Tampere University

MALIHA TASNIM

REQUIREMENTS ENGINEERING IN OPEN-SOURCE SOFTWARE PROJECTS: PROCESSES & CHALLENGES

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
June 2023

ABSTRACT

Maliha Tasnim: Requirements Engineering in Open-Source Software Projects: Processes & Challenges
M.Sc. Thesis
Tampere University
Master's Degree Programme in Computing Science (Software, Web & Cloud)
June 2023

Open-source software (OSS) development has gained significant influence in the software industry, fostering collaboration and knowledge sharing among developers and users. This paper explores the practices and challenges of requirements engineering (RE) in OSS projects through a systematic literature review (SLR). By analyzing 43 selected papers, the study reveals various practices, techniques, and methods that aid RE activities in OSS projects while addressing the challenges faced by practitioners and proposing potential solutions.

The literature review highlights a growing interest in utilizing machine learning and statistical methods to support RE activities in OSS projects. Specifically, these methods focus on automating requirements identification and analysis by leveraging information extracted from project discussion forums, issue reports, and other online resources. Moreover, the review emphasizes the significance of community involvement in OSS projects. Several studies examine the interaction patterns, expertise levels, and influence of developers within these projects. This underscores the importance of fostering a strong and engaged developer community to enhance the quality of requirements and project outcomes.

The findings from the SLR offer valuable insights for both OSS project managers and researchers. They provide guidance on effectively managing requirements in OSS projects, considering the unique challenges and characteristics of the open-source environment. By incorporating the recommended practices and techniques, project managers can streamline the RE process and improve the overall success of OSS projects. Overall, this thesis contributes to the understanding of RE in OSS projects by synthesizing existing knowledge and highlighting emerging trends. The insights gained from this research can inform decision-making and strategy development for both practitioners and researchers in the OSS community. By embracing these findings, stakeholders can leverage the collaborative power of OSS development and optimize the handling of requirements to deliver high-quality software solutions.

Key words and terms: Requirements Engineering, Open Innovation, Open-Source Software, Systematic Literature Review (SLR).

ACKNOWLEDGEMENT

I am immensely grateful to my thesis supervisor, Zheying Zhang, for her invaluable guidance and support. I would also like to extend my heartfelt appreciation to my advisor, Timo Poranen, and my fellow mate, Maruf Rayhan, for their unwavering encouragement and assistance. Their contributions have been instrumental in shaping my thesis journey. I am truly fortunate to have such amazing individuals by my side throughout this thesis endeavor.

Contents

1. Introduction

In recent decades, there has been a notable surge in interest in open innovation, both in academic circles and among businesses. For software-centric companies, the ever-evolving technological landscape and the expanding global competition in software production necessitate the search for new sources of innovation [1]. In the pursuit of maximizing return on investment (ROI), software companies are compelled to explore various avenues of open innovation [2]. These approaches involve the sharing of knowledge and resources, as well as the utilization of external knowledge and resources from diverse entities, spanning from other companies to individual developers.

Open-source is considered as an innovative and particular avenue of open innovation, which is generated within extensive online communities or closed communities [3]. These communities are open to individuals from diverse backgrounds, encompassing both hobbyist users and professionals who rely on the software as a critical component of their business operations. However, operating within such a mixed environment introduces challenges in effectively managing requirements. Tasks like comprehending and prioritizing requirements become notably intricate [3]. Overcoming these challenges is pivotal in harnessing the full benefits of open-source software production, which can encompass improved inter-organizational and external collaboration, reduced support costs, and adherence to vendor compliance [3].

In light of these circumstances, it is evident that open-source software (OSS) has emerged as a progressively expanding open innovation policy embraced by an increasing number of software firms [5]. Due to the characteristics of openness, generally OSS communities consist of a significant quantity of members. This large communities often make the management of RE processes complex and bulky. Open-source software (OSS) projects are characterized by a distributed community of developers, users, and other stakeholders, working in a collaborative and often decentralized manner [5]. Given the less formal organizational structure in OSS projects, the RE process diverges from conventional software development practices. Requirements are typically expressed for open review, elaboration, and discussion through informal channels such as mailing lists, forums, and issue tracking systems [4] [5]. Traditional requirements review activities are replaced by community feedback and testing. Comments and peer reviews are common practices in OSS projects, where developers assess each other's work to identify and resolve potential issues and discrepancies. By continuously building, integrating, and testing change requests and implementations, developers can ensure that requirements are properly implemented and satisfied [6]. This results in more dynamic and informal RE activities. Given these

observations, this thesis aims to explore the requirement engineering practices and challenges in the OSS context through a systematic literature review (SLR). The purpose of this thesis is 1) to identify RE practices to support OSS project managers and contributors in requirements creation, analysis, and management, thus enhancing their effectiveness and efficiency in conducting requirements analysis and implementation in OSS projects; and 2) to recognize challenges and research-based solutions for these RE activities in OSS projects. To achieve these goals, two research questions (RQs) has been formulated.

1.1 Research Questions

Research questions are conducted in such a way which can address the goal of this thesis through the literature review. The research questions and consecutive relevant sub-questions are given as below:

RQ1: What requirement engineering activities, tools and techniques are used by practitioners in open-source software project?

RQ2: What are the challenges faced by the practitioners in OSS projects? What are the possible solutions to overcome the challenges?

The first research question will address the possible requirements engineering activities, associated tools and techniques which are commonly used in open-source projects in the current software industry and identify the roles of the practitioners who are involved in this process in the community. After addressing these facts, this research question will aim to draw a connection on how these RE activities and sub-activities play a crucial role to accelerate the output efficiency of open-source software projects. The follow up second research question will address the possible challenges which the practitioners face in the OSS projects to imply these RE activities. After identifying the challenges, this research question will also point out the possible solutions to overcome those challenges.

1.2 Research Methods

Systematic literature review (SLR) method is used in this thesis to answer the research questions. This method basically ensures a systematic and rigorous process of identifying, selecting, and analyzing relevant literature. It aims to include all relevant studies available on this thesis topic, minimizing the risk of bias and providing a comprehensive overview of existing knowledge. By systematically reviewing and synthesizing existing literature, this method helps to identify gaps in the current knowledge and areas where further research is needed, which is properly identified in this thesis paper. Several steps have been followed to conduct this SLR method for this thesis paper. First of all, proper search

strings are selected to search relevant papers and articles from databases like IEEEXplore, Scopus, ACM Digital Library, and Web of Science. For the initial screening of the search results, inclusion and exclusion criteria are applied. After that, forward and backward snowballing process are implied to get the final selected papers. From those selected papers, required data are extracted using excel sheet. Detailed research methods are discussed in Chapter 5 with associated procedures. It also needs to be mentioned that, this thesis is an extension version of a conference paper which is already submitted**.**

1.3 Thesis Structure

This thesis is conducted by utilizing the systematic literature review (SLR) method. Thesis structure is as following: Chapter 2 will explain about the definition of requirements engineering along with its advantages and processes in software engineering field. Chapter 3 will discuss about the definition of open innovation and open-source software. This chapter will also briefly explain the primary differences of open and closed innovation. Chapter 4 will cover existing literature reviews on OSS development and identifies a gap in the literature and a need for investigating RE practices in OSS projects. Chapter 5 will discuss about the research methodology and how it is executed. Chapter 6 and 7 will continue about the results and discussion. Chapter 8 will proceed with the limitations of the work and conclusion.

2. Requirements Engineering

Requirements engineering (RE) is the process of eliciting, analyzing, documenting, validating, and managing the needs and expectations of stakeholders, ensuring that the software product meets their needs [7] [8] [9]. Establishing a shared understanding of the requirements among stakeholders is essential for successfully developing, maintaining, and evolving a software product, with various techniques and tools to support the communication and cooperative interaction process between requirements analysts and stakeholders [10] [11].

Requirements engineering is an essential aspect of software development and project management. It is the process of identifying, documenting, tracking, and communicating the needs and expectations of stakeholders in a project or product development [12]. The aim of requirements engineering management is basically to ensure that the end product meets the demands of all stakeholders, delivers within budget and schedule, and satisfies the quality standards [13]. By following a systematic approach to requirements engineering management, organizations can ensure that their projects are delivered on time, within budget, and to the satisfaction of all stakeholders [12].

2.1 Roles of Requirements Engineering

The importance of requirements engineering management cannot be overstated, as it helps to minimize the risk of project failure. Requirements that are not well defined, documented, or managed can lead to misunderstandings, delays, and cost overruns [14]. On the other hand, proficient management of requirements engineering can contribute to ensuring timely delivery, adherence to budgetary constraints, and meeting the expectations of stakeholders for a given project [14]. Advantages of requirements engineering can be pointed as below:

1. Requirements engineering ensures that the right product is built, with the right features, to the right standards.
2. Requirements engineering helps reduce rework, avoid misunderstandings, and minimize confusion, leading to increased efficiency.
3. Requirements engineering helps engage stakeholders early and often, ensuring that their needs and expectations are met.
4. Requirements engineering provides a clear trail of how the product has evolved, making it easier to track changes and manage risk.
5. Requirements engineering supports collaboration and communication between development teams, stakeholders, and other relevant parties.
6. Requirements engineering provides visibility into project progress, making it easier to monitor progress and manage resources.

Studies show that, insufficient Requirements Engineering (RE) in software projects can lead to various challenges and negative outcomes [14]. Along with the exploration about the pains associated with inadequate RE, studies also shed light on its consequences. One major pain is the risk of misaligned expectations between stakeholders, resulting in misunderstandings, rework, and delays in project delivery [14]. Inadequate requirements can lead to poor software quality, decreased user satisfaction, and increased maintenance costs. Insufficient RE also hinders effective project planning and resource allocation, making it difficult to meet project goals and deadlines [12]. Additionally, lacking clarity in requirements can lead to difficulties in scope management, leading to scope creep and project scope instability [12]. Basically, the critical importance of robust RE practices are proven to mitigate these pains and ensure successful software project outcomes [14].

2.2. Requirements Engineering Processes

Organizations can ensure that their requirements are well understood, accurately captured, and properly managed throughout the project or product lifecycle by following several significant steps [15]:

Requirements elicitation is a critical phase in the software development lifecycle. It involves identifying, collecting, and documenting the needs, expectations, and constraints of stakeholders [13]. This process relies on various techniques such as interviews, surveys, workshops, and observations to gather information. The aim is to understand the stakeholders' requirements and ensure that all relevant perspectives are considered [13]. Effective requirements gathering requires strong communication and interpersonal skills to elicit accurate and complete requirements. During this process, requirements analysts engage with stakeholders to identify their pain points, goals, and desired outcomes. They also gather information about the business processes, user needs, system constraints, and technical considerations [12]. The gathered requirements are typically documented in a structured manner, such as in requirement specifications or user stories, to ensure clarity and traceability throughout the development process. Requirements gathering sets the foundation for the subsequent phases of analysis, design, and implementation [12].

Requirements analysis is the process of examining and refining the gathered requirements to ensure their clarity, consistency, and feasibility [13]. It involves analyzing the requirements for completeness, correctness, and alignment with the project objectives. The goal is to refine the initial requirements into a more precise and unambiguous set that can serve as a basis for system design and development [13]. During requirements analysis, analysts scrutinize the requirements to identify any conflicts, ambiguities, or gaps. They assess the feasibility of implementing the requirements within the given constraints and evaluate the

potential risks and impacts [12]. Techniques such as use case modeling, data modeling, and scenario-based analysis are commonly employed to validate and verify the requirements. This process aims to ensure that the requirements are well-defined, verifiable, and aligned with the stakeholders' needs. Requirements analysis also involves prioritizing requirements based on their importance and impact, identifying dependencies between requirements, and creating traceability links to ensure coverage and manage changes effectively [12]. The outcome of requirements analysis is a refined and validated set of requirements that can guide the subsequent development activities.

Requirements documentation is the process of capturing and recording the gathered and analyzed requirements in a structured format [15]. It involves transforming the requirements into clear, concise, and unambiguous artifacts that serve as a reference for all project stakeholders. The purpose of documentation is to ensure that the requirements are well-understood, communicated effectively, and can be easily shared, reviewed, and validated [12]. Requirements documentation includes creating documents such as requirement specifications, use case diagrams, data flow diagrams, user interface prototypes, and other visual representations [12]. These artifacts provide a detailed description of the system's functional and non-functional requirements, user interactions, system behavior, and interface specifications. They also document any assumptions, constraints, and dependencies associated with the requirements. Well-documented requirements facilitate effective communication and understanding among stakeholders, provide a basis for system design and development, and serve as a reference for future maintenance and updates. They also support traceability, impact analysis, and compliance verification [13]. The documentation should be organized, easily accessible, and regularly updated to reflect changes in the requirements throughout the project lifecycle. Overall, requirements documentation is a crucial process that ensures clarity, consistency, and traceability of requirements, enabling effective collaboration and decision-making among project stakeholders [13].

Requirements management is the process of effectively managing the requirements throughout the software development lifecycle [12]. It involves activities such as requirements identification, prioritization, tracking, and control. The goal is to ensure that requirements are properly defined, documented, and maintained to meet the project objectives [15]. During requirements management, stakeholders' needs and expectations are continually monitored and evaluated to identify any changes or new requirements [15]. The process involves establishing a mechanism to handle requirement changes, assessing their impact, and obtaining necessary approvals. It also includes establishing a traceability matrix to track the relationships between requirements, project deliverables, and system

components [15]. Requirements management helps in controlling scope creep and minimizing the risk of delivering a solution that does not meet stakeholder expectations [12]. It ensures that changes to requirements are properly managed and communicated to all relevant parties. Effective requirements management contributes to project success by enabling better decision-making, reducing rework, and improving overall project efficiency [12].

Requirements validation is the process of evaluating and verifying the requirements to ensure that they meet the intended purpose and are of high quality [15]. It involves reviewing and assessing the requirements to check for correctness, consistency, completeness, and feasibility. The objective is to validate that the requirements accurately represent the stakeholders' needs and can be effectively implemented [15]. During requirements validation, techniques such as inspections, walkthroughs, and reviews are used to assess the requirements against predefined quality criteria. This process involves engaging with stakeholders, domain experts, and the development team to identify potential issues, conflicts, or gaps in the requirements [13]. The goal is to identify and rectify any deficiencies or inconsistencies early in the project lifecycle to avoid costly rework or misunderstandings [13]. Validating requirements helps in reducing the risk of developing a system that does not fulfill stakeholder expectations. It ensures that the requirements are realistic, achievable, and aligned with the project objectives [13]. By validating the requirements, organizations can enhance the quality and accuracy of the final product, improve customer satisfaction, and mitigate risks associated with requirement-related errors or misunderstandings [15].

Requirements communication is the process of effectively conveying and sharing the requirements information among project stakeholders [15]. It involves creating a common understanding of the requirements to ensure clarity, alignment, and collaboration throughout the development process. Effective requirements communication facilitates effective decision-making, problem-solving, and consensus-building among stakeholders [15]. During requirements communication, various techniques such as meetings, presentations, documentation, and visual aids are used to convey the requirements in a clear and concise manner [13]. It involves tailoring the communication approach based on the stakeholders' background, roles, and communication preferences. The goal is to ensure that all stakeholders have a shared understanding of the requirements and can provide valuable input and feedback. Requirements communication fosters effective collaboration and reduces the risk of misinterpretation or misunderstanding of the requirements [13]. It encourages stakeholders to actively participate in the requirements process, provide feedback, and contribute to the overall success of the project [13]. By promoting effective requirements

communication, organizations can enhance stakeholder engagement, improve project outcomes, and reduce the likelihood of requirement-related issues [13].

Requirements traceability is the process of establishing and maintaining the relationships between requirements and other project artifacts throughout the software development lifecycle [15]. It involves linking requirements to design elements, test cases, code components, and project deliverables. The goal is to ensure that each requirement is properly implemented, tested, and verified, and that changes to requirements are adequately managed [15]. During requirements traceability, traceability matrices or tools are used to establish bidirectional links between requirements and related artifacts. This allows stakeholders to track the origin, impact, and dependencies of each requirement [15]. Traceability helps in impact analysis, change management, and ensuring that all requirements are satisfied by the final system. Requirements traceability enhances project transparency, facilitates impact analysis, and helps in managing scope changes [12]. It enables organizations to assess the impact of requirement changes on other project elements, identify potential risks, and ensure that no requirements are overlooked during implementation. By establishing traceability, organizations can improve project visibility, reduce rework, and enhance overall project control [12].

Different types of tools and techniques are applied to follow the above requirements engineering management processes. Some of the prominent requirements management tools are JIRA, Trello, and Asana [15]. These tools play a crucial role in effectively managing and organizing requirements throughout the software development lifecycle. These tools provide a structured and collaborative platform for capturing, tracking, and prioritizing requirements, making the RE process more efficient and transparent [15]. JIRA, one of the most widely used RE tools, offers features for creating and managing requirements as well as tracking their progress. It allows users to define requirements, assign them to team members, set deadlines, and monitor their status [15]. JIRA's customizable workflows and issue tracking capabilities enable teams to easily manage requirement changes, prioritize tasks, and communicate effectively [15]. With its integration options, JIRA can be linked to other development tools, facilitating seamless collaboration between different teams, and enhancing overall project visibility [15]. Trello, on the other hand, provides a user-friendly Kanban-style interface for managing requirements and tasks [15]. It enables teams to create boards, lists, and cards to represent requirements and their associated activities. Trello's drag-and-drop functionality allows easy movement of cards across different lists, reflecting the progress of requirements from creation to completion [15]. This visual approach makes it easy to track the status of requirements, assign responsibilities, and collaborate with team members in real-time [15]. Asana is another popular RE tool

that offers comprehensive features for requirements management [15]. It provides a centralized platform for capturing and documenting requirements, setting priorities, and tracking progress. Asana's project and task management capabilities facilitate effective communication and coordination among team members [15]. It allows for the creation of detailed requirements with associated sub-tasks, due dates, and dependencies, ensuring that the entire team stays aligned and focused on achieving project goals [15].

These RE tools are useful because they streamline the requirements management process, improve collaboration, and enhance project visibility [12]. They provide a centralized repository for requirements, making it easy to access and update information. With features like task assignment, progress tracking, and notifications, these tools enable effective communication and ensure that everyone is on the same page [13]. They also offer reporting and analytics features, allowing teams to gain insights into the status of requirements, project progress, and resource allocation. Overall, RE tools like JIRA, Trello, and Asana simplify the management of requirements, enhance team productivity, and promote efficient collaboration [13]. By leveraging these tools, organizations can effectively capture, track, and prioritize requirements, resulting in improved project outcomes and increased stakeholder satisfaction [15].

3. Open Innovation: Open-Source Software

Open innovation is an approach which has gained widespread attention in recent years, particularly in the field of business and management [3]. It refers to a paradigm shift in the way organizations approach innovation, moving away from traditional, closed-door approaches to more open, collaborative models [16]. The idea behind open innovation is to leverage the collective knowledge, expertise, and resources of multiple stakeholders, including employees, customers, suppliers, partners, and even competitors, to drive innovation and create value [17].

3.1 Open Innovation

The rise of open innovation can be attributed to several factors, including the increasing complexity of products and services, the need for organizations to stay ahead of rapidly changing technologies and market trends, and the desire to harness the creativity and ideas of a broader network of stakeholders [17]. Additionally, advances in communication and collaboration technologies have made it easier than ever to connect with stakeholders and exchange information, enabling organizations to tap into a vast pool of knowledge and resources beyond their traditional boundaries.

Open innovation has the potential to offer significant benefits to organizations, including increased speed to market, greater access to new ideas and technologies, improved product quality and customer engagement, and reduced costs [3]. However, it also presents several challenges, including cultural resistance, intellectual property protection, and the management of partnerships and collaborations. Despite these challenges, the trend towards open innovation continues to grow, and it is increasingly seen as a key factor in driving innovation and competitiveness in a rapidly changing business environment [3].

3.2 Differences between open and closed Innovation

Open innovation refers to a philosophy and approach to innovation where companies actively seek ideas and solutions from a variety of sources, both within and outside the company [17]. This may include collaborating with external partners, leveraging customer feedback, or encouraging employees to generate and share new ideas. The objective of open innovation is to accelerate the pace and effectiveness of innovation by leveraging a broader range of knowledge and resources [17]. Closed innovation, on the other hand, refers to a traditional, centralized approach to innovation where ideas and solutions are generated and developed solely within the company. This approach relies on the internal R&D and product development teams to drive innovation and may limit the company's ability to take advantage of external ideas and resources [17].

3.3 Open Innovation Processes

A comprehensive analysis of the literature has revealed three fundamental open innovation processes that are applicable across industries, regardless of whether a company specializes in software, physical goods, or other domains [18]. These processes, coined as inside-out, outside-in, and coupled process by Gassmann and Enkel, represent distinct approaches [18]. Interestingly, companies examined by Gassmann and Enkel predominantly adopted one of these processes as their primary approach, while simultaneously incorporating elements from the other processes [18].

The inside-out, outside-in, and coupled processes serve as the identified open innovation frameworks. Additionally, open-source software introduces a significant decision point for software-intensive companies, as it presents the choice between an open or closed approach [18]. Therefore, open-source software is recognized as a unique method of implementing open innovation, even though software companies could also adopt the fundamental open innovation processes [18].

3.4 Open-Source Software

Open-source Software (OSS) is a type of software whose source code is made available to the public, allowing anyone to view, modify, and distribute it. It is a collaborative and community-driven approach to software development, fostering transparency, collaboration, and innovation [20]. Open innovation, on the other hand, is a concept that emphasizes the importance of seeking external knowledge and ideas to drive innovation within organizations. Open-source Software and open innovation are closely related and have a mutually beneficial relationship [16].

First and foremost, OSS embodies the principles of open innovation. By providing open access to its source code, OSS encourages participation from a diverse community of developers, users, and contributors. This inclusivity allows for the exploration and exchange of ideas, leading to the development of high-quality software through collective intelligence [16]. This collaborative approach aligns with the core tenets of open innovation, which advocate for the integration of external knowledge and expertise into the innovation process.

Open-Source Software also enables open innovation by promoting knowledge sharing and learning. The open nature of OSS projects allows developers to examine, study, and learn from existing codebases. This availability of code acts as a valuable educational resource, enabling developers to acquire new skills, understand different programming techniques, and improve their own software development practices [19]. This knowledge

dissemination contributes to the broader pool of expertise and enhances the overall innovation ecosystem.

Furthermore, OSS projects often incorporate feedback and contributions from users and developers worldwide. This crowdsourcing of ideas and feedback allows for rapid iteration and improvement of software products [3]. Developers can leverage the collective insights and experiences of a global community to identify bugs, suggest enhancements, and propose new features [3]. This collaborative feedback loop nurtures continuous innovation and enables OSS projects to evolve rapidly, often outpacing proprietary software in terms of innovation and feature development.

The relationship between open-source software and open innovation extends beyond the development phase. Many organizations have embraced OSS as a strategic tool for open innovation [20]. By leveraging open-source solutions, organizations can tap into a vast array of pre-existing software components and libraries, reducing development costs and time-to-market for their own products and services [18]. Moreover, organizations can actively participate in OSS communities, contributing back to the projects they rely on and influencing their future direction. This symbiotic relationship fosters innovation ecosystems where organizations and OSS projects mutually benefit from each other's contributions [18].

Basically, Open-source Software and open innovation share a deep connection. OSS embodies the principles of open innovation by promoting collaboration, knowledge sharing, and community-driven development [20]. It empowers developers, encourages collective intelligence, and provides a fertile ground for innovation to thrive. Open-source Software also complements open innovation strategies adopted by organizations, enabling them to leverage external knowledge, reduce development costs, and actively participate in innovation ecosystems [21]. The combination of OSS and open innovation has revolutionized the software industry, drove rapid innovation, and transformed the way software is developed, shared, and improved.

3.5 Requirements Engineering for OSS Development

From the elaborative discussion about Requirements Engineering (RE) in the previous chapter, it is evident that, RE is a critical component in the development of software projects, ensuring that user needs are captured, and system functionality is defined [8]. In the context of Open-Source Software (OSS) development, RE plays a crucial role due to the decentralized and community-driven nature of OSS projects [4]. This chapter provides an in-depth exploration of Requirements Engineering in OSS development, emphasizing its

importance and presenting best practices derived from previous research. Effective Requirements Engineering is essential for OSS projects, establishing a clear understanding of user needs, defining system functionality, and guiding the development process [4]. OSS projects rely on collaboration and community involvement, making robust RE practices vital. Well-defined requirements foster collaboration, minimize ambiguities, and ensure a shared vision among diverse contributors [4].

While OSS development presents unique challenges, the adoption of best practices can help address these hurdles. Community involvement is a cornerstone of successful OSS projects [5]. Early engagement of the OSS community and stakeholders during the requirements elicitation process is crucial. Platforms such as forums, mailing lists, and social media facilitate discussions and gather valuable input [5]. The iterative and incremental approach to requirements development and management is highly effective in the context of OSS [5]. Breaking down complex requirements into smaller, manageable user stories or tasks enables distributed contributions and promotes agility [4]. Documentation and traceability play vital roles in Requirements Engineering for OSS. Clear and up-to-date documentation of requirements, decisions, and changes ensures transparency, aids collaboration, and facilitates future understanding of the software system [4]. Maintaining effective communication channels is essential to foster discussions, gather feedback, and resolve conflicts among distributed contributors [10]. Prioritization and effective management of requirements are key to success in OSS development [4]. Prioritizing requirements based on their impact and feasibility, while considering available resources and community interests, helps guide the development process and ensure efficient allocation of limited resources [5].

In conclusion, Requirements Engineering holds great importance in the development of OSS projects, facilitating collaboration, aligning stakeholder expectations, and guiding the development process [4]. By adopting best practices such as community involvement, iterative development, and effective communication, OSS projects can overcome the challenges unique to their environment [5]. In short, it is crucial for researchers and practitioners to seek, understand and implement effective Requirements Engineering (RE) practices in OSS development.

4. Related Work

A number of systematic literature studies and mapping studies have been conducted to explore various aspects of OSS development, ranging from investigating the distinctions between traditional and OSS development activities to assessing community participation and engagement. A summary of the related work is given in Table 1 where research goals, research questions, method of the study and research summary is presented.

Studies	Goals	Research Questions	Research Method	Research Summary
Llanos and Castillo, 2012 [22]	Review of software development processes enacted by the OSS community [22]	What activities do OSS process models contain? [22]	Mapping study	Identified 22 primary studies in their systematic mapping study and reviewed software development processes enacted by the OSS community, with a focus on the requirements, design, and implementation activities in OSS projects
Gandomani et al. 2013 [23]	Assessment of the relation and integration of agile software development (ASD) and Open-source software development methodology (OSSD) [23]	RQ1: Could ASD and OSSD have any relationship? RQ2: Are practices of one of them applicable in the second? RQ3: Can they integrate with each other? [23]	Literature review	Evaluated the relationship between agile software development (ASD) and OSS development methods, finding evidence that agile and open-source practices complement each other. The findings showed that incorporating some ASD practices in

				OSS development was feasible
Franco-Bedoya et al. 2017 [24]	Evaluation of the current state of the art in OSS ecosystems (OSSECOs) research [24]	RQ1. What are the demographic characteristics of the studies about OSSECOs? RQ2. What is an OSSECO? RQ3. Which representations have been proposed for OSSECOs? [24]	Mapping study	Assessed the current state of the art in OSS ecosystems (OSSECOs) research through a mapping study of 87 primary studies. The aim of this research was to explore and understand the genesis of the OSSECO terms from related definitions, the software ecosystems in the context of OSS, as well as the modelling and analysis techniques of OSSECOs
Kiran and Ali, 2018 [25]	Investigation of requirements elicitation techniques in open-source projects [25]	How the process of requirement elicitation is carried out for open-source software? [25]	Literature survey	Explored requirement elicitation techniques in OSS projects, inspecting how the requirement elicitation process is carried out through a literature survey

Kaur et al. 2020 [26]	Review of the community participation and engagement in OSS projects	RQ1: Which studies have been published in the literature related to community dynamics? RQ2: What empirical evidence is provided for the topics addressed in the community dynamics studies?	Mapping Study	Conducted a systematic mapping study to gain a comprehensive understanding of community participation and engagement in OSS projects. Their findings provide insights into the dynamics of OSS projects and highlight the significance of community contributions and factors that drive and influence community engagement

Table 1: Overview of the selected studies

A. Castro Llanos and Acuña Castillo [22]

This paper aims to explore the variations and similarities in the development processes between open-source software (OSS) communities and the traditional software engineering practices established by IEEE Standard 1074:2006 [22]. The authors conducted a systematic mapping study to identify the activities that constitute the OSS development process, specifically focusing on requirements, design, and implementation [22]. The study begins by highlighting the increasing significance of OSS and the need to understand how OSS processes differ from traditional software engineering processes [22]. While some studies have demonstrated differences in the OSS project development processes, the authors argue that successful OSS systems exist despite deviating from the standard model.

However, other researchers claim that OSS development is not fundamentally different but rather represents an alternative perspective on software engineering activities [22].

To investigate these differences and similarities, the authors compare the development processes enacted by the OSS community with the established IEEE Standard 1074 [22]. They consider IEEE Standard 1074 as the baseline process model for traditional development. To achieve this, the authors conducted a systematic mapping study (SMS) as their research method [22]. The SMS involved identifying relevant keywords and search terms, followed by a search across electronic databases such as IEEE Xplore, ACM Digital Library, SpringerLink, Science Direct, and Scopus [22]. The inclusion criteria for selecting primary studies included papers that discussed or described the OSS development process, listed OSS or free source software development process activities, proposed an OSS development process, or focused on a particular OSS project's development process [22]. Papers that did not discuss or present the OSS development process or its activities were excluded [22].

Through the SMS, a total of 22 primary studies were identified and analyzed. The findings showed that the OSS community does not strictly adhere to prescriptive software engineering models [22]. The requirements process in OSS projects involves the evolution of requirements using web artifacts and continuous interactions in forums and messaging [22]. Unlike traditional software engineering, OSS requirements are asserted rather than elicited. Additionally, OSS projects prioritize modular design and implementation, allowing anyone, including developers and users, to contribute designs and code. The paper's structure includes a discussion of the match between the software development activities in IEEE Standard 1074 and those enacted by the OSS community [22]. The activities related to requirements, design, and implementation are compared, and the specific characteristics of the OSS requirements process are examined.

In short, the paper utilizes a systematic mapping study to investigate the differences and similarities between the development processes of OSS communities and traditional software engineering practices [22]. It provides insights into the activities involved in OSS development, highlighting variations in requirements, design, and implementation. The findings contribute to understanding the unique aspects of OSS development and its departure from traditional software engineering models [22].

B. Gandomani et al. [23]

This paper explores the connection between Agile software development (ASD) and Open-source Software Development (OSSD) [23]. The authors note that while both approaches have gained popularity in the past decade and have their respective advocates, there is a lack of rigorous studies examining the relationship and integration between ASD and OSSD. The study aims to assess the extent of the relationship and potential integration between ASD and OSSD through a systematic literature review. The research questions guiding the study are as follows: (1) Could ASD and OSSD have any relationship? (2) Are practices of one of them applicable in the other? (3) Can they integrate with each other? [23] The authors adopt guidelines proposed by Kichenham for conducting their systematic review. They search various online databases such as Scopus, IEEE Xplore, ACM Digital Library, Springer Link, Taylor, and Francis, and Science Direct [23]. The inclusion and exclusion criteria are established to ensure the selection of relevant studies. The quality of the studies is assessed based on specific questions related to the existence of a relationship, similar practices, simultaneous application, useful practices, successful integration, and feasibility of integration [23].

The results of the review indicate that there is a relationship between ASD and OSSD, particularly in terms of project management. Many authors acknowledge this relationship, and some even consider OSSD as a type of ASD. The study also reveals that there are similar practices and concepts in both ASD and OSSD, such as self-organized teams and incremental development [23]. Several case studies demonstrate successful simultaneous application of ASD and OSSD, highlighting the benefits of combining these approaches. However, when it comes to the integration of ASD and OSSD, the evidence is limited. Only a small portion of the studies explicitly address integration, and there is a lack of comprehensive integration examples. While some authors believe in the feasibility of integration and collaboration between ASD and OSSD, clear case studies supporting this claim are scarce. It is suggested that adoption plays a crucial role in successfully integrating these methodologies [23].

Basically, the study highlights the relationship and support between ASD and OSSD, showcasing their potential to complement each other in various practices. However, comprehensive integration between the two approaches remains an area of exploration, with limited evidence available [23]. The findings emphasize the need for further research and the importance of adoption for successful integration [23].

C. Franco-Bedoya et al. [24]

This paper focuses on the evaluation of the current state of the art in research related to Open-source Software Ecosystems (OSSECOs) [24]. Open-source software and software ecosystems are established areas of study in software engineering, and their interplay has significant implications for how organizations develop, acquire, use, and commercialize software. Software ecosystems offer a holistic perspective on understanding the dynamics and heterogeneity of collaborative software development, making them a suitable framework for analyzing OSS systems [24].

The objective of the study is threefold: (a) to identify the most relevant definitions associated with OSSECOs, (b) to explore the unique characteristics of OSSECOs, and (c) to investigate how knowledge about OSSECOs is represented in the literature. To achieve these goals, the authors employ a systematic mapping methodology following recommended practices [24]. They conduct automatic and manual searches across various sources and use rigorous selection criteria to retrieve relevant papers. In total, 82 papers are selected and evaluated, with identified threats to validity being addressed when possible. The results of the analysis provide insights into the research questions. The authors identify 64 terms related to OSSECOs and organize them into a taxonomy, enabling a comprehensive understanding of the terminology associated with the topic [24]. Additionally, a genealogical tree is constructed to trace the origins and evolution of the OSSECO term from related definitions. The available definitions of software ecosystems in the context of OSS are analyzed, highlighting the nuances and variations in understanding. Furthermore, existing modeling and analysis techniques for OSSECOs are classified, shedding light on the current approaches employed in the field. Based on the systematic mapping, the authors draw several conclusions [24]. They note that research on various topics related to OSSECOs, such as modeling and analysis techniques, quality models, and standard definitions, remains limited [24]. This observation suggests a need for further investigation into how organizations and OSS communities perceive and understand OSSECOs. By providing an overview of the research in the field, the paper contributes to consolidating knowledge and identifying areas that require more attention and exploration [24].

The paper presents background information on OSS and the evolution of software ecosystems [24]. It describes the systematic mapping protocol, including the criteria for selecting and analyzing the primary studies on OSSECOs. The demographic characteristics of the literature, such as publication sources, research volume over the years, and the distribution between industry and academia, are analyzed. This paper basically provides a comprehensive analysis of the state of the art in OSSECOs research [24]. It contributes

to the understanding of terminology, characteristics, and representations related to OSSECOs while identifying areas that require further investigation. The findings highlight the need for continued exploration and development of models, techniques, and definitions to enhance the understanding and application of OSSECOs in software engineering [24].

D. Kiran and Ali [25]

This paper focuses on the process of requirement elicitation in open-source software development [25]. The abstract highlights the increasing trend of open-source software development and the challenges involved in gathering requirements for such systems. The study aims to explore the techniques used for requirement elicitation in open-source software development and simplify the process [25]. Requirement elicitation is the critical step of gathering requirements from stakeholders for software development. Poor requirement elicitation can lead to software system failures. In open-source software development, where developers from different regions collaborate, requirement elicitation becomes more complex. The paper emphasizes the importance of requirement engineering in the context of open-source software development and the need for effective techniques [25]. The nature of open-source systems, which are often distributed and involve a large number of participants, makes the requirement elicitation process different from traditional software development [25]. The requirements are usually presented in natural language text format and are gathered through discussions, emails, messaging, and internet communication. The informal and unstructured nature of documentation and communication in open-source projects contributes to the unique challenges in requirement elicitation [25]. The paper categorizes various requirement elicitation techniques for open-source systems [25]. It mentions groupware tools as an effective method for eliciting requirements by facilitating communication among groups of stakeholders. Web surveys are another technique discussed, where requirements are collected through online surveys, allowing for data collection from a dispersed population. Interviews, a common requirement elicitation technique, are also mentioned [25].

The study highlights the advantages and limitations of using different requirement elicitation techniques in the context of open-source software development [25]. For example, groupware tools enable collaboration among stakeholders, while web surveys can efficiently gather requirements from a large geographical area. Interviews provide direct interaction but may be time-consuming and subjective [25]. The paper concludes with a comparative analysis of the various techniques in relation to different categories of open-source software. It acknowledges the importance of requirement elicitation in open-

source systems and emphasizes the need for selecting appropriate techniques based on the specific context [25].

This paper addresses the challenges of requirement elicitation in open-source software development and provides an overview of various techniques that can be used. It emphasizes the importance of understanding the unique characteristics of open-source projects and tailoring the requirement elicitation process accordingly [25]. The insights from this study can guide developers and organizations in effectively gathering requirements for open-source software systems.

E. Kaur et al. [26]

In this paper, the authors conducted a systematic mapping study to review the literature on community participation and engagement in Open-source Software (OSS) projects [26]. Their aim was to identify research topics, gaps, methods, and publication venues in this area. The study analyzed 67 research papers using the snowballing technique and revealed the significance of active community engagement for the success of OSS projects [26]. It emphasized the need for greater community participation through the adoption of tools, practices, and processes [26]. The authors identified five main research topics: the joining process, contribution barriers, motivation, retention, and abandonment [26]. While motivation and contribution barriers have been extensively studied, there is relatively less research available on the joining process and abandonment, indicating potential research gaps in these areas [26]. The study employed surveys and questionnaires as the primary research methodology, with most studies utilizing this approach [26].

The authors highlighted the importance of studying the joining process and abandonment in OSS projects, as these areas have received less attention in previous research [26]. They also provided insights into the publication venues and frequency of publications in community dynamics studies, as well as the countries participating in the research [26]. By presenting this information, the authors aimed to assist researchers in understanding the latest trends in the field and identifying potential research opportunities [26]. This study, being a systematic mapping study, provided a concise summary of the research area, types of research conducted, and available results [26]. While less resource-intensive compared to a systematic literature review, it still offered a valuable overview of the research landscape [26]. Moreover, the paper contributed to the existing literature by focusing on aspects of community dynamics that were not adequately covered in previous studies, such as the joining process, contribution barriers, and abandonment in OSS projects [26]. Through their systematic mapping approach, the authors provided a compre-

hensive analysis of the current state of research on community participation and engagement in OSS projects [26]. This paper serves as a valuable resource for researchers interested in community dynamics in OSS projects, offering an overview of existing research, identifying research gaps and topics, and presenting insights into research methods and publication venues [26]. The findings presented in the paper can guide future research endeavors and help researchers stay informed about the latest trends in this area [26].

While the related work explored various aspects of OSS development, none of the abovementioned studies comprehensively investigated requirements engineering processes and practices in OSS projects. An SLR focusing on requirements engineering practices would fill this gap, providing a complete understanding of the challenges and practices associated with requirements analysis and management in OSS projects. Despite the fact that these studies provided some insightful information about OSS development, there are some limitations to consider. First, the complete spectrum of OSS development cannot be represented by concentrating on the activities that comprise OSS process models. Secondly, some studies narrowed the focus by emphasizing requirement elicitation techniques and integration between agile software development (ASD) and OSSD practices. Lastly, the studies may not have accounted for the most recent trends and challenges associated with open-source software development as they are based on previous OSS practices. In contrast this thesis aims to identify the best practices for OSS project managers and contributors involved in requirement identification, analysis, and management, as well as examine the latest research on requirements practices in OSS projects and potential challenges and solutions.

5. Research Methodology

In this section literature search strategy and data extraction from the selected studies are described. To conduct the research methodology, comprehensive guidelines by Kitchenham and others [27], [28] are adopted to establish the search strategies and the review protocol before implying the systematic literature review. A systematic literature review is basically a process to identify, evaluate and interpret all available research relevant to a particular research question or specific genre or particular topic [27].

5.1 Search Strategy

1) Search String:

The following search string is applied to search the relevant journals and articles:

- (”open source software” OR oss) AND (project OR development)) AND (”requirements management” OR ”requirements engineering” OR ”requirements analysis” OR ”handle requirements”) AND (process OR tool OR technique OR activit* OR challeng* OR solution OR benefit))

The search string is formulated to include keywords identified from research questions and diversified using synonyms. It is started with the terms ”requirements management”, ”open source software project”, ”process”, ”tool”, and ”challenge”, and continue to identify the synonyms, abbreviations, or alternatives to those words or terms to widen the coverage of the search. This is also the reason for using Asterix (*) to include variations of certain keywords in the search. The key terms are concatenated using AND, so that it is easier to search for relevant studies. On the other hand, concatenation of the alternatives and synonyms using OR is implied so that it could maximize the percentage of search results being relevant. To ensure consistency and completeness, the presence of terms is verified in several iterations to look for relevant terms from papers chosen from the rounds of search results.

2) Information sources and search process:

The same search string is used to search for relevant studies. Four most representative databases for research in software engineering are chosen in the study, and they are as below:

I. Scopus
II. IEEEXplore
III. ACM Digital Library
IV. Web of Science.

Scopus was the first database where the search was done, duplicate values that were found from Scopus are removed from research results in other databases. Overall, 256 unique results came from database searches.

3) Inclusion and exclusion criteria:

Inclusion/Exclusion	Criteria
Inclusion	• Papers on open-source software development process • Papers on how to handle issues and contributions in OSS projects • Papers applying methods, techniques, or tools for requirements elicitation, analysis, and management in OSS projects • Papers analyzing contributors' influence in handling requirements for • OSS projects
Exclusion	• Not in English • Focuses on OSS evaluation and adoption • Focuses on requirements specification for an OSS project • Focuses on using OSS projects as examples for tool/method evaluation • Focuses on open-source tools • Duplicate papers • Out of topic and using the terms for other purposes • Non-peer-reviewed papers • Secondary studies, vision papers, or tutorial

Table 2: Inclusion and Exclusion Criteria

The inclusion and exclusion criteria for the study have been specified based on the research questions. The inclusion criteria require that studies investigate relevant topics to address our research objectives. The included studies are those which meet four specific criteria. Exclusion criteria include those commonly used in SLR, such as non-English papers, duplicate papers, non-peer reviewed papers, research plans and roadmaps, and the use of a secondary study method. Additionally, studies related to OSS adoption, evaluation of a requirements tool using OSS projects, or the development of requirements tools are outside the scope of this study and are listed in the exclusion criteria. The complete

list of inclusion and exclusion criteria is given in Table 2. The initial screening of search results for relevant studies is conducted after several revision and applied the inclusion and exclusion criteria to select relevant papers. The results are compared after the independent screening process to reach a consensus on the papers to be included. If there is a dispute between results, the full paper is reviewed, checked and read again, and then after the final confirmation it is determined whether the paper should be included or not. There are altogether 39 selected papers.

Each of the 39 papers underwent the forward and backward snowballing process. All the references listed in the selected papers are reviewed and all the papers that reference the selected ones are evaluated. The process provided 3 additional papers in forward snowballing and 1 additional in backward snowballing. Hence, the total number of selected papers was 43. The number of papers resulting from each process is summarized in Figure 1.

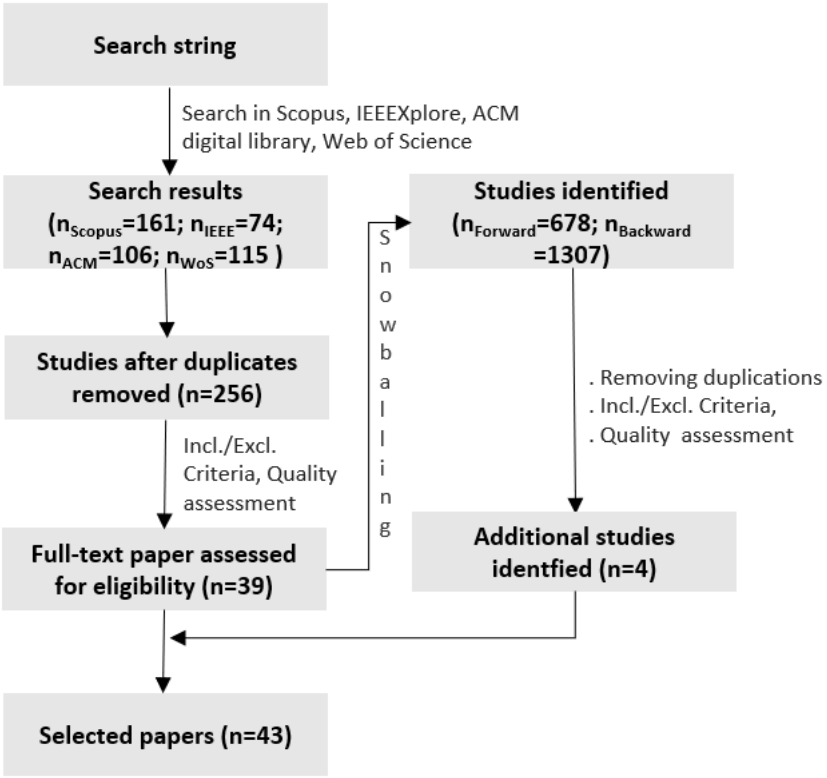

Figure 1: Search and Selection process

5.2 Data Extraction

In order to extract data from the 43 studies, a coding schema is developed that is based on the research objectives and questions. In the coding schema, eleven steps are implied for data extraction and those steps are as below:

1. Title of the paper
2. Extracted data (Brief summary of the paper)
3. Goal of the paper
4. Method of the paper (case study, survey research, experiments, action research, constructive method, etc.)
5. Requirement Engineering activities
6. Techniques that are involved in those RE activities
7. Tools that are used in those RE activities and techniques
8. Stakeholder involvement in RE activities and their roles
9. Identified challenges/constraints/problems in OSS projects
10. How these challenges affect the overall RE management in project
11. Corresponding solutions to identified challenges in OSS projects

This coding schema is conducted through an excel datasheet as like in Figure 2:

RM activities	Techniques that are involved in those RM activities	Tools that are used in those RM activities and techniques	Stakeholder involvement in RM activities and their roles	Identified challenges/constraints/problems in OSS projects	How these challenges affect the overall RM in project	Corresponding solutions to identified challenges in OSS projects
The paper focuses on the requirement engineering process in relation to OSS communities, exploring how Sony Mobile's involvement in these communities affects their requirements engineering practices.	The paper does not explicitly mention any specific techniques for requirement management, but it does highlight the importance of engagement and informal processes in Sony Mobile's requirement engineering towards the OSS communities.	The paper focuses on Sony Mobile's use of Jenkins and Gerrit in their continuous integration tool chain as a means of involvement with OSS communities.	The paper does not explicitly discuss stakeholder involvement in requirement management activities, but it does highlight the importance of engagement with the OSS communities in Sony Mobile's requirements engineering process.	The paper does not provide a comprehensive list of challenges, constraints, or problems in OSS projects but does highlight the need for systematic and automated testing, which is still in its infancy.	The paper suggests that the need for systematic and automated testing is still in its infancy, which may impact the overall requirement management process for Sony Mobile in OSS projects.	The paper does not provide specific solutions to the challenges identified but does suggest that the innovative outcomes resulting from Sony Mobile's involvement in OSS communities, such as free features and maintenance, increase speed and quality in development.

Figure 2: Sample datasheet of coding schema

This coding process is conducted in three phases. Firstly, eight papers are selected from the list and extracted data intensively using the coding schema. After that, the extracted data has been compared, the data items and the coding schema have been refined and revised. This coding schema helps to identify characteristics of the selected studies such as their goals, research methods, requirements engineering management activities and practices reported, methods and tools used, challenges faced in requirements management in OSS development, and proposed solutions. After finalized the coding schema, the papers are coded using the schema in the second stage. At this point, several discussion meetings are conducted with the supervisor to interpret and analyze the codes based on perceived similarities, and to discuss any confusion and discrepancies. In cases of disagreement, the full paper is read again thoroughly and intensively. In the third phase, the data is summarized by identifying themes that emerged from the codes. These identified themes formed the categories reported in the results section.

6. Results

6.1. Overview of Selected Studies

43 papers are selected through the process described earlier. The papers are listed in Appendix A and cited as SP* in the following discussion. They were published from 2002 to 2022. The distribution of the number of papers over time is shown in Figure 3. Among these 33 papers were published in conferences and 10 were published in journals.

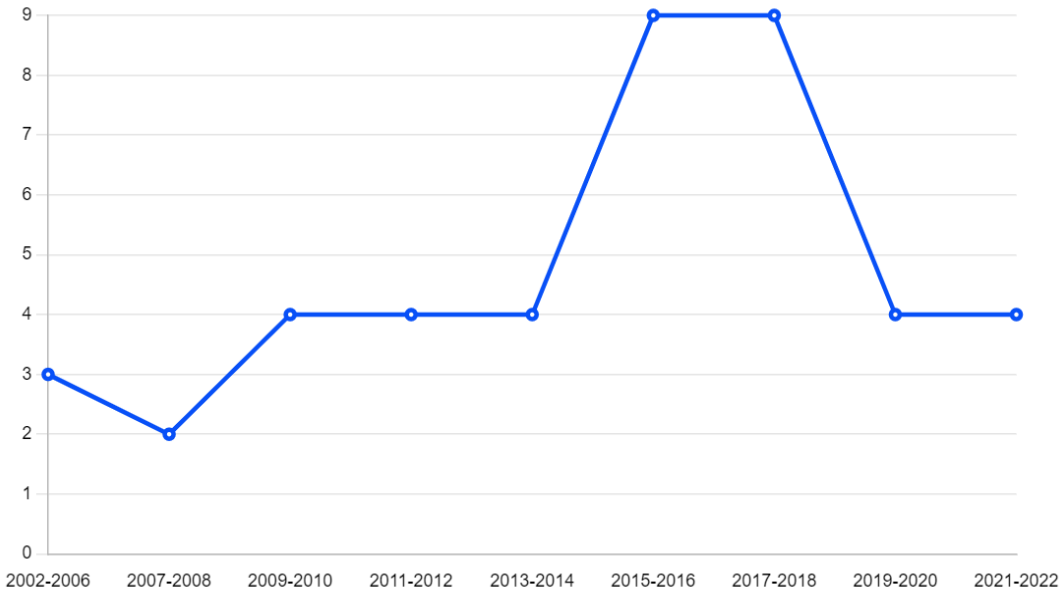

Figure 3: Year wise distribution of selected studies

From 2005 to 2008, the number of selected studies stayed unchanged. Nevertheless, in 2009-2010, a modest rise in the number of studies was seen. The most notable rise in the selected number of studies occurred between 2015 and 2016 when 9 studies were published. This shows that several research was conducted, and conferences received high-quality submissions during this time period. The number of chosen research reduces little between 2017 and 2018 with nine publications. The decline continued until 2022. When analyzing the characteristics of selected studies, it is identified that the themes emanating from the coding of research goals and methods of the selected studies. The studies have been categorized into different areas and found that Data analytics (26.6%), RE processes (35.56%), RE activities (31.11%), and Techniques and tools (6.67%) are the key methods that emerged from requirements management practices.

The selected studies are analyzed and grouped into two categories based on their research topics, as shown in Figure 4. There are 29 studies (67.44%) on the characteristics of the requirements engineering process and practices in OSS projects and 14 studies (32.56%) that focus on specific requirements analysis and management activities and propose and discuss techniques or tools supporting those activities. Among the studies on techniques and tools supporting requirements management tasks, there are 11 studies (25.58%) on the use of data analytics methods for analysis.

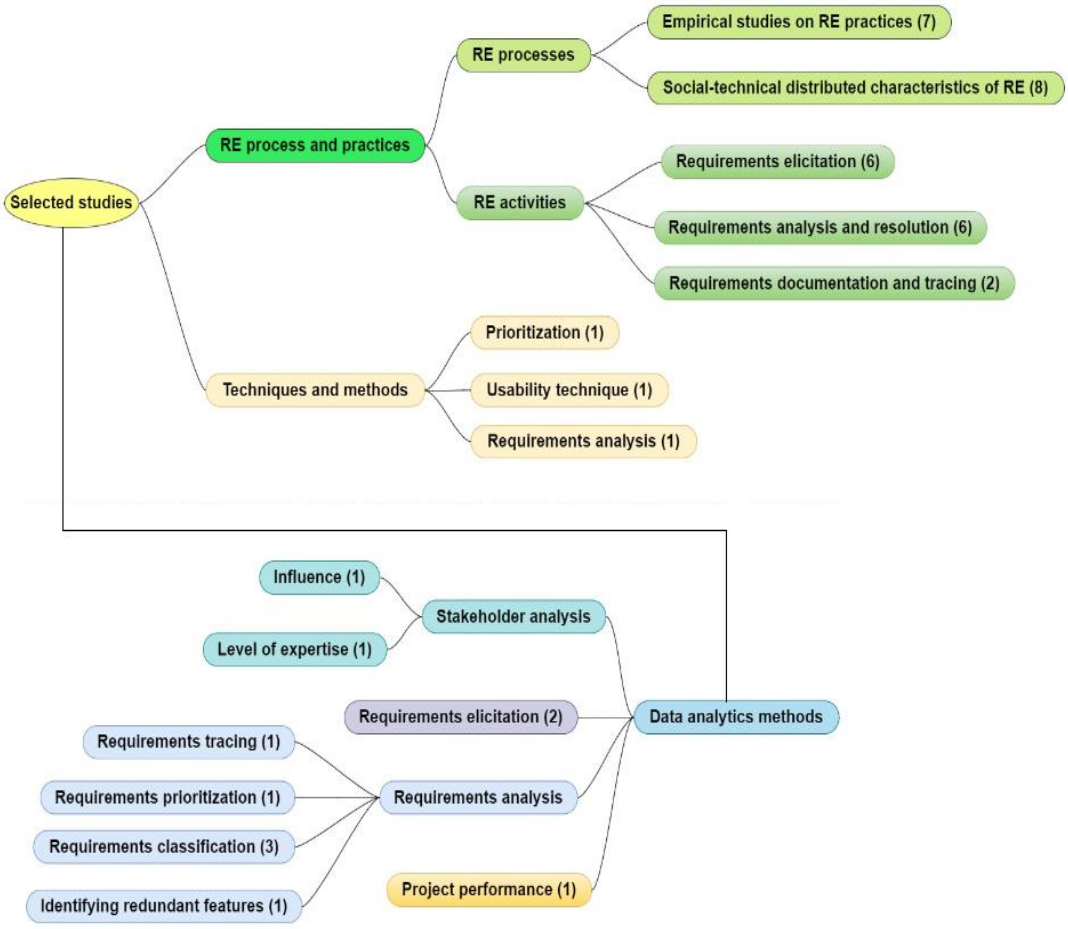

Figure 4: Study topics divided into categories

The 29 studies on RE processes and practices revealed several main themes. Numerous studies focused on the social-technical distributed aspects of RE practice in open-source communities, examining the processes and frameworks involved [SP9], [SP10], [SP15], [SP16], [SP18], [SP25], [SP34], and [SP40]. Meanwhile, other research investigated the informal and open innovation nature of software engineering practices, particularly regarding the RE process [SP13], [SP24], [SP30], [SP42], [SP8], [SP21], and [SP26]. Furthermore, several papers conducted empirical study on specific RE activities, e.g., requirements elicitation [SP14], [SP17], [SP20], [SP23], [SP32], [SP39], analysis [SP5], [SP12], [SP19], [SP31], [SP36], [SP43], and documentation and tracing [SP35] and [SP38].These studies explored various aspects of RE in the context of OSS development. Out of the 14 studies on techniques and tools for requirements analysis and management, 11 recent studies proposed and evaluated data analysis methods such as natural language processing (NLP), machine learning (ML), or statistical analysis algorithms to accomplish various tasks. These tasks include analyzing stakeholders' influence [SP3] and expertise [SP29], eliciting requirements from developer communities or forums [SP6] [SP28], identifying redundant features [SP33], classifying or prioritizing requirements [SP1], [SP22], [SP27], [SP4], tracing requirements [SP37] and monitoring project performance [SP2]. Additionally, 3 studies proposed and evaluated requirements elicitation and analysis techniques that primarily relied on participants' discussions and voting [SP4], [SP7], and [SP11].

6.2. RQ1: What RE activities, tools and techniques are used by practitioners in open-source software project?

Synthesizing data from 43 selected papers, the practices, techniques, and methods that support RE activities in OSS projects are summarized in Table 3. After analyzing the data, various sorts of RE activities are identified where several sorts of tools and techniques are involved. Based on the following RE activities, the identified practices are categorized as in Table 3.

1) Requirements Elicitation: Proposing and identifying change requests and new features that align with a project's roadmap is crucial in software development and ongoing improvements. Numerous studies have emphasized the responsibility of core developers and team members in determining the majority of features (SP8, SP14, SP23, SP25, SP32, SP34, SP42). Feature requests and bug reports are often conveyed through provisionments, a statement that describes a feature provided by existing software systems, competing products, or prototypes put forth by developers advocating for the changes(SP30). These provisionments and assertions are openly discussed within the developer community, drawing upon personal experiences or knowledge of user needs. Such discussions

typically take place on forums, during project workshops, through emails, or by using reporting tools such as Jira, Bugzilla, or other issue tracking systems (SP8, SP17, SP25, SP32, SP34). Meanwhile, studies SP40 and SP23 have suggested improvements to the decision-making process for selecting proposed use cases, constraints, and feature re quests.

Requirements Engineering Activities	**Practices**	**Related Papers**
Requirements elicitation	Assertion & discussion in developer's community	SP8, SP14, SP17, SP23, SP25, SP30, SP31, SP32, SP34, SP40, SP42
	Automated requirements identification	SP27, SP28, SP43
Requirements analysis	Automated requirements classification	SP1, SP4, SP6, SP33, SP43
	Requirements prioritization	SP7, SP8, SP14
	Refinement and resolution	SP5, SP12, SP14, SP19, SP25
	Requirements analysis	SP25, SP31, SP32, SP34, SP35, SP36, SP40, SP42
Requirements documentation	Informal documentation	SP14, SP25, SP34, SP35, SP38, SP40, SP42
Requirements management	Feature management	SP32
	Traceability	SP26, SP31, SP32, SP35, SP37, SP40, SP38
Stakeholder analysis	Community involvement	SP17, SP39, SP42
	Data analytics methods	SP2, SP3, SP18, SP20, SP28, SP29
Social-technical characteristics	Socio-technical distributed cog-nitive process	SP9, SP10, SP15, SP16
	Informal requirements	SP25, SP30
Others		SP2, SP11, SP13, SP14, SP21, SP22, SP24, SP40, SP41

Table 3: Requirement Engineering Practices

Furthermore, to automate the labor-intensive process of identifying requirements from diverse information sources, NLP techniques, ML methods, and grammar-based parsing strategies (SP43) have been applied in recent studies. For example, a logistic regression model has been proposed to identify security requirements from reported issues (SP27).

2) Requirement Analysis and Validation: Requirements analysis involves tasks such as identifying redundant requirements and classifying and prioritizing issues and change requests. These tasks rely on automated methods, contributors' feedback and comments, and diverse information sources. Several studies proposed automated approaches for requirements classification, using and comparing algorithms such as TF-IDF, Naive Bayes, Decision Tree, and Random Forest (SP1, SP4). Additional automated approaches have been suggested for identifying redundant feature requests (SP33) and classifying security requirements (SP43). Besides automated methods, requirements prioritization and refinement often involve informal discussions, comments, and voting processes, where community members, e.g., project maintainers, developers, and users, collaboratively assess the proposed features and improvements (SP5, SP8, SP14, SP25). Requirements validation is typically conducted through unit testing and integration testing of change requests (SP7, SP8, SP14, SP19). Researchers have explored community communication, scenarios of usage, How-to guides, external publications, bug reports, issue tracking, and documentation as sources for requirements analysis (SP14, SP25). Furthermore, several studies have explored approaches and frameworks to better understand OSS requirements and address communication problems in OSS development (SP34, SP36, SP40).

3) Requirements documentation: Documentation is typically less formalized in OSS projects. Studies indicate that requirements are documented through discussion and voting (SP14, SP25, SP34, SP40, SP42) and in various formats such as forums, guidelines, READMEs, wikis, and website resource files (SP35, SP38). Although not always required, templates for reporting issues and generating change requests are frequently used in OSS projects to provide guidelines for contributors when submitting their work or reporting problems. For instance, guidelines for documenting and managing issues for agile software development on GitHub have been proposed in studies (e.g., SP35).

4) Requirements management: In OSS projects, requirements management and tracing are achieved through issue tracking systems, source control systems, informal documentation, and tags or labels. The degree of traceability varies based on project size, complexity, community involvement, as well as the abstraction level of requirements. Concrete and technical requirements are often reported in the project's issue tracker and can be traced with the support of issue references and status tracking (SP35, SP40), while

features documented in READMEs, wikis, and website resource files may lack explicit referencing to source code (SP38). Additionally, Requirements tracing has been explored through the use of automated approaches based on the Universal Sentence Encoder following a semantic search and innovative clustering technique (SP26), Vector Space Model (VSM), Term Frequency-Inverse Document Frequency (TF-IDF) techniques (SP37).

5) Stakeholder analysis: Many studies highlight the importance of community involvement and participation in OSS projects, as well as the need for effective stakeholder management and communication throughout the software development life cycle. These studies result in preliminary guidelines for the organization of community-oriented software development (SP17, SP39, SP42). Many recent studies applied quantitative analysis techniques such as social network analysis (SP2, SP3, SP18, SP20), Markov Network (SP28, SP29), and graph theory (SP20) to model and analyze stakeholder interactions patterns and their influence, and their expertise level to a given topic.

6) Social-Technical Characteristics: OSS projects represent complex relationships between the social aspects (e.g., collaboration, communication, stakeholders) and technical aspects (e.g., software architecture, code base, tools). Understanding these characteristics is essential for promoting a healthy OSS ecosystem. Some studies have investigated the impact of social-technical characteristics on OSS projects, examining the role of distributed and cognitive activities across releases (SP9, SP10, SP15) and the influence of internal social capital in driving RE activities (SP16). Studies highlight that informal requirements are common in OSS projects (SP25, SP30).

Moreover, various other aspects of RE practices in OSS projects have been explored, including the impact of crowd-sourcing on OSS project performance and process effectiveness evaluation (SP2), the feasibility of adopting the visual brainstorming usability technique in OSS project (SP11), models for analyzing and managing requirements in OSS and Software Ecosystems (SECOs) (SP13), challenges in managing requirements in an open innovation context (SP21), as well as the frameworks and models to describe the requirements analysis process in OSS projects (SP24, SP41).

6.3 RQ2: What are the challenges faced by the practitioners in OSS projects? What are the possible solutions to overcome the challenges?

Studies have demonstrated that diverse challenges may arise in the process of OSS development. Identified challenges and corresponding solutions from the selected papers are presented in Table 4.

Challenges	Corresponding Solution	Related Papers
Classifying user inputs into requirements categories	Classification model/ML	SP1
Managing volume of system requirements in crowdsourcing	Organize, prioritize, employ CrowdRE, dedicated managers	SP2
Complexity of RE process in OSS ecosystems	SIA method for stakeholder analysis	SP3
Prioritizing new requirements and responsible developers	Prioritization tool with real-world data and ML classifiers	SP4
Prioritizing requirements for individual contributors	Utility-based prioritization approaches	SP7
Need for systematic and automated testing in OSS projects	Leverage benefits from industry involvement	SP8
Impact of social structures on RE quality	Quantitative analysis of OSS projects	SP9
Recruiting OSS users for usability testing	Modified visual brainstorming, improved recruitment process	SP11
Resource and coordination issues	Improved communication, resource allocation, prioritization	SP12
Challenges in OSS and SECO projects	Agile methodologies, stakeholder involvement, new tools	SP13

Informal requirements generation	Inform future RE improvements	SP14
Handling large volume and volatility of requirements	Understand socio-technical distribution of cognitive processes	SP15
Understanding internal social capital	Case study of four OSS development projects	SP16
Limited scope of empirical studies	Guidelines for community-oriented software development	SP17
Openness-related challenges in OSS projects	Network analysis for stakeholder understanding	SP18
Lack of knowledge in JIT requirements engineering	Further research in JIT requirements engineering	SP19
Decentralized social interactions in OSS	Understanding structural hole theory	SP20
Requirements scoping and management in open innovation	Improved data-mining tools, second-generation forums	SP21

Limited understanding of the relationships between requirements and project success, difficulty in identifying exceptions or anomalies, and limited research on the impact of requirements and design evolution	Bridge the knowledge gap through exploration and pattern identification, develop a comprehensive taxonomy for requirements, and replicate analyses on larger datasets to gain a broader understanding of failed projects and their impact	SP22
Complex web of sociotechnical processes, development situations, and dynamically emerging development contexts	Ethnographic methods are needed to elicit, analyze, validate, and communicate	SP25
Manual extraction of requirements from user feedback	Universal sentence encoder	SP26
Limited analysis in requirements identification models	Use more metrics, test complementary combinations	SP27
Eliciting requirements knowledge from online discussions	Tool-supported method with NLP, ML, and search techniques	SP28
Evaluating expertise of stakeholders in OSS projects	Enriched concepts from forum- and social network-based works	SP29
Inappropriate traditional RE practices in some projects	Taxonomy of requirements practices	SP31
Managing large numbers of feature requests	TF-IDF VSM for measuring text-based similarity	SP32

Presence of inconsistency, incompleteness, and ambiguities as the major hindrance for requirements communication	Proposed a novel framework to tackle communication problems and ambiguities	SP36
Lack of tool support to visualize and navigate large networks of feature requests and their relationships	TF-IDF VSM to measure the text-based similarity between feature requests	SP37
Lack of guidance in ERP requirements management	Further research for practical guidelines	SP39
Distributed user communities, user needs, and decision-making	Web-based project management technologies, planning strategies	SP40

Table 4: Overview of Identified Challenges and Solutions

The challenges can be grouped into several distinct categories, with various studies exploring and proposing solutions to address them.

1. Managing the large volume of issues and feature requests: Several studies have highlighted the challenges in managing the growing volume of system requirements, feature requests, and issues in OSS projects (SP2, SP15, SP21, SP22, SP32). There is often a lack of support for handling and managing a large number of feature requests in forums and wikis, making it harder for users to submit well-thought-out requests (SP33) and for developers to visualize and navigate extensive networks of feature requests (SP37). These challenges affect the overall RE process by making it difficult to understand, identify, classify, and clarify requirements, leading to delays in software development, increased bugs and errors, and higher development iterations (SP36). To address these challenges, various solutions have been proposed. These include the implementation of processes to organize and prioritize issues and change requests, the application of CrowdRE techniques, and the assignment of dedicated managers to effectively manage input from the crowd (SP2). Utility-based prioritization techniques have been proposed to facilitate decision-making in selecting the most important and urgent requirements and assigning them to suitable developers for review and implementation (SP7). Furthermore, recent research trends involve utilizing ML methods to handle the large volume of requirements, such as automated requirements identification and analysis (SP1, SP4, SP6, SP27, SP28,

SP33, SP43). For instance, a prioritization tool was proposed in SP4 to recommend relevant requirements (issues/bugs) to open-source developers. The study used data from Eclipse to build a prediction model, training and evaluating different classifiers. The results revealed that the Random Forest classifier offered the highest precision.

2. Distributed nature of OSS projects: The distributed nature of OSS projects presents unique challenges in communication, management of requirements, and resource allocation due to the cognitive distribution among multiple developers and their interaction in social networks. RE is considered to be less formal and dependent on online documentation and communication tools. To address these challenges, several approaches have been proposed, including stakeholder influence analysis (SIA) methods that analyze developers' social networks and evaluate their influence and expertise (SP2, SP3, SP18, SP20, SP28, SP29). These analyses support identifying responsible developers for requirements analysis and implementation. In particular, the study in SP20 investigated the impact of structural hole theory on the identification of new requirements in OSS projects. The findings enhance the RE process in decentralized environments by understanding the influence of social network structures on OSS development. Lack of resources, conflicting stakeholder interests, and coordination problems among contributors are identified as challenges (SP12) that can result in delays in the elaboration, refinement, and resolution of just-in-time requirements. To address these challenges, various solutions have been proposed including enhancing communication and coordination among contributors, appropriate resource allocation, prioritizing requirements based on stakeholder feedback, and resolving stakeholder conflicts through negotiation and mutual compromise (SP12).

3. Complex Social Structure: Another challenge in OSS projects is found in the studies, which is the complex and multifaceted characteristics of the RE process. The reason behind it is the competitive RE processes inherent in the ecosystem (SP3). The SIA method is proposed to support firms to maintain the structure of their stakeholder analysis processes in regard to an OSS ecosystem and identify the concepts of influence and interactions (SP3). The impact of diversified social structures on RE quality is another challenge addressed in the studies (SP9, SP16). The authors advised that understanding the role of internal social capital in RE activities could be beneficial to address these challenges in OSS projects. Additionally, the relationship between structural hole theory and the identification of new requirements is proposed to improve the RE process in decentralized environments like OSS projects (SP20). Another identified challenge is that the presence of inconsistency, incompleteness, and ambiguities as the major requirements communication hitches in OSSD context (SP36). It also identified the challenges posed by the

heterogeneous stakeholders with different educational backgrounds, geographical locations, and language use competencies. It can make the process of understanding, classifying, and clarifying unclear requirements a tedious task, leading to delays in software development processes, increased defects, and higher development iterations. As a solution, the authors proposed a novel framework to tackle communication problems and ambiguities in OSSD by merging various approaches for their collective impact. The framework aims to combine the positive attributes of automation-oriented domains and support humans to arrive at precise and unambiguous requirements (SP36).

4. Requirements prioritization and identification: Prioritization of new requirements and identification of responsible developers for the implementation is another crucial challenge specified by several studies (SP4, SP8). An effective solution is proposed in SP4 by developing a prioritization tool that provides a solution to this challenge. This tool recommends relevant requirements (issues/bugs) to open-source developers (SP4). The tool utilizes real-world data from Eclipse and machine learning classifiers to build a prediction model that can accurately predict issues. Another challenge is in this case, to elicit requirements knowledge from big online discussions and to elicit indicators of the level of expertise of the participants to a given discussion (SP28). This can make manual analysis of online discussions an effort demanding and error-prone task. A tool-supported method that combines Natural Language Processing (NLP) techniques, Machine Learning (ML), statistical and search-based techniques is proposed by the authors to address the identified challenge (SP28).

In addition, numerous solutions have been proposed to figure out these challenges, for instance utilizing a universal sentence encoder, which helps to identify matched requests coming from forum postings and issue tracker records (SP26), a tool-supported method combining NLP techniques, ML, statistical, and search-based techniques to address the challenge of requirements elicitation from big online discussions (SP28), and a novel framework to handle communication problems and ambiguities in OSSD by merging several approaches (SP36). The studies also recommend the implementation of planning and communication strategies to enhance the decision-making process (SP40) and the need for future research to derive more practical guidelines on the requirements definition for ERP development (SP39).

7. Discussion

In this study, a comprehensive and detailed analysis of the best practices of RE activities is performed in the context of OSS projects. The results and data from numerous studies are synthesized to analyze the practices, techniques, tools, and methods applied in various RE activities, for instance, requirements analysis, requirements elicitation, documentation, management, stakeholder analysis, and social-technical characteristics. Throughout the findings, the diversified range of RE practices and corresponding tools and techniques for example, NLP techniques, statistical and search-based techniques, ML approaches, etc. applied in OSS projects has been emphasized. The significance of community involvement and participation in OSS projects and as well as the importance of informal requirements for OSS project development and management are highlighted in our findings, with many studies pointing out the importance of effective stakeholder management and communication throughout the software development life cycle. Several studies have analyzed the social-technical characteristics of OSS projects with in-depth reasoning of the complex relationship between social aspects and technical aspects. To assure the efficient and effective management of requirements in OSS projects, it is important to manage traceability between requirements and implementation artifacts. Our research suggests that forums are generally used for feature management, while traceability has been implied through automated approaches, recommender systems, and lightweight representations.

7.1. Analysis of the findings

According to the results of the comprehensive analysis of the relevant literature, research trends on requirement management practice have grown significantly since 2005. Between the years 2005 and 2008, there were very few studies that concentrated on the topic of requirement management for open-source software projects. In terms of how trends change over time, it is possible to observe that from 2009 to 2012, a variety of studies were conducted. These studies focused on the need for classification and analysis as well as understanding the requirements for OSS, which includes socio-technical processes. During this time period, there was a lack of coherence in the research subject. There has been a rise in interest in the social-technical distributed aspects of requirement engineering in open-source projects from 2016 to 2018, as evidenced by the publication of six articles that particularly focus on this aspect of the topic. Since 2017, there has been a reduction in the number of research studies conducted on RE processes, in particular software engineering (SE) practices on the subject of open innovation (OI) and the requirements management process. To enhance project performance, eliminate redundant data, and for classification and analysis of requirements, an increasingly popular trend is the use of data analytics techniques. The use of data analytics began in 2011, but the majority

of work started in 2015. There were multiple works published in 2018 that focused on the tools and techniques for requirement analysis. After 2015, in 2022, a study concentrated on documentation for an OSS project, disclosing an intriguing topic that can be combined using data analytics for future studies. Basically, it is important to note that the majority of studies on Requirements Management on Open-Source Projects were conducted in 2018 and 2019, with ten studies being the most productive years.

Over the past years, the research in RE practices in the context of OSS projects has evolved. Prior to 2010, studies solely drew attention to understanding the social processes, developer community, and informal text descriptions of requirements. Several case studies and surveys were applied to point out how requirements analysis and management differ from traditional RE practices. Since 2009, there has been a deviation in discussing methods and techniques for individual RE activities such as requirements elicitation, prioritization, and tracing. Researchers have performed a flourishing interest in community involvement and participation in RE development for OSS projects. From then, the use of mixed methods approaches, combining qualitative and quantitative data techniques were also widely used. With the expanding application of NLP and ML methods, automotive requirements identification and analysis in OSS projects have also become a trending topic since 2015.

Overall, the findings reported in the selected papers are inspiring as they show an increasing interest in improving RE practices in OSS projects, with a particular focus on community involvement and automation of tasks like requirements identification, classification, and tracing. Looking ahead, we see the research trends will continue to focus on community involvement, automation of tasks, and the integration of large language models (LLMs) and artificial intelligence (AI) tools to assist the RE process for OSS development. These promising trends will undoubtedly contribute to the advancement of RE practices and the growth of OSS projects. In a nutshell, this thesis provides a comprehensive overview of the best practices of RM activities in OSS projects. These findings can serve as a valuable resource for software practitioners and researchers in the OSS community, helping them to understand the diverse range of practices and techniques used in RM activities and how to effectively manage requirements in OSS projects.

7.2. Threats to Validity

To ensure the validity of our SLR study, search strategies and a review protocol are established based on guidelines [27] [29] [28] and followed the four categories recommended in [30] to discuss potential threats. To ensure the comprehensiveness of the search, a diversified search string is utilized, incorporating synonyms relevant to the research questions. The queries are executed in major citation databases such as Scopus and Web of Science, as well as digital library portals including IEEE Xplore and ACM Digital Library. Additionally, snowballing is employed as a complementary method to extend the coverage of studies, encompassing the review of references in the selected studies and evaluation of papers that referenced the selected ones. This process resulted in the identification of four additional relevant publications. To mitigate threats associated with construct validity, the search strategy, review protocol, and data extraction process are entirely based on established guidelines [27] [28] [29]. These guidelines served as a framework to ensure adherence to best practices in conducting an SLR, minimizing the potential for bias and errors [27] [28] [29]. By following a standardized approach, the study design is aimed to maintain consistency and accuracy throughout the research process.

To minimize threats related to inaccurate data extraction and conclusion validity, measures are implemented to promote consistency and rigorous analysis. Sample extractions are thoroughly discussed to achieve consensus among me and the supervisors, facilitating uniformity in the data extraction process. Furthermore, the extracted data were diligently recorded in a shared excel file, allowing for easy access and verification. This approach aimed to enhance transparency and enable the replication of the study by other supervisors, thus strengthening the reliability of the findings.

While the possibility of omitting relevant studies is acknowledged, it is asserted that the SLR study represents a comprehensive overview of relevant literature within the defined scope. The rigorous application of established guidelines and the employment of diverse search strategies and complementary methods aimed to minimize the likelihood of excluding important studies. By diligently following these procedures, it is endeavored to ensure that the findings are based on a robust and comprehensive collection of relevant literature. Regarding threats to external validity, it is acknowledged that minor limitations may exist. External validity pertains to the generalizability of the study findings to broader populations or contexts. Although specific details regarding the mitigation of these threats are not provided, it can be inferred that the thesis aimed to achieve a reasonable level of external validity by conducting an extensive search across relevant databases and digital libraries. By adhering to established guidelines, employing diversified search strategies, utilizing multiple databases, and implementing rigorous data extraction and review processes, this thesis is aimed to mitigate potential threats to validity. While ac-

knowledging the potential omission of relevant studies, it is asserted that the SLR provides a comprehensive overview within the defined scope. Additionally, it is acknowledged that minor threats to external validity may exist, emphasizing the need for careful consideration when generalizing the findings to broader populations or contexts.

8. Conclusion

The systematic literature review conducted in this study aimed to investigate the practices and research related to requirements engineering (RE) in open-source software (OSS) projects. By synthesizing the findings from 43 selected studies, we have gained valuable insights into the techniques, methods, and practices employed in various RE activities, such as requirements elicitation, analysis, documentation, management, and stakeholder analysis. Through our review, we have identified a diverse range of practices and techniques applied in OSS projects. Notably, the use of machine learning (ML) methods, natural language processing (NLP) techniques, and other statistical approaches have emerged as effective tools for requirements identification and analysis. These advanced technologies leverage diverse sources of information associated with OSS projects, enabling more accurate and efficient RE processes. One interesting trend highlighted by recent studies is the growing interest and potential for integrating AI tools into RE activities. Researchers and practitioners are recognizing the benefits of utilizing AI to assist in requirements engineering, enabling automated analysis, identification of patterns, and decision-making support. This indicates a shift towards more data-driven and intelligent approaches to RE in OSS projects. Additionally, our findings underscore the significance of community involvement and participation in OSS projects. The nature of open-source development fosters active stakeholder dynamics and emphasizes the importance of effective communication throughout the software development lifecycle. The involvement of diverse stakeholders brings valuable insights and perspectives, contributing to the overall quality of the requirements and the success of the project.

In summary, this thesis provides a comprehensive review of the best practices for RE activities in OSS projects. The synthesis of findings from various studies offers software practitioners and researchers in the OSS community a valuable resource to understand and leverage the wide range of practices and techniques utilized in RE. By adopting these practices, practitioners can effectively manage requirements in OSS projects, leading to improved project outcomes and increased stakeholder satisfaction. As the field of RE in OSS continues to evolve, it is crucial to stay updated with the latest practices and trends. The findings of this study not only serve as a snapshot of the current state of RE in OSS but also provide a foundation for future research and exploration in this area. By building upon these findings, researchers can further advance the field by proposing novel approaches, validating existing techniques, and addressing the unique challenges and opportunities presented by the OSS context. In conclusion, this study contributes to the body of knowledge in RE for OSS projects by providing a comprehensive overview of practices, techniques, and research trends. The insights gained from this study will support

practitioners and researchers in making informed decisions regarding RE practices, enabling them to navigate the complexities of OSS projects more effectively. By leveraging these findings, the OSS community can continue to innovate and deliver high-quality software that meets the evolving needs of users worldwide.

References

[1] Huizingh, E.K.: "Open innovation: State of the art and future perspectives." Technovation, vol. 31, no. 1, pp. 2-9 (2011)

[2] European Commission: Open Innovation Open Science Open to the World - a vision for Europe. European Commission, Brussels (2016)

[3] Enkel, E., Gassmann, O., Chesbrough, H.: "Open R&D and open innovation: exploring the phenomenon." R&D Management, vol. 38, no. 4, pp. 311-316 (2009)

[4] Scacchi, W.: Understanding requirements for open-source software. In: Design Requirements Engineering: A Ten-Year Perspective: Design Requirements Workshop, Cleveland, OH, USA, June 3-6, 2007, Revised and Invited Papers. pp. 467–494. Springer (2009)

[5] Paech, B., Reuschenbach, B.: Open source requirements engineering. In: 14th IEEE International Requirements Engineering Conference (RE'06). pp. 257–262. IEEE (2006)

[6] Beller, M., Bacchelli, A., Zaidman, A., Juergens, E.: Modern code reviews in open-source projects: Which problems do they fix? In: Proceedings of the 11th working conference on mining software repositories. pp. 202–211 (2014)

[7] Cheng, B.H., Atlee, J.M.: Research directions in requirements engineering. Future of Software Engineering (FOSE'07) pp. 285–303 (2007)

[8] Pohl, K.: Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated (2010)

[9] Wiegers, K., Beatty, J.: Software requirements. Pearson Education (2013)

[10] Pohl, K.: The three dimensions of requirements engineering: a framework and its applications. Information systems 19(3), 243–258 (1994)

[11] de Gea, J.M.C., Nicol´as, J., Alem´an, J.L.F., Toval, A., Ebert, C., Vizca´ino, A.: Requirements engineering tools. IEEE software 28(4), 86–91 (2011)

[12] Young, R.R.: The Requirements Engineering Handbook. ARTECH HOUSE, INC. 685 Canton Street Norwood, MA 02062 (2004)

[13] Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. John Wiley & Sons, Chichester, UK (1998)

[14] Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. John Wiley & Sons, New York (1997)

[15] Wiegers, K.E.: Software Requirements, 2nd edn. Microsoft Press, Redmond, WA (2003)

[16] Huizingh, E.K.: "Open innovation: State of the art and future perspectives." Technovation, vol. 31, no. 1, pp. 2-9 (2011)

[17] European Commission: Open Innovation Open Science Open to the World - a vision for Europe. European Commission, Brussels (2016)

[18] Gassmann, O., Enkel, E.: "Towards a Theory of Open Innovation: Three Core Process Archetypes" (2004)

[19] Fitzgerald, B.: "The Transformation of Open Source Software." MIS Quarterly, 30(3), 587-598 (2006)

[20] Osterloh, M., Rota, S.: "Open source software development - Just another case of collective invention?" Research Policy, 36(2), 157-171 (2007)

[21] von Hippel, E., von Krogh, G.: "Open source software and the "private-collective" innovation model: Issues for organization science." Organization Science, 14(2), 209-223 (2003)

[22] Castro Llanos, J.W., Acuña Castillo, S.T.: Differences between traditional and open source development activities. In: Product-Focused Software Process Improvement: 13th International Conference, PROFES 2012, Madrid, Spain, June 13-15, 2012 Proceedings 13. pp. 131–144. Springer (2012)

[23] Gandomani, T.J., Zulzalil, H., Ghani, A.A.A., Sultan, A.B.M.: A systematic literature review on relationship between agile methods and open source software development methodology. arXiv preprint arXiv:1302.2748 (2013)

[24] Franco-Bedoya, O., Ameller, D., Costal, D., Franch, X.: Open source software ecosystems: A systematic mapping. Information and software technology 91, 160–185 (2017)

[25] Kiran, H.M., Ali, Z.: Requirement elicitation techniques for open source systems: a review. International Journal of Advanced Computer Science and Applications 9(1) (2018)

[26] Kaur, R., Chahal, K.K., Saini, M.: Understanding community participation and engagement in open source software projects: A systematic mapping study. journal of king saud university-computer and information sciences 34(7), 4607–4625 (2022)

[27] Kitchenham, B., Charters, S., et al.: Guidelines for performing systematic literature reviews in software engineering. Tech. Rep. EBSE-2007-01, Keele Durham Univ. (2007)

[28] Kitchenham, B.A., Madeyski, L., Budgen, D.: SEGRESS: Software engineering guidelines for reporting secondary studies. IEEE Transactions on Software Engineering 49(3), 1273–1298 (2022)

[29] Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology 64, 1–18 (2015)

[30] Wohlin, C., Runeson, P., H¨ost, M., Ohlsson, M.C., Regnell, B., Wessl´en, A.: Experimentation in software engineering. Springer Science & Business Media (2012)

APPENDIX A - LIST OF INCLUDED STUDIES

[SP1] Pérez -Verdejo, J.M., Sanchez-Garcia, J., Ocharán-Hernández, J.O., Mezura-Montes, E., Cortés-Verdín, K.: Requirements and GitHub issues: An automated approach for quality requirements classification. Programming and Computer Software 47, 704–721 (2021)

[SP2] Robinson, M., Sarkani, S., Mazzuchi, T.: Network structure and requirements crowdsourcing for OSS projects. Requirements Engineering 26, 509–534 (2021)

[SP3] Linåker, J., Regnell, B., Damian, D.: A method for analyzing stakeholders' influence on an open source software ecosystem's requirements engineering process. Requirements Engineering 25, 115–130 (2020)

[SP4] Samer, R., Felfernig, A., Stettinger, M.: Towards issue recommendation for open source communities. In: IEEE/WIC/ACM International Conference on Web Intelligence. p. 164–171. WI '19, Association for Computing Machinery, New York, NY, USA (2019)

[SP5] Bhowmik, T., Do, A.Q.: Refinement and resolution of just-in-time requirements in open source software and a closer look into non-functional requirements. Journal of Industrial Information Integration 14, 24–33 (2019)

[SP6] Wang, W., Mahakala, K.R., Gupta, A., Hussein, N., Wang, Y.: A linear classifier based approach for identifying security requirements in open source software development. Journal of Industrial Information Integration 14, 34–40 (2019)

[SP7] Felfernig, A., Stettinger, M., Atas, M., Samer, R., Nerlich, J., Scholz, S., Tiihonen, J., Raatikainen, M.: Towards utility-based prioritization of requirements in open source environments. In: 2018 IEEE 26th International Requirements Engineering Conference (RE). pp. 406–411 (2018)

[SP8] Munir, H., Linåker, J., Wnuk, K., Runeson, P., Regnell, B.: Open innovation using open source tools: a case study at Sony Mobile. Empirical Software Engineering 23, 186–223 (2018)

[SP9] Gopal, D., Lyytinen, K.: Effects of social structures in requirements quality of open source software project development. In: ICIS 2017 Proceedings. pp. 406–411 (2017)

[SP10] Xiao, X., Lindberg, A., Hansen, S., Lyytinen, K.: "Computing" requirements for open source software: A distributed cognitive approach. Journal of the Association for Information Systems 19(12), 1217–1252 (2018)

[SP11] Llerena, L., Rodríguez, N., Castro, J.W., Acuña, S.T.: How to incorporate a usability technique in the open source software development process. In: Pereira, O.M. (ed.) The 30th International Conference on Software Engineering and Knowledge Engineering, Hotel Pullman, Redwood City, California, USA, July 1-3, 2018. pp. 182–181. KSI Research Inc. and Knowledge Systems Institute Graduate School (2018)

[SP12] Do, A.Q., Bhowmik, T.: Refinement and resolution of just-in-time requirements in open source software: A case study. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). pp. 407–410 (2017)

[SP13] Linåker, J., Wnuk, K.: Requirements analysis and management for benefiting openness. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW). IEEE (sep 2016)

[SP14] Kuriakose, J., Parsons, J.: How do open source software (OSS) developers practice and perceive requirements engineering? An empirical study. In: 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE). pp. 49–56 (2015)

[SP15] Gopal, D., Lindberg, A., Lyytinen, K.: Attributes of open source software requirements – the effect of the external environment and internal social structure. In: 2016 49th Hawaii International Conference on System Sciences (HICSS). pp. 4982–4991 (2016)

[SP16] Gopal, D.: Effect of social networks on requirements engineering in open source projects. In: 22nd Americas Conference on Information Systems, AMCIS 2016, San Diego, CA, USA, August 11-14, 2016. Association for Information Systems (2016)

[SP17] Neulinger, K., Hannemann, A., Klamma, R., Jarke, M.: A longitudinal study of community-oriented open source software development. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) Advanced Information Systems Engineering. pp. 509–523. Springer International Publishing, Cham (2016)

[SP18] Linåker, J., Rempel, P., Regnell, B., M¨ader, P.: How firms adapt and interact in open source ecosystems: Analyzing stakeholder influence and collaboration patterns. In: Daneva, M., Pastor, O. (eds.) Requirements Engineering: Foundation for Software Quality. pp. 63–81. Springer International Publishing, Cham (2016)

[SP19] Bhowmik, T., Reddivari, S.: Resolution trend of just-in-time requirements in open source software development. In: 2015 IEEE Workshop on Just-In-Time Requirements Engineering (JITRE). pp.17–20 (2015)

[SP20] Bhowmik, T., Niu, N., Singhania, P., Wang, W.: On the role of structural holes in requirements identification: An exploratory study on open-source software development. ACM Trans. Manage. Inf. Syst. 6(3) (sep 2015)

[SP21] Wnuk, K., Pfahl, D., Callele, D., Karlsson, E.A.: How can open source software development help requirements management gain the potential of open innovation: An exploratory study. In: Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 271–279 (2012)

[SP22] Vlas, R., Vlas, C.: A requirements-based analysis of success in open- source software development projects. In: AMCIS 2011 Proceedings. pp. 1–10 (2011)

[SP23] Noll, J.: Requirements acquisition in open source development: Firefox 2.0. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (eds.) Open Source Development, Communities and Quality. pp.69–79. Springer US, Boston, MA (2008)

[SP24] Scacchi, W., Jensen, C., Noll, J., Elliott, M.S.: Multimodal modeling, analysis, and validation of open source software development processes. International Journal of Information Technology and Web Engineering 1(3), 49–63 (2006)

[SP25] Scacchi, W.: Understanding the requirements for developing open source software systems. IEE Proceedings - Software 149(1), 24–39 (2002)

[SP26] Tizard, J., Devine, P., Wang, H., Blincoe, K.: A software requirements ecosystem: Linking forum, issue tracker, and FAQs for requirements management. IEEE Transactions on Software Engineering pp. 1–13 (2022)

[SP27] Wang, W., Hussein, N., Gupta, A., Wang, Y.: A regression model based approach for identifying security requirements in open source software development. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). pp. 443–446 (2017)

[SP28] Morales-Ramirez, I., Vergne, M., Morandini, M., Perini, A., Susi, A.: Exploiting online discussions in collaborative distributed requirements engineering. In: Castro, J., Filho, G.A.C., Liaskos, S. (eds.) Proceedings of the Eighth International i*Workshop, iStar 2015, in conjunction with the 23rd International Requirements Engineering Conference (RE 2015), Ottawa, Canada, August 24-25, 2015. CEUR Workshop Proceedings, vol. 1402, pp. 7–12. CEUR-WS.org (2015)

[SP29] Vergne, M., Susi, A.: Expert finding using markov networks in open source communities. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) Advanced Information Systems Engineering. pp. 196–210. Springer International Publishing, Cham (2014)

[SP30] Alspaugh, T.A., Scacchi, W.: Ongoing software development without classical requirements. In: 2013 IEEE 21st International Requirements Engineering Conference (RE). pp. 165–174. IEEE Computer Society, Los Alamitos, CA, USA (jul 2013)

[SP31] Ernst, N.A., Murphy, G.C.: Case studies in just-in-time requirements analysis. In: 2012 Second IEEE International Workshop on Empirical Requirements Engineering (EmpiRE). pp. 25–32 (2012)

[SP32] Laurent, P., Cleland-Huang, J.: Lessons learned from open source projects for facilitating online requirements processes. In: Glinz, M., Heymans, P. (eds.) Requirements Engineering: Foundation for Software Quality. pp. 240–255. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[SP33] Shi, L., Chen, C., Wang, Q., Boehm, B.: Is it a new feature or simply "don't know yet"?: On automated redundant OSS feature requests identification. In: 2016 IEEE 24th International Requirements Engineering Conference (RE). pp. 377–382. IEEE Computer Society, Los Alamitos, CA, USA (sep 2016)

[SP34] Scacchi, W.: Understanding requirements for open source software. In: Design Requirements Engineering: A Ten-Year Perspective: Design Requirements Workshop, Cleveland, OH, USA, June 3-6, 2007, Revised and Invited Papers. pp. 467–494. Springer (2009)

[SP35] Salo, R., Poranen, T., Zhang, Z.: Requirements management in GitHub with a lean approach. In: SPLST. pp. 164–178 (2015)

[SP36] Gill, K.D., Raza, A., Zaidi, A.M., Kiani, M.M.: Semi-automation for ambiguity resolution in open source software requirements. In: 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE). pp. 1–6. IEEE (2014)

[SP37] Heck, P., Zaidman, A.: Horizontal traceability for just-in-time requirements: the case for open source feature requests. Journal of Software: Evolution and Process 26(12), 1280–1296 (2014)

[SP38] Puhlf ̈urß, T., Montgomery, L., Maalej, W.: An exploratory study of documentation strategies for product features in popular github projects. In: 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 379–383. IEEE (2022)

[SP39] Johansson, B., de Carvalho, R.A.: Management of requirements in ERP development: a comparison between proprietary and open source ERP. In: Proceedings of the 2009 ACM symposium on Applied Computing. pp. 1605–1609 (2009)

[SP40] McGrath, O.G.: Balancing act: community and local requirements in an open source development process. In: Proceedings of the 34th annual ACM SIGUCCS fall conference: expanding the boundaries. pp. 240–244 (2006)

[SP41] Bastani, B.: A requirements analysis framework for open systems requirements engineering. ACM SIGSOFT Software Engineering Notes 32(2), 1–19 (2007)

[SP42] Noll, J., Liu, W.M.: Requirements elicitation in open source software development: a case study. In: Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. pp. 35–40 (2010)

[SP43] Vlas, R.E., Robinson, W.N.: Two rule-based natural language strategies for requirements discovery and classification in open source software development projects. Journal of management information systems 28(4), 11–38 (2012)