-fixing tasks should be assigned story points in Scrum- SESE post #162145)*. Similarly, developers often seek technical solutions for software testing (sub-theme), such as unit testing and Test-Driven Development *(Is unit testing or test-driven development worthwhile?- SESE post #140156)*, on Q&A platforms.

We also identified configuration management (sub-theme) challenges, e.g., branch management, environment configuration, version control, complex dependency change, build

pipelines, and release management (See Fig. 8). For instance, *(How do you handle integrating code from multiple branches/developers each sprint?- SESE post #372716)*. Furthermore, developers' discussions revealed challenges relevant to code quality and maintenance such as *(I've inherited 200K lines of spaghetti code - what now? SESE post #155488)* and project estimation *(What can I do to get better at estimating how long projects are going to take?- SESE post #39411)*.

In summary, practitioners face a variety of project management challenges when adopting different approaches and practices for software development. The main areas of concern include project initialization, Scrum framework adoption, software testing, configuration management, code quality, and project estimation. These insights are vital for software practitioners to improve their methodologies and better support development activities. Furthermore, researchers can use these findings to develop novel software development approaches that address current trends and challenges faced by practitioners in the field.

5.4.2 Team Management Challenges

This category includes developers' posts from the selected Q&A platforms that discuss challenges to effectively managing a team in software development approaches. We identified a total of 13 primary challenges, which were subsequently grouped into 3 subcategories (See Fig. 8).

Upon careful examination of developers' posts, we noticed that development teams encounter various challenges when adopting agile frameworks and implementing their practices. For instance, *(How can we make Agile enjoyable for developers that like to personally, independently own large chunks from start to finish- SESE post #80751)*, specifically when shifting the team from Waterfall to Agile.

Developers also encountered challenges relevant to team building and roles management, such as *(Why can't the Scrum Master and the project manager be the same person?- SEPM post #4707)*. Another example of developers' posts in this category is seeking approaches to improve team efficiency, e.g. *(How can we reduce downtime at the end of an iteration? SESE post #66708)*.

We further noticed five core challenges related to team efficiency management (See Fig. 8), for example, *(How to measure software development performance?- SO post #1168131)*, *(How to tell whether your programmers are under-performing?- SESE post #177167)* and *(Product owner and/or scrum master in performance review of developers- SESE post #439092)*.

It can be concluded that developers face several challenges when managing teams in software development approaches, particularly when transitioning from one methodology to another, such as from Waterfall to Agile. Key concerns include making Agile enjoyable for developers who prefer to independently own large project portions, managing and defining team roles, and improving team efficiency. Addressing these challenges is crucial for organizations to ensure smooth transitions between development methodologies and to enhance overall team productivity and effectiveness.

5.4.3 Optimization

In this category, developers' discussion posts related to challenges focusing on agile toolset and resources. Various posts focuses on Agile tools choice, comparison, and usage, with examples including Azure Web App *(Azure Web App (ASP.NET MVC) becomes cold every*

ten minutes and takes +10-20s to load- SO post #50115939), Azure DevOps Server *(backup of Azure DevOps repositories - SO post #62174938)*, Kanban Board *(Where to Find a Desktop Kanban board application?- SEPM post #829)*, Jenkins *(Is it a good idea to make Ansible and Rundeck work together, or using either one is enough?- SO post #31152102)*, Visual Studio *(What is Security Development Lifecycle Checks option in Visual Studio?- SO post #18304632)*, and Virtual Machine *(Thoughts on Development using Virtual Machines- SESE post #103501)*. Additionally, developers also discussed resources-related problems, for example, lack of skilled functional programmers *(How much functional programming expertise can programmers be expected to have? SESE post- #209071)* and *(Is there a software-engineering methodology for functional programming? SO post- #4852251)*, inadequate tools and libraries *(Where do I find some good examples for DDD?- SO post #540130)* that can lead to suboptimal project outcomes and hinder overall progress in software development (See Fig. 8).

These findings are encouraging for both software developers and the research community. Software developers can provide best practices and post-mortem reports on software methodologies to help in understanding the pros and cons of development approaches. Similarly, software researchers can seize this opportunity to develop research studies comparing various software approaches (traditional and Agile) in developing software applications, particularly considering market-based software products.

5.4.4 Concepts and Definitions

In this category, developers' discussion posts focus on asking general questions in Q&A platforms to gather community knowledge on the concepts and definitions of software development approaches and practices. We identified 7 relevant challenges, which were further categorized across three sub-themes (See Fig. 8).

One common sub-theme in the selected posts involves the agile definition and concepts understanding. For example, practitioners sought information on Domain-Driven Design *(Value objects in DDD - Why immutable?- SO post #4581579)*, Scrum *(What is the difference between Sprint and Iteration in Scrum and length of each Sprint?- SO post #1227318)*, Test-Driven Development (TDD) *(Why is agile all about the test-driven development (TDD) and not development-driven test?- SESE post #326485)*, Continuous Integration (CI) *(What is the purpose of a dedicated "Build Server"?- SO post #1099133)*, Scaled Agile Framework (SAFe), Nexus and Large Scale Scrum (LeSS) *(Scaled scrum/agile frameworks (SAFe vs. Nexus vs. LeSS) comparison - SEPM post #17441)*.

Incomprehension is also a sub-theme in this category; it revolves around issues in technical language barriers and lack of technical knowledge, such as *(How to sell Agile development to (waterfall) clients?- SESE post #215562)* and *(Getting non-programmers to understand the development process- SESE post #4)* .

Efficiency is the third sub-themes of the concept and definition category, where practitioners seek clarification on how algorithms and mathematics efficiently apply to software development approaches, e.g., *(What does mathematics have to do with programming?- SESE post #136987)* and *(Is big-O really that relevant when working in industry?- SESE post #20832)*. Lastly, several posts in this category aim to optimize the performance of software development approaches, e.g., *(Why can't the IT industry deliver large, faultless projects quickly as in other industries?-SESE post #158640)*.

The findings from RQ3 underscore the significance of the preliminary phases in software development. Challenges faced during the inception of projects, particularly in requirement analysis and decision-making, signal the necessity for an enhanced foundational framework.

It is imperative for organizations to allocate resources towards specialized training modules that emphasize these initial stages. Ensuring a robust foundation can substantially mitigate potential issues in subsequent development phases.

The challenges associated with the Scrum framework reveal a potential gap between academic understanding and its pragmatic implementation. While agile methodologies like Scrum are widely taught, but applying it in real life can be different. Mentorship initiatives, where seasoned Scrum professionals guide novices, could offer invaluable insights that conventional training might overlook. Additionally, the issues pertaining to team management and software testing highlight the importance of the availability of a holistic approach to software development, emphasizing the significance of team dynamics and efficient testing methodologies.

In summation, the ongoing discourse on foundational software development approaches indicates a notable knowledge disparity. This presents not only a challenge but also an opportunity for the tech industry and academia to collaboratively organise strategies to bridge this gap. The volume of these discussions indicates a pronounced desire for knowledge among developers. Addressing these foundational concerns can pave the way for a more informed and innovative software development future. In conclusion, RQ3 insights contribute to understanding of the fundamental challenges and opportunities, serving as a strategic guide for future endeavors in software development.

Key Findings of RQ3

Finding 9: Practitioners face challenges in various software development approaches, which are grouped into four high-level themes: Project Management, Team Management, Optimization, and Concepts and Definitions. **Finding 10**: Most challenges in software development are related to project management (See Fig. 8). Ineffective project management can lead to delays, budget overruns, and poor-quality software. Thus, it's vital for practitioners to identify and overcome project management challenges for successful software development project completion.

6 Implications

We now summarised the research and industrial implications of the study findings as follows:

6.1 Research Implications

Examining practitioners' discussions on popular Q&A sites holds significant value for researchers in enhancing software development approaches and application quality (Rosen and Shihab 2016). The trends observed in this study suggest that researchers should devote more attention to the software development approaches domain by providing comprehensive documentation, case studies, and software post-mortem reports. The strong correlation between topic popularity and difficulty highlights the need to focus on and invest effort into various software development process activities that are relatively difficult to understand and execute. Moreover, the thematic mapping of challenges in software development approaches offers a valuable framework for researchers to explore key issues faced by practitioners. By focusing on these challenges, researchers can design targeted studies to understand the

underlying causes and develop effective solutions (tools, frameworks, or techniques) tailored to the specific needs of practitioners.

6.2 Industrial Implications

The research findings on developers' discussion topics on Q&A sites can be employed by software practitioners to understand the current state and trends in software development approaches and the difficulties or challenges developers encounter when adopting software methodologies for application development. Practitioners could examine the significant challenges identified in this study to enhance their development approaches and practices. Furthermore, the research study emphasizes the importance of practitioners attaching accurate and appropriate tags to posted questions to facilitate effectively identifying challenges related to software development approaches. Several instances of posts inaccurately tagged with "development approaches" but asking specific technical questions were found, e.g., *(What is the opposite of initialize (or init)?- SESE post #163004)*. It is crucial to understand the definition of software development approaches. Therefore, future research could consider automated assistance to help practitioners correctly tag posted questions. Moreover, recognizing and mapping the challenges associated with different software development approaches can help practitioners make more informed decisions when selecting methodologies that best suit their projects, team dynamics, and organizational structure. This can lead to improved project outcomes and reduced risks of project failure.

7 Threats to Validity

The validity of this study may be affected by a range of potential threats. We have analyzed the possible threats in terms of the four fundamental types of validity threats, namely internal validity, external validity, construct validity, and conclusion validity, as outlined by Wohlin et al. (2012).

7.1 Internal validity

Internal validity refers to the degree to which certain factors influence the results and analysis of the extracted data. In the context of this study, potential threats to internal validity could occur during various phases, including:

Data collection limitations A potential threat to internal validity pertains to the data collection for the proposed approach. We gathered data on software development approaches from SO, SESE, and SEPM using tags assigned by software practitioners on these platforms. Our tag-based method required us to consider all related tags to enhance the proposed approach's performance. However, we limited our search to general tags such as "development approaches", "agile", "software development life cycle", and some commonly used software development framework names like "Scrum", "DevOps", and "Kanban". This approach may have missed some related practitioners' posts on software development approaches, although it effectively collected relevant developers' questions and excluded false positives. Guaranteeing 100% relevance of over 13k collected posts without comprehensive manual validation is challenging. To minimize this threat, we excluded practitioners' posts containing code snippets and conducted analyses on the remaining data.

Topic modeling subjectivity Topic modeling with LDA has proven effective in analyzing large amounts of textual data. However, the method requires subjectivity in assigning labels to practitioners' topics based on individual understanding, which could lead to misinterpretation risks. To mitigate this threat, we manually reviewed 15 developers' posts with the highest relevance to each topic and cross-checked them with the paper's first three authors.

Temporal Bias in Q&A Analysis The dynamic nature of question frequency in Q&A repositories like SO presents a challenge to our study's internal validity. As topics are matured and common queries answered, the repetitive question tends to decline which could potentially skew our assessment of topic popularity and difficulty. Furthermore, this phenomenon, along with platform-specific moderation practices, could limit the applicability of our findings beyond the specific time frame and platform of our study. Despite this limitation, it provides a meaningful direction for future work. In further studies, a longitudinal analysis (Menard 2002) can be conducted to track topic evolution over time and across various platforms, providing a more comprehensive view of topic popularity and difficulty, thus increasing the study's reliability. For now, we acknowledge this limitation and interpret our results with it in mind, ensuring that our conclusions take into account the potential biases introduced by these factors.

7.2 External Validity

External validity refers to the degree to which the findings of a study can be generalized to other populations, settings, or conditions beyond the specific sample or context of the study. The generalizability of the proposed results may pose a threat to external validity. We performed a qualitative analysis of 200 highly ranked posts to complement the results from topic modeling with more in-depth insights. However, the findings of this qualitative analysis were based on a small sample of posts and may not be entirely generalizable. To address this limitation, we utilized AMS (Bajaj et al., 2014) to select developers' posts as possible representatives of practitioners' discussions on Q&A platforms.

7.3 Construct Validity

Construct validity refers to the degree to which the measures used in a study accurately measure the intended constructs or concepts of interest. In this study, a potential threat to construct validity is the operationalization of software development approaches based on tags and developers' posts. There might be discrepancies between the actual software development practices and the tags or posts analyzed in this study. To address this threat, we gathered data from multiple platforms (SO, SESE, and SEPM) and utilized a combination of topic modeling and qualitative analysis to provide a more comprehensive understanding of software development approaches.

Operationalization of Terms The operationalization of terms *populaity* and *difficulty* may pose a threat to construct validity due to their abstract nature and potential measurement limitations. Popularity, gauged by views, scores, favorites, and comments, may not fully reflect the nuances like sentiment or depth of engagement. Similarly, difficulty, assessed by the acceptance rate, response time, and answer-to-view ratio, might inaccurately represent a question's complexity, which can be influenced by factors like clarity or niche relevance. These perceptions can also evolve over time with changes in the field. However, we have followed the well-defined measurement strategies (Le et al. 2021; Rosen and Shihab 2016;

Ahmed and Bagherzadeh 2018) in the existing relevant literature studies (Peruma et al. 2022; Bagherzadeh and Khatchadourian 2019; Abdellatif et al. 2020; Zahedi et al. 2020) to measure the difficulty and popularity of the identified software development approaches related topics. We acknowledge the potential incompleteness of our measures. However, by adhering to these previously validated measurement strategies, we have ensured a robust and informed approach to our study. We further propose that future studies could enrich the measurment by incorporating a broader set of indicators, such as sentiment analysis, topic novelty, and longitudinal changes.

Subjectivity in Tags Selection In this study, we employed a systematic approach to identify relevant tags. However, given the ever-evolving nature of software development, there's a possibility that our tags might not encapsulate emerging terminologies or practices. By specifically excluding posts with code snippets and limiting our research to three major Q&A platforms, we might be missing out on a broader spectrum of discussions and insights. However, to ensure the integrity and relevance of our research, each tag was carefully reviewed and validated by multiple authors, ensuring a comprehensive and collaborative approach. This collaborative validation ensured that the tags truly resonated with the intended software development concepts. Additionally, each tag's description was meticulously scrutinized to ensure alignment with our research objectives, with deviations promptly excluded. Furthermore, by drawing from authoritative sources and relevant literature, we ensured that the tags selected were both contemporary and grounded in established software development paradigms. This multi-layered, collaborative, and rigorous approach significantly reduced the chances of false positives and negatives, enhancing the construct validity of our findings.

7.4 Conclusion Validity

Conclusion Validity pertains to the level of credibility or reasonableness of the study's conclusions. In this study, the conclusions may rely on the subjective interpretation and knowledge of a sole author, which could result in unaddressed disagreements or discrepancies between co-authors. To minimize this possibility, the first author conducted the gathering and analysis of study data. The rest of the authors thoroughly examined the data during several meetings. Any disparities or conflicts in data analysis were resolved through transparent discussions and cooperative efforts among all authors. Additionally, the study authors conducted several brainstorming sessions to arrive at the final conclusions.

8 Conclusions and Future Work

In our investigation of 13,903 software development posts across SO, SESE, and SEPM forums, we have provided a comprehensive understanding of contemporary practices and trends in the domain. We have employed both topic modeling and qualitative analysis of the collected data, setting our work apart from past research that has primarily examined software development in a narrower capacity. Our findings have shown a steady rise in the quantity of queries related to software development approaches, indicating a heightened interest in this field. However, there has been a decline in the successful response rate to these queries since 2014, suggesting that software development approaches are becoming more complex and challenging for developers.

Using the LDA algorithm, we have identified 15 key discussion topics related to software development approaches, which include more general popular topics such as (e.g., Software

development methodology concepts (T10)) and difficult (e.g., Events, bounded contexts and Microservices in DDD (T2)). Interestingly, the top three most difficult topics all demonstrated an upward trend over time, suggesting they are areas of ongoing struggle for many practitioners. Moreover, we uncovered a negative correlation between topic popularity and difficulty, as per Kendall's Tau correlation test (Kendall 1970). It implies that popular topics likely reflect the areas of interest and concern for practitioners and may represent the field's most critical concepts and practices. These findings are significant regarding difficult topics, as they highlight the need for greater attention and focus on improving practitioners' understanding and ability to apply them effectively.

Additionally, we have identified and categorized 49 challenges faced by practitioners when using software development approaches. These challenges are mapped across 14 sub-themes and 4 high-level themes, which enhance practitioners' understanding of the issues in detail and help them better prepare for overcoming them while implementing software development approaches.

Looking ahead, our intention is to delve deeper into these challenges. We aim to further explore the additional challenging factors, pinpoint their root causes, and suggest best practices to tackle them. To accomplish this, our research will expand to include other well-known Q&A repositories such as GitHub. We will also validate our findings and seek additional insights through industrial surveys and interviews with experts in software development approaches.

Acknowledgements The authors acknowledge that they used generative AI tools (ChatGPT) and writing assistant tool (Grammarly) to fix writing issues in this paper. After using this service, the authors extensively reviewed and modified the content and they take full responsibility for the overall content of the paper.

Author Contributions Arif Ali Khan and Peng Zhou designed the study, collected the data from multiple software repositories, and conducted the data analysis. Javed Ali Khan and Muhammad Azeem Akbar interpreted the results and were reviewed by Mahdi Fahmideh. The manuscript was drafted by Arif Ali Khan, and all authors provided critical feedback and revisions. The final version was approved for submission by all authors.

Funding Open Access funding provided by University of Oulu (including Oulu University Hospital).

Data Availibility Statement The replication package based on the collected data and analysis is given in the Notion repository Khan et al. (2022).

Declarations

Conflicts of interest The authors declare that they have no conflict of interest regarding the research presented in this article. The research was conducted in an objective and impartial manner, and the results and conclusions were based solely on the data and analysis presented.

References

Abdellatif A, Costa D, Badran K, Abdalkareem R, Shihab E (2020) Challenges in chatbot development: A study of stack overflow posts. In: Proceedings of the 17th international conference on mining software repositories pp 174–185

Ahmed S, Bagherzadeh M (2018) What do concurrency developers ask about? a large-scale study using stack overflow. In: Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement pp 1–10

Ali Khan J, Liu L, Wen L, Ali R (2020) Conceptualising, extracting and analysing requirements arguments in users' forums: The crowdre-arg framework. Journal of Software: Evolution and Process 32(12):e2309

Al-Saqqa S, Sawalha S, AbdelNabi H (2020) Agile software development: Methodologies and trends. Int J Interactive Mobile Technol 14(11)

Aymerich B, Díaz-Oreiro I, Guzmán JC, López G, Garbanzo D (2018) Software development practices in costa rica: A survey. In: International conference on applied human factors and ergonomics, Springer pp 122–132

Bagherzadeh M, Khatchadourian R (2019) Going big: a large-scale study on what big data developers ask. In Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering pp 432–442

Bajaj K, Pattabiraman K, Mesbah A (2014) Mining questions asked by web developers. In: Proceedings of the 11th Working conference on mining software repositories pp 112–121

Bajec M, Vavpotič D, Krisper M (2007) Practice-driven approach for creating project-specific software development methods. Inf Softw Technol 49(4):345–365

Bano M, Zowghi D, da Rimini F (2017) User satisfaction and system success: an empirical exploration of user involvement in software development. Empir Softw Eng 22:2339–2372

Barua A, Thomas SW (2014) Hassan AE, What are developers talking about? an analysis of topics and trends in stack overflow. Empir Softw Eng 19(3):619–654

Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R et al (2001) Manifesto for agile software development. Snowbird, UT

Berntzen M, Hoda R, Moe NB, Stray V (2022) A taxonomy of inter-team coordination mechanisms in large-scale agile. IEEE Trans Software Eng 49(2):699–718

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3(Jan): 993–1022

Boehm BW (1988) A spiral model of software development and enhancement. Computer 21(5):61–72

Braun V, Clarke V (2006) Using thematic analysis in psychology. Qual Res Psychol 3(2):77–101

Brisson S, Noei E, Lyons K (2020) We are family: analyzing communication in github software repositories and their forks. In: 2020 IEEE 27th International conference on software analysis, evolution and reengineering (SANER), IEEE pp 59–69

Bustard D, Wilkie G, Greer D (2013) The maturation of agile software development principles and practice: Observations on successive industrial studies in 2010 and 2012. In: 2013 20th IEEE international conference and workshops on engineering of computer based systems (ECBS), IEEE, pp 139–146

Chen T-H, Thomas SW, Hassan AE (2016) A survey on the use of topic models when mining software repositories. Empir Softw Eng 21(5):1843–1919

Cohen J (1960) A coefficient of agreement for nominal scales. Educ Psychol Measur 20(1):37–46

Cruzes DS, Dyba T (2011) Recommended steps for thematic synthesis in software engineering. In: 2011 international symposium on empirical software engineering and measurement, IEEE pp 275–284

Dwivedi AK, Tirkey A, Rath SK (2018) Software design pattern mining using classification-based techniques. Front Comp Sci 12(5):908–922

Haque MU, Iwaya LH, Babar MA (2020) Challenges in docker development: A large-scale study using stack overflow. In: Proceedings of the 14th ACM/IEEE International symposium on empirical software engineering and measurement (ESEM) pp 1–11

Hastie T, Tibshirani R, Friedman JH, Friedman JH (2009) The elements of statistical learning: data mining, inference, and prediction, vol 2. Springer

Itzik D, Roy G (2023) Does agile methodology fit all characteristics of software projects? review and analysis. Empir Softw Eng 28(4):1–89

Jacobi C, Van Atteveldt W, Welbers K (2016) Quantitative analysis of large amounts of journalistic texts using topic modelling. Digit Journal 4(1):89–106

Kendall MG (1938) A new measure of rank correlation. Biometrika 30(1/2):81–93

Kendall M (1970) Rank correlation methods 4th edition charles griffin. High Wycombe, Bucks

Khan AA, Khan JA, Akbar MA, Zhou P, Fahmideh M (2022) Data replication package for the paper: Insights into software development approaches: Mining q&a repositories, https://zhoupppp.notion.site/SPRPs-1da3b78e78234e5293becaaba846ef63 accessed: 23 Aug 2022

Khan JA, Liu L, Jia Y, Wen L (2018) Linguistic analysis of crowd requirements: an experimental study, In: 2018 IEEE 7th International workshop on empirical requirements engineering (EmpiRE), IEEE pp 24–31
Khan JA, Xie Y, Liu L, Wen L (2019) Analysis of requirements-related arguments in user forums. In: 2019 IEEE 27th International requirements engineering conference (RE), IEEE pp 63–74
Khan JA, Yasin A, Fatima R, Vasan D, Khan AA, Khan AW (2022) Valuating requirements arguments in the online user's forum for requirements decision-making: The crowdre-varg framework. Software Practice and Experience 52(12):2537–2573
Khan JA, Liu L, Wen L, Ali R (2019) Crowd intelligence in requirements engineering: Current status and future directions. Foundation for software quality, Springer, In International working conference on requirements engineering, pp 245–261
Khan AA, Shameem M, Nadeem M, Akbar MA (2021) Agile trends in chinese global software development industry: Fuzzy ahp based conceptual mapping. Appl Soft Comput 102:107090
Kim G, Humble J, Debois P, Willis J, Forsgren N (2021) The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations, IT Revolution
Klünder J, Hohl P, Fazal-Baqaie M, Krusche S, Küpper S, Linssen O, Prause CR (2017) Helena study: Reasons for combining agile and traditional software development approaches in german companies. In: International conference on product-focused software process improvement, Springer pp 428–434
Kuhrmann M, Tell P, Hebig R, Klünder J, Münch J, Linssen O, Pfahl D, Felderer M, Prause CR, MacDonell SG et al (2021) What makes agile software development agile? IEEE Trans Software Eng 48(9):3523–3539
Kuhrmann M, Diebold P, MacDonell S, Münch J (2017) 2nd workshop on hybrid development approaches in software systems development. In: International conference on product-focused software process improvement, Springer pp 397–403
Kuhrmann M, Diebold P, Münch J, Tell P, Garousi V, Felderer M, Trektere K, McCaffery F, Linssen O, Hanser E, et al. (2017) Hybrid software and system development in practice: waterfall, scrum, and beyond. In: Proceedings of the 2017 international conference on software and system process pp 30–39
Kuhrmann M, Nakatumba-Nabende J, Pfeiffer R-H, Tell P, Klünder J, Conte T, MacDonell SG, Hebig R (2019) Walking through the method zoo: does higher education really meet software industry demands?. In: 2019 IEEE/ACM 41st International conference on software engineering: software engineering education and training (ICSE-SEET), IEEE pp 1–11
Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data, biometrics 159–174
Le THM, Croft R, Hin D, Babar MA (2021) A large-scale study of security vulnerability support on developer q&a websites. In: Evaluation and assessment in software engineering pp 109–118
Menard S (2002) Longitudinal research, vol 76, SAGE Publications
Mushashu ET, Mtebe JS (2019) Investigating software development methodologies and practices in software industry in tanzania, In: 2019 IST-Africa Week Conference (IST-Africa), IEEE pp 1–11
Paixão KV, Felício CZ, Delfim FM, Maia MdA (2017) On the interplay between non-functional requirements and builds on continuous integration. In: 2017 IEEE/ACM 14th International conference on mining software repositories (MSR), IEEE pp 479–482
Peruma A, Simmons S, AlOmar EA, Newman CD, Mkaouer MW, Ouni A (2022) How do i refactor this? an empirical study on refactoring trends and topics in stack overflow. Empir Softw Eng 27(1):11
Pinto G, Castor F, Liu YD (2014) Mining questions about software energy consumption. In: Proceedings of the 11th working conference on mining software repositories, pp 22–31
Pressman RS (2005) Software engineering: a practitioner's approach, Palgrave macmillan
Riaz MN (2019) Implementation of kanban techniques in software development process: An empirical study based on benefits and challenges. Sukkur IBA J Comput Math Sci 3(2):25–36
Röder M, Both A, Hinneburg A (2015) Exploring the space of topic coherence measures, In Proceedings of the eighth ACM international conference on Web search and data mining, pp 399–408
Rosen C, Shihab E (2016) What are mobile developers asking about? a large scale study using stack overflow. Empir Softw Eng 21(3):1192–1223
Royce WW (1987) Managing the development of large software systems: concepts and techniques. In: Proceedings of the 9th international conference on Software Engineering pp 328–338
Storey M-A, Singer L, Cleary B, Figueira Filho F, Zagalsky A (2014) The (r) evolution of social media in software engineering. Future of software engineering proceedings 100–116
Sun X, Li B, Leung H, Li B, Li Y (2015) Msr4sm: Using topic models to effectively mining software repositories for software maintenance tasks. Inf Softw Technol 66:1–12
Tell P, Klünder J, Küpper S, Raffo D, MacDonell S, Münch J, Pfahl D, Linssen O, Kuhrmann M (2021) Towards the statistical construction of hybrid development methods. J Software: Evolution and Process 33(1):e2315
Treude C, Barzilay O, Storey M-A (2011) How do programmers ask and answer questions on the web?(nier track). In: Proceedings of the 33rd international conference on software engineering pp 804–807

Ullah T, Khan JA, Khan ND, Yasin A, Arshad H (2023) Exploring and mining rationale information for low-rating software applications. Soft Computing 1–26
Vidoni M (2022) A systematic process for mining software repositories: Results from a systematic literature review. Inf Softw Technol 144:106791
Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer Science & Business Media
Yang X-L, Lo D, Xia X, Wan Z-Y, Sun J-L (2016) What security questions do developers ask? a large-scale study of stack overflow posts. J Comput Sci Technol 31(5):910–924
Zahedi M, Rajapakse RN, Babar MA (2020) Mining questions asked about continuous software engineering: A case study of stack overflow. In: Proceedings of the evaluation and assessment in software engineering pp 41–50
Zhou C, Li B, Sun X (2020) Improving software bug-specific named entity recognition with deep neural network. J Syst Softw 165:110572
Zhou P, Ali Khan AA, Liang P, Badshah S (2021) System and software processes in practice: Insights from chinese industry. In: Evaluation and assessment in software engineering, pp 394–401

Arif Ali Khan received a Ph.D. degree in software engineering from the City University of Hong Kong, Hong Kong, in 2017. He is currently an Assistant Professor with the M3S Empirical Software Engineering Research Unit, at the University of Oulu, Oulu, Finland. Previously, he served as a faculty member in the Faculty of Information Technology at the University of Jyväskylä, Finland and the College of Computer Science and Technology at Nanjing University of Aeronautics and Astronautics, China. He has participated in and managed several empirical software engineering-related research projects. He has expertise in software process improvement, quantum software engineering, microservices architecture, artificial intelligence (AI) ethics, agile software development, DevOps, global software development, multicriteria decision analysis, soft computing, and evidence-based software engineering. He has published over 110 articles in peer-reviewed software engineering journals and conferences. More information available at: https://www.oulu.fi/en/researchers/arif-ali-khan.

Javed Ali Khan is working as a senior lecturer, at the Foundation of Software Engineering (FSE) group, Department of Computer Science, University of Hertfordshire, UK. Previously, he worked as an Assistant Professor cum chairperson in the Department of Software Engineering, University of Science and Technology Bannu, Pakistan. He completed his PhD in Software Engineering from Tsinghua University (QS ranked 12th), Beijing, PR. China. He regularly publishes papers in reputable software engineering journals and conferences. His areas of interest are Requirements Engineering, CrowdRE, Argumentation and argument mining, mining software repositories, human values in Software, Quantum Software Engineering, Feedback Analysis, Empirical Software Engineering, Sentiment and opinion mining, Requirements Prioritization, Mining fake reviews, sarcasm detection, and Health Analytics.

Muhammad Azeem Akbar received his Ph.D. degree in software engineering from Chongqing University, Chongqing, China, in 2019. He previously worked as a Post-Doctoral Researcher at the Nanjing University of Aeronautics and Astronautics, Nanjing, China, and as a Senior Researcher at Lero (the Irish Software Research Centre), Limerick, Ireland. He is currently an Associate Professor at the Lappeenranta-Lahti University of Technology, Lappeenranta, Finland. His expertise includes quantum software engineering, artificial intelligence (AI) ethics, agile software development, continuous software engineering, and applied soft computing. He has organized multiple software engineering workshops and served as a guest editor for several special issues in major software engineering journals, such as Information and Software Technology, and Journal of Software: Practice and Experience.

Peng Zhou Peng Zhou is a Master's research student at the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research in Empirical Software Engineering focuses on repository mining for software processes. Through his studies, he aims to uncover insights that can enhance software development approaches.

Mahdi Fahmideh received the Ph.D. degree in information systems from the Business School, University of New South Wales, Sydney, NSW, Australia. He is currently a Senior Lecturer (equivalent to an Associate Professor in North America) in cyber security with the School of Business, University of Southern Queensland, Toowoomba, QLD, Australia. His research aims at providing solutions based on ABCD technologies (i.e., technologies of artificial intelligence, blockchain smart contracts, cloud computing, and big data), to help IT-enabled organizations with digital transformation and tackle business problems. His research outputs can be in the form of conceptual models, system development methodologies, and decision-making frameworks. Having worked in the software industry prior to taking up an academic position, he has eight years of first-hand experience as an analyst programmer and consultant in implementing the core backend of software systems for different industry sectors such as publishing, government, and insurance.