Tampere University

Charles Nebo

REQUIREMENT ELICITATION TECHNIQUES IN STUDENT PROJECTS

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

ABSTRACT

Charles Nebo: Requirement Elicitation Techniques in Student Projects
M. Sc. Thesis
Tampere University
Master's Degree Programme in Software Development
June 2023

The growing importance of software and modern technologies, along with the increasing emphasis on software quality, has led to the widespread use of software products worldwide. In response, the software engineering curriculum has been evolving to prepare students for future careers in the software industry. This study aims to provide valuable insights into requirement elicitation techniques in student projects and discuss the best practices for selecting suitable techniques, specifically focusing on the challenges of eliciting requirements and the techniques most appropriate for student projects.

To achieve these goals, this research thoroughly examines the challenges associated with requirements elicitation techniques in student projects through a comprehensive literature review and case studies. Additionally, it identifies and analyses requirements elicitation techniques that are well-suited for student projects. The data for this study is collected from the documentation of previous student projects conducted as part of the software project management course at Tampere University in Finland.

The findings of this research reveal four categories of challenges related to requirements elicitation in student projects: communication-related challenges, stakeholder-related challenges, challenges associated with developers, and factors related to the personalities of the participants involved in the elicitation process. The study emphasises the crucial role of effective communication among project teams in ensuring successful requirements elicitation. It also highlights how project teams select elicitation techniques based on their compatibility with the employed development process, and they may even combine multiple techniques depending on the nature and complexity of the project.

The contribution of this research lies in providing a comprehensive overview of the current state of requirements elicitation techniques, including their applicability, strengths, weaknesses, and the current state of practice. The implications of this study extend to students and the development community, as it enhances understanding of the challenges faced in gathering and managing requirements.

Key words and terms: requirements elicitation, techniques, student projects, case study.

Contents

1 INTRODUCTION 1

2 SOFTWARE ENGINEERING PROJECT COURSES 4

2.1 Active Classroom Learning Method 4

2.2 Project-Based Learning 4

2.3 Challenges in Implementing the Engineering Project Courses 6

3 REQUIREMENTS ELICITATION TECHNIQUES 8

3.1 Conversational Methods 9

3.1.1 Interview 9

3.1.2 Questionnaires or Surveys 9

3.1.3 Brainstorming 10

3.1.4 Focus Group or Workshop 10

3.2 Observational Methods 11

3.2.1 Social Analysis or Ethnography 12

3.3 Analytic Methods 12

3.3.1 Requirements Reuse 13

3.3.2 Documentation Studies 13

3.3.3 Laddering Method 14

3.3.4 Repertory Grid 14

3.4 Synthetic Methods 16

3.4.1 Scenarios 16

3.4.2 Use Cases 17

3.4.3 Prototypes 18

3.4.4 Joint Application Development (JAD) 18

3.4.5 Contextual Inquiry 19

4 CHALLENGES IN STUDENT PROJECTS 24

5 CHALLENGES OF ELICITING REQUIREMENTS IN STUDENT PROJECTS 26

6 CASE STUDY: ELICITING REQUIREMENTS FOR THE eTANDEM WEB APP 29

6.1 Context 29

6.2 Data Collection and Analysis 29

6.3 Method and Process 30

6.4 Project Management 31

6.5 Managing and Leading Project Teams 33

6.6 Requirements Elicitation in Student Projects 34

7 RESULT 37

7.1 Challenges Students Faced in Eliciting Requirements. 38

7.2 Correlation with Project Reports 39

8 CONCLUSION 42

REFERENCES 44

1 INTRODUCTION

The introduction provides a comprehensive overview of three main themes: software engineering education, student projects, and requirements engineering. Software engineering education is vital in preparing students to become successful software engineers. It aims to equip students, typically studying computer science or information technology, with the necessary principles, skills, and expertise (Ouhbi & Pombo, 2020). The educational background in software engineering significantly influences the success of software engineers (Cico et al., 2021; Sivaloganathan, 2004). Modern software engineering curricula have evolved to meet industry demand, incorporating various topics such as programming, modelling, and requirements engineering (Cico et al., 2021; Marques et al., 2014). Software engineering education encompasses hard and soft skills essential for a successful career. Hard skills refer to the technical abilities acquired through formal education, while soft skills encompass communication and project management abilities crucial for the workplace (Morais et al., 2021). Recognising soft skills in software development, the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) strongly recommend their integration into software engineering education (ACM, 2013).

However, software engineering education presents challenges for both educators and students. The abstraction involved in modelling and requirements engineering and the technical nature of requirements elicitation demands guidance and support for students (Knobloch et al., 2018; Morais et al., 2021). Furthermore, the software industry's dynamic nature necessitates continuous curriculum updates to align with emerging trends and technologies (Cico et al., 2021).

Student projects are vital to software engineering education, allowing students to apply their theoretical knowledge in real-world scenarios. These projects aim to cultivate the technical and soft skills necessary for future software engineers to thrive in industrial settings. In recent years, there has been a shift in instructional approaches, moving away from traditional classroom models towards project-based and problem-based learning (Yu, 2014).

Initiatives led by associations such as the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) have transformed the educational landscape, emphasising technology, teamwork, and interactive learning experiences (Yu, 2014). These initiatives promote self-paced, place-based, or remote learning environments where students develop critical thinking and cognitive skills (Ouhbi & Pombo, 2020). Through collaborative projects, students learn to work effectively in teams, enhance their problem-solving abilities, and develop communication and project management skills (Morais et al., 2021).

While student projects offer valuable experiential learning opportunities, they challenge instructors and students. Instructors must invest significant time and effort in setting up and facilitating these projects, assuming multiple roles such as mentors, evaluators, and occasional project sponsors (Yu, 2014). On the other hand, students face challenges in delivering functional software within specified timeframes and managing the complexities of software project management (Yu, 2014). The requirements engineering process, in particular, has been identified as problematic in software projects, with issues such as incomplete specifications and requirement changes posing significant hurdles (Mäkiaho et al., 2017).

Requirements engineering is a critical process in software development projects, ensuring that the project's objectives and stakeholder needs are effectively captured and translated into software requirements. It involves eliciting, analysing, documenting, and managing requirements throughout the software development lifecycle (Coughlan & Macredie, 2002; Zhang, 2007). However, the intangible nature of models, the emphasis on programming skills in job interviews, and the wide range of stakeholder requirements present challenges for the widespread adoption of requirements engineering (Jiang et al., 2008).

Scholars and practitioners recognise the importance of applying appropriate requirements engineering practices to enhance the success of software (Glass et al., 1995; Hickey et al., 2003). Various techniques and models have been developed to address diverse problem domains, software project types, and stakeholder needs (Jiang et al., 2004). However, the effective selection and application of these techniques remain a challenge.

In software engineering education, students often encounter difficulties with requirements and programming tasks due to the abstraction involved (Knobloch et al., 2018; Quintanilla Portugal et al., 2016). Requirements elicitation modelling, in particular, presents its challenges, requiring iterations and effective communication to achieve satisfactory solutions (Berre et al., 2018). Programming also poses challenges due to the complex abstraction required for problem interpretation and the systematic application of problem-solving methods (Babb et al., 2014; Canedo et al., 2018).

Consequently, software engineering education prepares students for successful careers as software engineers, encompassing hard and soft skills. Student projects offer valuable experiential learning opportunities, promoting teamwork, problem-solving, and communication skills. However, these projects pose challenges for instructors and students, particularly in engineering. The effective application of requirements engineering techniques and models remains a critical aspect of software development projects, requiring iterative processes, effective communication, and problem-solving skills.

Software projects employ various requirements elicitation techniques, such as conversational, observational, analytic, and synthetic approaches (Zhang, 2007). Students commonly use methods and communication systems to gather requirements, incorporating conversational, observational, analytical, and synthetic methods. Each technique serves specific situations and environments, ensuring a comprehensive understanding of project needs. Analysts must thoroughly comprehend the various tactics involved in the elicitation process (Carrizo et al., 2014; Davis et al., 2006; Zhang, 2007).

This thesis aims to study requirements elicitation techniques in students' projects and discuss the best practice of requirements elicitation technique selection. To achieve this goal, the thesis will tackle the following research questions:

RQ1. What are the challenges of eliciting requirements in student projects?

RQ2. What requirement elicitation method is suitable for student projects?

Research question 1 is addressed by examining various elicitation approaches and requirements activities currently utilised in the requirements elicitation process. This thesis analyses the different elicitation approaches and activities commonly employed in the software industry, as well as opinions from researchers. Research question two aims to identify suitable requirement techniques for student projects.

The remainder of the paper is structured as follows: Chapter 2 describes software engineering project courses, while Chapter 3 describes requirements elicitation techniques. Chapter 4 describes the challenges in student projects. Chapter 5 describes the challenges of eliciting requirements in student projects, while Chapter 6 describes the case study eliciting requirements for the eTandem web app. Chapter 7 presents the case study result. Chapter 8 presents the Conclusion.

2 SOFTWARE ENGINEERING PROJECT COURSES

Learning to design, build, test, and manage software projects is critical to software engineering courses. According to (Valencia et al., 2016), these courses often include software requirements, design, testing, quality assurance, project management, tools, and processes. The traditional teaching model in software engineering education emphasises passive student participation and a lecture-centred approach. Research findings indicate that this conventional method no longer effectively supports student learning, as it relies heavily on transcription, memorisation, and repetition while neglecting the development of critical thinking and active engagement (Major et al., 2001; Morais et al., 2021). Consequently, many students experience academic difficulties, leading to high failure rates. To address these challenges, educators have embraced alternative learning approaches involving students actively in the educational process (Babb et al., 2014). Common obstacles faced by students in software engineering education include challenges in abstract thinking, programming, logical reasoning, and modelling. Students desire to participate in discussions and engage meaningfully with their peers actively.

This chapter discusses the integration of innovative engineering project courses within software engineering education, aiming to shift the pedagogical paradigm from teacher-centred instruction to student-centred learning. By providing an overview of different project types in software engineering education, this chapter highlights the potential benefits of this approach in fostering student engagement and facilitating meaningful learning experiences.

2.1 Active Classroom Learning Method

Active learning strategies play a pivotal role in the proposed engineering project course. This student-centred approach emphasises collaborative learning, critical thinking, and problem-solving as essential components of the educational experience (Babb et al., 2014). Through active participation in hands-on activities and teamwork, students are empowered to develop crucial competencies required in software engineering.

2.2 Project-Based Learning

The engineering project courses adopt project-based learning as their core instructional strategy. Project-based learning (PjBL) encourages students to use their knowledge and abilities in real-world situations. Students gain a more robust comprehension of software engineering topics through practical application of theoretical knowledge and participation in real-world problem-solving activities (Morais et al., 2021).

The PjBL strongly emphasises practical, real-world problem-solving (Valencia et al., 2016). In PjBL, students take on various roles necessary for completing a project, working collaboratively in teams to solve the problem from inception to conclusion. Project-based learning aims to empower students with a sense of control over their education and equip them with the skills required for success in the contemporary workplace. Students may collaborate or work individually depending on the nature of the assignment. According to Morais et al. (2021), the primary aims of project-based learning are to support students in developing flexible knowledge, practical problem-solving skills, educational and collaborative talents, and intrinsic motivation.

Other studies by Marques (2015), Marques et al. (2018), and Marques et al. (2014) characterise project-based learning as a systematic instructional strategy involving students acquiring knowledge and skills through an extended inquiry process rooted in challenging real-world situations. The process is thoughtfully designed around creating meaningful products and engaging activities (Kokotsaki et al., 2016).

Project-based learning aligns with the established curriculum, incorporating the recommended practices for software engineering projects by ACM/IEEE. The following aspects must be considered and integrated into the project:

- Project cycle time provides a year-long timeframe for students to reflect on their prior learning and explore potential solutions.
- Team orientation involves assigning small software teams, typically 5-7 individuals, to specific roles during the project's development.
- Client involvement, ensuring the active participation of a company representative alongside the team leader.
- The project should incorporate a software development methodology that produces observable project deliverables. This allows students to gain valuable experience through the instructional activity. Therefore, projects that rely solely on theoretical aspects, such as developing formal specifications, are unsuitable.
- Evaluation involves assessing project outputs through walkthroughs, interviews, and short experiments to determine the effectiveness and limitations of the deliverables (Dyba et al., 2014).

- Assessment encompasses evaluating the project's effectiveness in terms of the software engineering methodologies and procedures employed, with student reflection playing a crucial role, even without a functioning system.

The engineering project course offers several distinct advantages over traditional teaching methods. Firstly, it promotes active student engagement and ownership of the learning process. Through hands-on project work, students comprehensively understand software engineering principles and their practical implications. Additionally, the course fosters teamwork, communication skills, critical thinking, and problem-solving abilities, which are highly sought-after skills in the software engineering profession (Major et al., 2001).

2.3 Challenges in Implementing the Engineering Project Courses

Although project-based learning offers numerous benefits in engineering education, its implementation is challenging. These challenges may arise from various factors, including sufficient resources to support the project-based approach (Aldabbus, 2018).

One primary challenge is the allocation of appropriate resources. Project-based learning often requires additional materials, equipment, and technologies to support students' practical engagement with real-world problems. Institutions must ensure they have the resources to facilitate project work, including access to laboratories, software tools, and appropriate industry-standard equipment. Limited availability or inadequate allocation of resources can hinder the effective implementation of project-based learning, limiting students' opportunities for hands-on experience and skill development (Clear et al., 2001; García-López et al., 2020; Mann et al., 2004).

Another challenge involves managing time and coordinating schedules. Project-based learning typically involves complex and multifaceted tasks that require substantial time commitments. Students may need to balance their project work with other academic responsibilities and extracurricular activities, which can lead to time management difficulties. Coordinating team meetings, arranging consultations with instructors, and aligning project schedules with external stakeholders, such as industry partners or clients, can also present logistical challenges (Herbert, 2018; Mann et al., 2011; Zhang et al., 2010). Additionally, assessing and evaluating student performance in project-based learning can be challenging. Traditional assessment methods may not align with the dynamic nature of the project work, which often involves iterative processes and evolving solutions. Designing appropriate assessment criteria and evaluating student outcomes fairly and consistently can be complex. It requires careful consideration of individual and team contributions' ability to assess the development of critical thinking, problem-solving, and collaboration skills (Kaul et al., 2015; Majanoja et al., 2018).

Moreover, supporting and guiding students throughout the project process can be demanding for instructors. The role of the instructor in project-based learning shifts from being a primary source of information to that of a facilitator and mentor. Instructors must provide guidance, feedback, and scaffolding to ensure students stay on track, meet project objectives, and develop the necessary skills. This requires significant effort from instructors and expertise in project management and effective facilitation techniques.

Finally, fostering effective teamwork and collaboration among students can be a challenge. Project-based learning often involves group work, requiring students to collaborate, communicate, and resolve conflicts effectively. Building cohesive and productive teams can be challenging, especially when students come from diverse backgrounds with varying experience levels and expertise. Instructors must provide support and establish clear expectations for teamwork while assessing any issues arising during the project (Behdinan et al., 2015).

It is necessary to take a complete approach to address these issues, which includes careful planning, adequate resource allocation, ongoing support for students and instructors, and constant review and enhancements of the project-based learning implementation. By proactively addressing these challenges, institutions can maximise the benefits of project-based learning and provide students with valuable experiences that prepare them for real-world engineering practices.

3 REQUIREMENTS ELICITATION TECHNIQUES

The concept of requirement elicitation techniques pertains to the methods employed by analysts to ascertain the needs of customers and users, thus enabling the development of software systems that effectively fulfil those requirements. The primary goal of requirement elicitation is to comprehensively understand the problem domain, which is the foundation for constructing a requirement model. By engaging in this process, analysts aim to resolve conflicting requirements and enhance the utility and stability of software systems. It is important to note that requirement elicitation is an iterative process that necessitates recognising and involving stakeholders, the critical actors in the software development process (Zhang, 2007).

Requirement elicitation plays a crucial role in the software development lifecycle, as it sets the stage for the subsequent phases of system design, implementation, and testing. Effective elicitation techniques enable analysts to gather accurate and relevant information about the desired functionality and constraints of the software system. The use of this data during the development process allows for informed decisions and guarantees that the finished product meets the needs and expectations of all parties involved (Bahurmuz et al., 2021; Carrizo et al., 2014; Pacheco et al., 2018; Sabariah et al., 2018).

The iterative nature of requirement elicitation implies that the process involves continuous refinement and adjustment. As the analysts gain a deeper understanding of the problem domain and interact with stakeholders, they can identify and resolve any conflicts or inconsistencies in the requirements. This iterative method enables the discovery of new requirements that could have gone unnoticed at first and changing current requirements in the light of changing stakeholder demands and priorities (Davis et al., 2006; Zhang, 2007).

Analysts employ various techniques to facilitate the process, recognising the importance of effective requirement elicitation. These techniques range from conversational and observational approaches to analytical and synthetic methods. Each technique offers unique advantages and is suitable for specific situations and contexts. The selection of the appropriate elicitation technique depends on factors such as the nature of the project, the requirements complexity, and the stakeholders' availability and expertise. (Carrizo et al., 2014).

3.1 Conversational Methods

A conversational method is a form of verbal communication between stakeholders and the development team. It provides a natural and direct means of expressing needs and ideas through face-to-face interactions. This method encompasses various activities, including interviews, workshops, brainstorming sessions, and focused groups, all aimed at extracting relevant information related to the problem domain. The primary objective is to facilitate effective communication and exchange of information regarding the project's requirements (Yousuf et al., 2015; Zhang, 2007).

Conversational methods effectively elicit requirements, mainly when direct interaction and personal engagement are crucial. However, it is essential to note that this approach can be labour-intensive due to the extensive documentation required. It involves setting up meetings, analysing documents, and performing ad hoc tasks to ensure effective communication and understanding between stakeholders and the development team. Overall, the conversational method serves as a valuable approach for requirement elicitation, allowing for direct engagement and real-time collaboration between stakeholders and the development team. While it may require significant effort in documentation and coordination, its effectiveness in capturing and exchanging relevant information makes it a preferred choice in many projects (Coughlan et al., 2002; Mushtaq, 2016). This conversational method includes interviews, questionnaires or surveys, brainstorming, focus groups or workshops.

3.1.1 Interview

Interviews are widely used in requirement elicitation and involve direct face-to-face interaction to collect user data. Analysts conduct interviews with customers to gather qualitative information that influences user requirements. Open-ended interviews, without predefined questions, or structured interviews, with predetermined questions, are utilised. Interviews are highly effective in traditional settings or global software development, enabling analysts to obtain precise information about requirements within the defined scope (Maiden et al., 1996).

3.1.2 Questionnaires or Surveys

Questionnaires involve written questions presented to a target group of respondents to collect data. They are suitable when time is a constraint, allowing the collecting a large amount of data from multiple respondents with less effort and expense. However, before distributing questionnaires, analysts must clearly understand the target audience and domain concepts. Questionnaires work well for research topics in system development but require a more comprehensive understanding of the current scope. Misinterpreting questionnaire results are a concern, emphasising the need for straightforward, concise questions that provide adequate information within the defined scope (Yousuf et al., 2015; Zowghi et al., 2005). Recent research suggests combining questionnaires with other methods to enhance their effectiveness in the IT sector (Pacheco et al., 2018) and (Arif et al., 2009).

3.1.3 Brainstorming

Brainstorming is a creative technique to generate ideas and potential solutions to a problem. Participants voluntarily contribute ideas in a group setting to overcome inhibitions and foster innovation (Paetsch et al., 2003; Yousuf et al., 2015). The brainstorming process involves a generation phase where ideas are collected and discussed. The emphasis is on expanding and exploring viewpoints rather than critiquing or evaluating them. Participants are encouraged to freely express as many novel and unplanned ideas as possible, fostering a mindset of open thinking. User data can be collected through mind mapping, which involves visually organising thoughts on paper, or through group brainstorming sessions facilitated by a moderator. Brainstorming has gained popularity in the IT industry for requirements elicitation, with studies suggesting its effectiveness in enhancing workplace ideas and improving the quality of group-generated plans (Mushtaq, 2016).

3.1.4 Focus Group or Workshop

Focus groups are qualitative research methods that involve gathering a small group of individuals with common characteristics or interests related to the software system under consideration. A moderator leads the group in open-ended discussions and exchanges, exploring participants' opinions, attitudes, preferences, and experiences regarding the software system and its features (Kasirun et al., 2008). Extended focus groups have addressed communication challenges between stakeholders and analysts in field studies, enhancing the requirements elicitation process (Pacheco et al., 2018) and (Fernandes, 2014; Pitula, 2011; Amin et al., 2021). Empirical results from these studies contribute valuable insights and lessons learned that consolidate knowledge and guide practitioners. Here are data collection methods in the conversation category and how they are currently used in Table 1.

Table 1. Summary of Conversational Methods.

Method Name	Data Collection Methods	State of Practice
Interview (Pacheco et al., 2018; Palomares et al., 2021).	Face-to-face interaction, Open-ended interview.	The interview is widely used in the industry; it is popular and most effective and can be used with other methods.
Questionnaires or Surveys (Ang et al., 2011) (Zowghi et al., 2005).	List of questions to respondents.	They were frequently used. Researchers advised using questionnaires with other methods.
Brainstorming (Paetsch et al., 2003).	The facilitator leads mind mapping and group brainstorming sessions.	Popular, effective, and widely used.
Focus group or Workshop. (Pacheco et al., 2018).	Meetings.	Popular, effective, and widely used.

3.2 Observational Methods

The observation method involves the development team's self-study of the domain environment to understand the work context and human activities comprehensively. Sometimes, it can be challenging for the development team to articulate the intricacies of work processes within a given environment. Thus, the observation method is a valuable tool for immersing the development team in the situational context and obtaining evidence that aids in comprehending the work patterns (Zowghi et al., 2005).

The observation method is beneficial when stakeholders require assistance articulating their needs, especially when the development team strives to understand the work context better. It proves to be a practical approach for comprehending the work environment and the tasks performed by the users. Furthermore, this method enhances the analysts' familiarity with the organisational culture and the preferred work style of individuals within the organisation. Applying the observation method allows analysts to directly assess the work domain and gain firsthand insights into how work is executed (Paetsch et al., 2003). However, it is essential to acknowledge a potential disadvantage associated with the observation method. Stakeholders may quickly feel offended or change their behaviour based on the attention they receive from the development team. This sensitivity to observation can impact the authenticity and accuracy of the observed work activities (Babb et al., 2014; Yousuf et al., 2015).

3.2.1 Social Analysis or Ethnography

The social analysis requirement method incorporates social and human factors into the software development process. It aims to understand the broader social context and the interaction between individuals or groups, which can significantly impact the success of software projects (Maiden et al., 1996; Zhang, 2007). The context of the social field encompasses the real world where the application will be used, such as the homes, public spaces and end-user environments (Zhang, 2017).

The data collection process involves researchers immersing themselves in the social context, spending time with participants, and engaging in meaningful interactions. Researchers observe users' routines, work processes, and challenges to contextualise their requirements and preferences. The researcher can collect data through various techniques, such as interviews, focus groups, observations, surveys, and cultural probes. Researchers use open-ended questions during interviews and focus groups to encourage participants to share their thoughts freely. Additionally, researchers carefully analyse the data collected from cultural probes, gaining insights into users' perceptions of the software in their own words and expressions (Fuentes-Fernández et al., 2010).

Social analysis states of practice are gaining recognition and importance in software development. However, it is not yet universally adopted in the industry. Many organisations still primarily focus on traditional requirement engineering techniques and may overlook the significance of the human element. However, academic research and industry case studies continually demonstrate the benefits of incorporating social analysis into software development (Fuentes-Fernández et al., 2010). Here are data collection methods in the observation category and how they are currently used in Table 2.

Table 2. Summary of Observation Methods.

Method Name	Data Collection Methods	State of Practice
Social Analysis or Ethnography (Fuentes-Fernández et al., 2010).	Being in the workplace and observing users' routines, work processes	Social analysis has not yet been adopted in the software industry. Professionals continue to use the traditional methods.

3.3 Analytic Methods

The analytical method serves as an approach for extracting requirements from pre-existing documentation, drawing upon two primary sources: requirement reuse and documentation studies. This method proves valuable in capturing knowledge that domain experts may need to express explicitly. By thoroughly examining and analysing existing documentation, developers can gain insight into the required tasks embedded within workflows and product features (Zhang, 2007).

It is important to note that one area for improvement of the analytical method is its inability to capture requirements from users and customers directly. Instead, it examines documentation to uncover essential information about the application domain. This approach effectively elicits knowledge from domain requirements, including legacy systems and specifications. The workflows and product features documented within the existing materials often contain the necessary information to be leveraged to extract relevant domain knowledge.

In addition to the analytical method, several related techniques fall under its umbrella. These include requirements reuse, documentation studies, content analysis, card sorting, and repertory grids. Each of these techniques contributes to the overall analytical process, facilitating the extraction of valuable insights from existing documentation (Yousuf et al., 2015; Zhang, 2007).

3.3.1 Requirements Reuse

Requirement reuse involves creating software systems by leveraging existing artefacts rather than starting from scratch (Yousuf et al., 2015; Zhang, 2017). This can range from reusing requirements documented in previous specifications to utilising templates in catalogues adapted for each new project. Requirement reuse can be achieved through various methods such as structuring, matching, test-based, and scenario-oriented reuse—the structuring approach stores requirements in a well-organized structure for easy retrieval. Grouping and categorisation facilitate searching and identifying specific needs. However, most software requirement reuse approaches are validated primarily in the industry for text-based methods, with a low percentage of academic publications on this topic (Irshad et al., 2018).

3.3.2 Documentation Studies

Documentation studies systematically analyse existing printed and electronic documents to elicit requirements (Bowen, 2009). This method gathers requirements during the elicitation process to extract critical requirements relevant to the current project. Quantitative and qualitative strategies can be employed to collect user data through document analysis. Quantitative research relies on content analysis and statistical interpretation of data, while qualitative methods are favoured by researchers who actively participate in field studies. Triangulation, involving the verification of data accuracy from multiple sources, is another technique used. In industrial practice, project managers commonly employ document analysis to plan project strategies and gain a comprehensive understanding of project procedures. By analysing different documents, project managers can identify essential project requirements (Project Management Institute, 2013).

3.3.3 Laddering Method

The laddering technique, introduced by psychologists in the 1960s, is a valuable method for comprehending individuals' core values and beliefs (Corbridge et al., 1994). This technique is extensively employed in market research and knowledge acquisition, enabling the establishment of an individual's core set of constructs that shape their worldview (Corbridge et al., 1994). Derived from the repertory grids method, the laddering technique utilises structured interviews to establish attribute, consequence, and core value hierarchy based on the means-end theory that identifies product attributes and their associated effects (Stewart et al., 1998).

One of the primary advantages of the laddering technique is its ability to provide insights into stakeholders' core values and beliefs, facilitating a comprehensive understanding of their perspectives. It is instrumental in market research, allowing researchers to understand consumer preferences and decision-making processes deeply. Additionally, the laddering technique establishes a hierarchical structure by identifying attributes, consequences, and core values, aiding in requirements prioritisation. However, the laddering approach faces certain limitations. When dealing with many requirements, the method may become more complex, potentially resulting in difficulties in managing and prioritising multiple needs. Furthermore, like any other requirement elicitation method, the laddering technique is not immune to challenges related to requirements, communication, stakeholders, and developers, which may impact the quality and success of software projects.

3.3.4 Repertory Grid

The repertory grid method involves constructing an attribute matrix for each entity and soliciting attributes from stakeholders that apply to each entity, along with values for the cells inside each entity (Nuseibeh et al., 2000). Domain experts predominantly employ

this method, and it proves helpful in identifying agreements and resolving conflicts between stakeholder groups. The grid approach aims to maintain consistency in analysing all aspects and qualities while facilitating the resolution of disagreements among stakeholders. The use of nominal values, such as "yes" or "no," contributes to the numerical values, as suggested by (Grunert et al., 2005). However, representing domains consisting of nominal categories can challenge the matrix representation within the operating system. The data collection process in the repertory grid method entails building a matrix with rows and columns. Each row represents system entities and desired qualities, while the columns represent rankings based on stakeholder input. This approach is particularly valuable for identifying consensus and disagreement among stakeholder groups (Ylönen, 2021). According to Laplante (2017) and Nuseibeh et al. (2000), a notable strength of the repertory grid method lies in its ability to subject numerical values to various statistical techniques, including correlation and principal component analysis. Additionally, software for statistical analysis and repertory grid approaches is readily available. The repertory grid method is valuable in resolving conflicts and identifying agreements or disagreements between stakeholder groups in the early phases of software development. However, the author of this study did not find any source material where repertory grid methods were explicitly used for requirements elicitation for system development.
Regarding industrial application, Pacheco et al. (2018) state that the repertory grid method is one of the most widely used techniques in the cognitive category. However, their report does not explicitly mention its application in the context of requirement elicitation. Although no sources were found where the repertory grid method was used for requirements elicitation, Laplante (2017) suggests its usefulness in identifying different viewpoints from diverse stakeholder groups. Furthermore, it can assist in weighing various solution options during system upgrades and reconciling opposing views on specific functionality. Here are data collection methods in the analytic category and how they are currently used in Table 3.

Table 3. Summary of Analytic Methods.

Method Name	Data Collection Methods	State of Practice
Requirements reuse (Irshad et al., 2018).	Structuring, Matching, Test-based, Scenario-oriented reuse	All the software reuse approaches are not validated in the industry except the test-based method.
Document studies (Bowen, 2009). (Project Management Institute, 2013).	Reading and reviewing documentation.	Document analysis is popular among project managers to plan current project strategies or to review and extract requirement information from previous projects.
Laddering (Corbridge et al., 1994). (Maiden et al., 1996; Stewart et al., 1998).	Structured Interviews.	This method is widely used in market research as a viewpoint on consumer choices.
Repertory Grid (Laplante, 2017).	Matrix data collection	Researchers believe the repertory grid is a valuable technique, but the author of this study did not find where or how it was used in RE.

3.4 Synthetic Methods

Different situations may require various elicitation techniques to capture requirements, as Maiden et al. (1996) and Yousuf et al. (2015) suggested. Each method possesses its strengths in capturing specific types of requirements. Consequently, analysts may need to employ multiple methods in different sessions to explore essential aspects and develop a comprehensive understanding of the application domain. Synthetic methods offer an alternative by integrating conversation, observation, and analysis into a unified process, facilitating the achievement of a shared product vision by analysts and stakeholders. Examples of synthetic methods include Scenarios, Passive Storyboards, prototyping, interactive Storyboards, Joint Application Development (JAD) or Rapid Application Development (RAD), and Contextual Inquiry (Maiden et al., 1996; Yousuf et al., 2015).

3.4.1 Scenarios

Scenarios depict user interactions with the system, illustrating how they collaborate to achieve system-level functionality (Jaramillo, 2015). They are particularly valuable when describing the system from a user perspective and can be applied once initial requirements have been elicited. Writing scenarios in plain language requires a basic understanding of the system's operations and user interactions. Methods can also generate test cases and validate the requirements (Davis et al., 1992).

The studies by Davis (1992) highlight the effectiveness of scenarios in traditional, global, and agile software development. They propose methods as a primary technique for requirement construction and suggest combining them with prototyping to reduce errors during requirement collection. While prototyping is a practical approach, reports that development costs often pose a significant obstacle, leading industry practitioners to opt for prototypes over scenarios to mitigate expenses.

By providing action-oriented representations of real-world situations, scenarios have been consistently recognised as a valuable source of requirements and a commonly used form of knowledge representation in industrial applications. The studies propose a scenario-based approach for learning future production in tangible industry 4.0. They argue that scenarios facilitating active learning through real-world problem-solving can serve as a starting point for students to immerse themselves in authentic problem-solving processes, applying their social skills in a safe environment.

3.4.2 Use Cases

Use cases illustrate user interactions and the system, focusing primarily on what the system does for the user. They enable analysts to elicit and document customer requirements by identifying and describing different use cases for each actor involved with the system (Regnell et al., 1995). Use cases have traditionally been part of the Unified Modelling Language (UML), supporting an iterative software development process that allows for early user feedback. A use case specifies how the system assists a user in achieving a specific goal and outlines all the outcomes required to reach that goal. By focusing on the system's functional behaviour and its users, use cases are created from the user's perspective, avoiding discussions about the system's internal workings (Project Management Institute, 2013).

Understanding the system's functional behaviour aids in comprehending its connections to different use cases (Leffingwell et al., 2000). Use case diagrams, a popular technique for collecting user data during software development, serve as a means of communication between all parties involved by simulating user interactions with the system through actors. They effectively present stakeholders' viewpoints on the system being developed for the client. According to a study, case diagrams are easier to understand than class diagrams and effectively communicate the purpose of the system to stakeholders (Laporti et al., 2009).

A recent study emphasises the maturity and widespread use of use cases in industrial practice contexts. Use cases are considered a primary tool for precisely defining the context and requirements of the client, and they serve as a primary technique for collecting user data. They also prove valuable in eliciting security requirements, enhancing the effectiveness of these requirements when combined with other methods (Laporte et al., 2009; Pacheco et al., 2018).

3.4.3 Prototypes

Prototypes are early samples of a concept, serving as iterative versions of the product that gather customer requirements and continually improve based on user feedback. They are particularly useful when users have limited knowledge about their needs and stakeholders require early responses (Davis, 1992).

Recent research demonstrates the importance of prototypes for the success of many software projects despite potential deviations from traditional project life cycle plans. Prototyping addresses a common problem during software development, especially in the requirements elicitation phase.

Prototyping explores, experiments, and evolves the system through user data collection (Jensen et al., 2018). The exploration strategy aims to elicit or clarify user requirements, enabling developers to determine users' work tasks and identify potential problems with the new system. This approach provides engineers with a comprehensive understanding of users' work tasks. The exploratory approach can be successful through two critical processes: rapid throwaway prototyping and the spiral model (Boehm, 1991). The fast throwaway prototyping model involves generating prototypes, obtaining customer feedback, and validating system requirements and functionalities. In contrast, rapid prototyping consists in discarding the prototype as it does not yield the desired result. Experimental prototypes are used to assess the feasibility of proposed solutions, creating partial or complete functional simulations demonstrating different system aspects. Mock-ups serve as examples of experimental prototypes.

Recent research confirms the frequent usage of prototypes in organisations. However, stakeholders with different needs may prefer distinct approaches or tools. For example, engineers may create prototypes to assess project feasibility, while designers may employ prototypes to engage with users and determine their needs.

3.4.4 Joint Application Development (JAD)

Joint Application Development (JAD) is an interactive systems design concept that involves eliciting requirements through discussion groups in a workshop setting. It emphasises user involvement through facilitated group discussions led by a neutral moderator. JAD represents an organised and structured approach to requirements elicitation, combining brainstorming with user and stakeholder participation in design discussions.

Typically, JAD sessions involve 20 to 30 participants (Maiden et al., 1996). This technique allows engineers to initiate group sessions by providing a system overview, followed by discussions with stakeholders and users until the final requirements are established (Maiden et al., 1996).
According to recent studies, businesses frequently use JAD or RAD best practices. JAD performs less well in large, complex projects but produces more significant outcomes in focused, small-scale initiatives (Pacheco et al., 2018b).

3.4.5 Contextual Inquiry

Contextual Inquiry involves examining and understanding users' workplace tasks, issues, and preferences (Diefenbach et al., 2019). Researchers immerse themselves in the user's context or environment to observe their work activities. Contextual inquiry is similar to user interviews, as researchers often ask questions while watching task performance to understand observed behaviours better. This approach produces task analyses and user requirements, utilising an ethnographic user-centred design technique. (Darin et al., 2020; Kulkarni et al., 2012).
The contextual inquiry process relies on the analyst's close observation of task procedures. When observing and discussing work practices at the user's workplace, the analyst assumes a background role, collaborating with the user as an apprentice while the user serves as the expert. Contextual Inquiry is based on four fundamental principles: context, partnership, interpretation of data, and reasoning from facts to hypotheses and implications to design (Darin et al., 2020; Kulkarni et al., 2012).
Despite its numerous advantages, the Contextual Inquiry method is rarely employed in industrial contexts. According to Holtzblatt et al. (1993), researchers require assistance describing user requirements since contextual information always needs to be included. Moreover, the method can be cost-intensive and often requires stakeholder involvement.
Here are data collection methods in the Synthetic category and how they are currently used in Table 4.

Table 4. Summary of Synthetic Methods.

Method Name	Data Collection Methods	State of Practice
Scenarios (Jaramillo , 2015).	Data analysis.	Scenarios are used in various domains, such as market research, user experience and strategic planning.
Use Cases (Regnell et al., 1995).	Stakeholder interviews or Workshops.	Use cases are widely used in software development to capture functional requirements and define system behaviour.
Prototypes (Jensen et al., 2018).	Simulation or creating representative models.	They are widely used in designing and developing software, websites, mobile applications and physical products.
JAD (Maiden et al., 1996).	Collaborative Workshop, Brainstorming.	In use in many organisations. Effective in issues with varied stakeholders
Contextual Inquiry (Darin et al., 2020; Kulkarni et al., 2012).	Observation and Interaction with Users.	They are widely used in various industries and domains, including software development, user experience design, and product development.

Challenges in requirements gathering for software development have been the subject of intense debate among researchers and IT professionals. Tables 4, Table 5, and Table 6 present expert recommendations from the literature review. The tables provide approximate descriptions of each technique, including its contribution to a specific project, strengths and weaknesses, and the current state of practice. According to these experts, these techniques have the potential to overcome challenges in software development and can be effectively utilised in student software projects (Zhang, 2017; Dar et al., 2018; Bahurmuz et al., 2021; Sajjad et al., 2010; Sharma et al., 2013; Kaleel et al., 2013) and (Paetsch, 2003).

Table 5. Requirements Elicitation Methods.

Techniques	Description	Strength	Weakness
Interviews	The most popular method is used both in industry and academia. Allows face-to-face communication between the development team and stakeholders. It saves time because it helps the organisation quickly get the requirements of the software or system.	It is suitable for eliciting stakeholders' agenda of what is relevant. Different variations are available. Structured Interview: It is systematic and constant across stakeholders. Different variations are available (Maiden et al.,1996).	Unstructured Interviews: Takes too much time and side issues. The captured information may take much work to analyse.
Workshop	A workshop is a method used to gather information and requirements from stakeholders in a collaborative and interactive setting. (Maiden et al., 2004).	It can integrate other elicitation techniques into them and incorporate their combined usage into a defined requirements process (Maiden et al., 2004).	It may consist of dominant and biased participants (Fernandes et al., 2012). It may include dominant and biased participants (Fernandes et al., 2014).
Prototypes	It is designed at the early phase of the project's implementation to visualise the system's function.	It helps potential users and developers gather valuable feedback and insights, which can be used to improve the design and functionality of the product before investing in full-scale development (Maiden et al., 1996).	If misused, prototyping can have inherent dangers for system design (Maiden et al., 1996).
Laddering	The laddering technique utilises structured interviews to establish attribute, consequence, and core value hierarchy based on the means-end theory that identifies product attributes and their associated effects (Stewart et al., 1998).	The laddering process can be viewed in hierarchical structures based on the means-end chain. This graphical representation helps researchers and marketers grasp the connections between product attributes.	It is time-consuming due to extensive participant interviews to explore their thought processes. Not suitable for large-scale projects.
Joint Application Development (JAD)	JAD is a collaborative technique that brings together key stakeholders, end-users, and development teams in a collaborative workshop setting (Andrew, 1992; ACRE,1996).	It brings together all relevant stakeholders and promotes a shared understanding of requirements (ACRE, 1996).	Strong personalities within the JAD sessions may overshadow others, leading to limited contributions from specific stakeholders or overlooking valuable insights.

Table 6. Requirements Elicitation Methods.

Techniques	Description	Strength	Weakness
Questionnaires or Surveys	Reach multiple people in a short time. A comprehensively designed questionnaire reflects the actual stakeholder's requirements.	It is simple and does not require much training and preparation (Maiden et al., 1996).	It can capture large amounts of irrelevant data (ACRE, 1996). It does not give more room to investigate the topic further or expand on new ideas (Ari et al., 2009).
Social analysis or Ethnography	An observer spends a period in a society or culture. Usually, they are interested in organisational culture and design solutions. (Zhang, 2007).	It can integrate other elicitation techniques into them and incorporate their combined usage into a defined requirements process (Maiden et al., 2004).	It may consist of dominant and biased participants (Fernandes et al., 2012).
Contextual Inquiry	Contextual Inquiry involves examining and understanding users' workplace tasks, issues, and preferences (Diefenbach et al., 2019	By observing users in the real world, analysts can better grasp the context, constraints, and challenges users face during interactions.	Analysing and synthesising data from contextual inquiry can be complex, especially when dealing with a large amount of qualitative data.
Brainstorming	Requirements engineer asks a group of stakeholders to generate as many ideas as possible, emphasising generation instead of evolution (ACRE, 1996).	Suitable for eliciting high-level domain entities and questioning assumptions which might otherwise have constrained approaches considered (ACRE, 1996).	Susceptible to group process; unsystematic in classic form, though some varieties overcome this (ACRE, 1996).
Focus Group	It is a technique containing four to nine different users with various skills and focuses on the system's features (Naela, 2021).	Stakeholder representatives gather for a short but intensely focused period to create or review high-level features of the desired products (Zhang, 2007).	Participants may feel uncomfortable, as stated by the group (Pitula et al., 2011). There may be dominant participants (Fernandes et al., 2012).

Table 7. Requirements Elicitation Methods.

Techniques	Description	Strength	Weakness
Requirements Re-use	Requirement of the existing system. Software re-use is creating software systems from existing software rather than building one from scratch (Krueger, 1992).	There is consistency across the ecosystem- with reusable components, user - experience will be the same across all products. Easy to maintain the effects (ACRE, 1996).	This may increase maintenance costs. Maintenance costs may increase if the source code of a reused software system or component is unavailable (Ari et al., 2009).
Documentation studies	A method that gathers and analyses information from existing systems and other related information to understand how the current system works. (Bowen, 2009). (Zhang, 2007).	When Stakeholders are unavailable to provide insight into the current business processes, it becomes vital to use document analysis (Bowen, 2009).	Contrarily, it might give a partial grasp of the topic being studied.
Repertory Grid	The method involves asking participants to compare and contrast elements of a particular domain, such as products, people, or experiences, based on their constructs or criteria(Grunert et al., 2005)	Repertory Grid can be adapted to various domains and research questions, making it a versatile method for studying diverse topics.	Constructing and analysing repertory grids can be time-consuming, mainly when dealing with many participants and elements.
Scenarios	A scenario describes a sequence of actions and events for a specific case of some generic task the system intends to accomplish. Methods include use cases (Jaramillo, 2015).	Scenarios prioritise the user's perspective, enabling designers and developers to see the product through the eyes of the users and better understand their needs and expectations (Davis et al., 1992).	Scenarios rely on assumptions and interpretations, which can introduce biases. Different team members may develop different scenarios based on their understanding of the users.
Use cases	It describes how a system, product, or service is used by its actors (users or external systems) to achieve a specific goal (Regnell et al., 1995).	Use cases provide a clear and structured way to communicate system requirements and functionalities to stakeholders, including developers, testers, and end-users (Leffingwell et al., 2000).	Use cases may not capture all possible scenarios and interactions, leading to potential omissions or missed requirements. (Fernandes et al., 2012).

4 CHALLENGES IN STUDENT PROJECTS

Student software projects play a vital role in software engineering education, allowing students to apply their knowledge and skills to real-world problems. However, these projects are challenging, and students must overcome these challenges to achieve successful project outcomes. This chapter explores and discusses the challenges students encounter in student software projects. The content of this chapter is based on a systematic literature review study conducted by Tenhunen et al. (2023). The study showed a comprehensive review of students' challenges in student projects, analysing various aspects of project-based software engineering capstone courses and examining the advantages and disadvantages of different approaches employed in these projects. Through the analysis of 127 articles, several challenges emerged, including project implementation, maintenance of existing software, team size, technologies, teaching approaches, requirements engineering process, requirement modelling, and programming.

One of the challenges in student projects is project implementation and the maintenance of existing software. Agile methodologies are commonly adopted in student projects, prioritising functional software development over extensive documentation. Striking the right balance between documentation and deliverables can be a challenging task. According to Tenhunen et al. (2023), detailed software requirement specifications are essential while implementing agile practices such as user stories and backlog updates in student projects. However, it is necessary to note that software maintenance should receive more attention in student projects despite its significance in the software life cycle. Incorporating software maintenance activities and exposing students to existing software systems can significantly enhance their understanding of the maintenance phase. In a study that examines software integration, Weissberger (2015) explores the integration of software maintenance in a senior capstone project by combining agile and waterfall models. This approach aimed to equip students with the necessary skills and knowledge to effectively address the challenges of maintaining existing software systems. By integrating maintenance activities into the project, students were able to gain practical experience in handling software issues and enhancing the long-term viability of their projects.

Another challenge in student projects is team size. Determining the optimal team size is crucial for successful project execution. Large teams may encounter communication and coordination issues, while smaller groups may require assistance completing large-scale

projects. According to Mundra and others (2013), teams in student projects using the Agile methodology often have 2 to 6 members.
Technologies present another challenge for student projects. The choice of technologies employed in student projects can significantly impact team performance. Students may require assistance dealing with unfamiliar technologies, limited resources, or conflicting technology preferences within the team. Providing students with access to necessary hardware, software, and technical support is essential. A recent study by Tenhunen et al. (2023) highlights the importance of shared infrastructure tools and the evaluation of technological implementation in student projects.
Teaching approaches also pose a challenge in student projects. Project-based approaches are increasingly utilised in software engineering education to enhance student learning outcomes. Instructors take on multiple roles as mentors, evaluators, and occasionally project sponsors. Striking a balance between content coverage and process-oriented learning experiences is crucial. The particular pressures project-based teaching methodologies place on teachers and students are discussed (Yu, 2014).
Furthermore, the requirements engineering process presents challenges for students. Students often require assistance in managing the requirements engineering process, which involves handling incomplete specifications, lack of documented requirements, and requirement changes. A study on requirements management strategies for student projects highlighted common issues and difficulties (Mäkiaho, 2017).
Requirement modelling is another challenge faced by students in student projects. Requirement engineering modelling poses difficulties, including understanding the value of modelling, grappling with the intangibility compared to coding, and addressing the emphasis on programming skills in job interviews. In discussing the difficulties, students face when modelling requirements, Berre et al. (2018) intensely focus on the value of interaction and debate in finding satisfying answers.
Finally, programming poses a significant challenge in student projects. Teaching programming is complex, as students often require assistance interpreting abstract problems and developing systematic problem-solving methods. The difficulties students encounter when learning to program, and the high failure rate in programming courses are highlighted (Babb et al., 2014; Canedo et al., 2018).

5 CHALLENGES OF ELICITING REQUIREMENTS IN STUDENT PROJECTS

A crucial step in the software development process for student projects is requirements gathering. However, this process often presents challenges that impede the successful identification and translation of stakeholder needs into actionable project specifications. This chapter examines the challenges faced during requirements elicitation in students' projects and explores strategies for overcoming each limitation. By understanding and addressing these challenges, students can enhance their ability to gather comprehensive requirements and effectively meet project objectives (Sharma et al., 2014; Tenhunen et al., 2023).

One primary challenge encountered in requirements elicitation is the presence of ambiguous or vague requirements. Students may need help deciphering the exact expectations and objectives of the project due to unclear or imprecise language used by stakeholders. Overcoming this limitation involves employing requirement clarification sessions, actively engaging stakeholders in discussions, and utilising visualisation tools to ensure a shared understanding of requirements (Dar et al., 2018; Pacheco et al., 2018).

Another significant challenge is students' lack of domain knowledge in their specific student project domain. This lack of expertise can hinder their comprehension of industry-specific requirements, potentially leading to incomplete or inaccurate specifications. Students can overcome this limitation by conducting thorough research, consulting domain experts, and engaging in knowledge-sharing sessions to understand better the project domain (Bahurmuz et al., 2021; Liu et al., 2010) and (Bjarnason et al., 2011; Sajjad et al., 2010).

Engaging and involving stakeholders in the requirements elicitation process is crucial for obtaining comprehensive and accurate project requirements. However, stakeholders may have busy schedules or conflicting priorities, making their active participation challenging to secure. To overcome this challenge, students should establish clear lines of communication, schedule regular meetings, and employ various techniques such as interviews, surveys, and workshops to elicit and validate requirements effectively (Eveleens et al., 2010; Fricker et al., 2015; Sharma et al., 2014).

Furthermore, student projects often span an extended period during which requirements may change or evolve. External factors such as market trends or emerging technologies can influence project objectives, necessitating flexibility in adapting to the evolving requirements. Students can address this challenge by establishing a robust change control process, maintaining open lines of communication with stakeholders, and regularly reviewing and updating project requirements to accommodate evolving needs (Bahurmuz et al., 2021; Dar et al., 2018; Fan, 2018).

Scope creep, the tendency for project requirements to expand beyond the initial agreement, is a common challenge in student projects. It can result from poor requirement management, unclear project boundaries, or stakeholder requests for additional features. Students can mitigate scope creep by clearly defining project scope, employing change management processes, and regularly revisiting project objectives to ensure alignment with stakeholders' expectations (Bahurmuz et al., 2021; Svensson et al., 2010) and (Berntsson et al., 2012).

Communication and language barriers are also challenges in student projects. They often involve diverse stakeholders with different communication styles, cultural backgrounds, and levels of technical expertise. Communication and language barriers can hinder effective requirements elicitation and lead to misunderstanding. To overcome this limitation, students should strive for clarity in their communication, avoid technical jargon, actively listen to stakeholders, and employ visual aids or diagrams to enhance understanding (Dar et al., 2018; Dehlinger et al., 2011).

Limited resources pose another problem for students. They may need help accessing resources such as industry experts, specialised tools, or real-world datasets. These limitations can hinder their ability to gather comprehensive requirements or perform detailed analyses. Overcoming this challenge involves seeking alternative resources, leveraging online platforms and communities, and collaborating with academic institutions or industry partners to access necessary resources (Cheng et al., 2010; Sommerville, 2007).

Furthermore, student projects typically have strict timelines, limiting the time for requirements elicitation. To effectively manage time constraints, students should prioritise requirements, employ efficient requirement-gathering techniques such as interviews or surveys, and adopt agile project management methodologies that facilitate iterative feedback and adjustment of requirements within the given timeframe (Alam et al., 2017).

Balancing multiple perspectives is another challenge in student projects. Capstone projects often involve collaboration among students from different disciplines, each with their views and priorities. Balancing and integrating these diverse requirements can be challenging. Students can address this limitation by fostering effective collaboration, encouraging open dialogue, facilitating compromise and negotiation, and employing techniques such as requirements prioritisation to ensure a harmonious set of project specifications (Vanhanen et al., 2018).

Finally, a lack of experience in requirements elicitation is a challenge for students. They may need more experience in effective techniques and best practices. To overcome this limitation, students should seek mentorship from faculty advisors or industry experts, undergo training in requirements engineering, and actively engage in knowledge-sharing platforms and communities to enhance their skills in gathering and managing project requirements (Al-Zawahreh et al., 2015; Carrizo et al., 2014; Vanhanen et al., 2018).

6 CASE STUDY: ELICITING REQUIREMENTS FOR THE eTANDEM WEB APP

This case study was conducted to identify challenges students face in eliciting requirements and techniques most appropriate for student projects. By analysing the experiences and outcomes of the eTandem web application project, this case study aims to shed light on the challenges encountered and techniques during requirements elicitation in student projects.

6.1 Context

Software engineering projects give the student experience developing a software product throughout the software life cycle. Projects such as this give students practical experiences applying concepts they have learned in their software engineering and computer science classes. Students engage in various activities within these projects, such as project management, requirements elicitation, software design, implementation, testing, and requirement management. This case study focuses on the experience of the developed eTandem web application for a client seeking language teaching tools to facilitate language learning and teaching among students. The primary objective of this case study is to explore how this project addressed the challenges of eliciting requirements in student projects and to identify suitable elicitation techniques.

6.2 Data Collection and Analysis

This case study collected data from previous student projects at Tampere University, Finland. The student project courses offered twice a year during the autumn and spring semesters allow students to implement their theoretical understanding in real-world software engineering situations. The course aims to develop soft and hard skills essential to software engineering education.

This case study focuses on the 2019 implementation of the course, in which a six-member team, including the author of this thesis, was assigned to execute a project. The participating students in the project assumed the project manager and developer roles to contribute to the project effectively.

The developer role required participants to have completed core computer science courses, which involved working on projects, documenting software projects, presenting findings, understanding ethical norms, and being familiar with software toolkits.

Developer participants primarily consisted of students participating in the project work for the first time and often seeking assistance from the project manager when needed. On the other hand, students who had completed additional "project work" courses and studied software project management theory were qualified to become project managers. These project managers had more experience and had taken prerequisite courses to enhance their project management skills.
Throughout the project implementation, the course staff employed four primary approaches to data collection in student projects. The first approach involved collecting minutes from weekly meetings, which provided information about task descriptions, hours spent, encountered risks, risk mitigation measures, work metrics, and project problems. Personal reports were also collected, offering individual perspectives on the project, including task descriptions and personal and collective approaches to task execution. These reports also provided insights into working hours, technical proficiency, tool utilisation, teamwork, and the project managers' management capabilities. Additionally, the final reports served as a crucial data source, as they included a project plan document outlining the project's conclusion, steps taken, and other essential elements typically discussed in project reports.

6.3 Method and Process

During the fall of 2019, the project course was implemented with 12 teams, each consisting of 5-7 members. Each team was allocated 1000 hours to work on their projects, with 60 participating students. In this study, the author was part of a group of four students who assumed the role of project manager. Although requirements engineering was not explicitly included in the course, it had been covered as a mandatory component in previous studies for students enrolled in the software project management course. Each project team had its client and was encouraged to self-organize based on the guidelines provided by the course staff. The teams could select the tools that best suited their project requirements. The university provided essential software development and design tools, such as Balsamiq, Redmine, and various programming development tools. However, utilising these tools was optional, allowing teams to decide whether or not to use them based on their specific needs.
A course supervisor was assigned to each project team to offer assistance and guidance throughout the development process based on the team's requirements. However, the supervisor's role was limited to providing support and was not directly involved in the

project. The supervisor's responsibilities included selecting appropriate tools, addressing project-related issues, and occasionally mediating conflicts within the team.

Agile is the preferred development process for project teams; it promotes intensive communication and collaboration with stakeholders throughout the development process. Agile methodology allows the project team to accommodate and embrace changes, offering flexibility not commonly found in other software development approaches. Project teams can easily incorporate changes and prioritise tasks by implementing a product through a series of iterations and maintaining a dynamic backlog. The backlog, part of the Trello platform, is crucial in managing project requirements. It facilitates effective communication by bridging the gap between verbal descriptions and visual representations of backlog items. This enables the project team to track the state changes of each requirement, including stages such as “new”, “in progress”, “done”, and “rejected” requirements. Providing clients and supervisors with online visibility of the backlog list items is a welcoming gesture that promotes transparency and encourages client involvement as an integral part of the team.

The project team had five mandatory meetings with the supervisor and clients: a project plan inspection, three review meetings and a final meeting. These meetings served various purposes, including project planning, product validation, and gathering stakeholder feedback to inform the next iteration. To provide students with flexibility, most project teams opted for regular meetings instead of daily meetings, allowing them to attend to other obligations unrelated to project work. Additionally, ad-hoc meetings were scheduled to support team members who required assistance meeting deadlines or faced task challenges.

In our team, we primarily conducted online meetings and regular meetings most of the time. We utilised the meeting to integrate tasks in GitHub and plan for the next iteration. Weekly meetings were held to ensure smooth coordination and collaboration among team members.

6.4 Project Management

As with any software development effort, there is a need for project teams to define the project management activities to track and correct issues to minimise their impact on the project. This is especially true since the students have a fixed schedule that does not slack in the project if the deadline is delayed.

There are steps to follow for developing software in student projects through the project guidelines, which mirror the project plan. These are the project initiation, project planning, execution, monitoring, communication, risk management, and stakeholder management, as discussed below.

Project teams often define project goals, objectives, scope, and stakeholders in software development at the beginning of the project. The project initiation involves identifying the feasibility, constraints, and risks. This is followed by defining the project planning, enabling teams to develop a comprehensive project plan for project deliverables, creating a work breakdown structure (WBS), determining project schedules, estimating resources and costs, and creating a risk management plan.

Project teams also carry out the project through implementing the project plan, task delegation to team members, resource management, and monitoring schedule compliance. Project communication is the strategy for managing projects by project teams. The aim is to communicate and coordinate project activities. Coordinating project activities takes various communication tools, and teams can choose them based on the project need or the team members' experience. Effective communication, collaboration, and coordination among team members are necessary for project success.

On the other hand, project monitoring and control are used to track progress, compare it against the project plan, and make necessary adjustments. It involves monitoring project milestones, managing changes, resolving issues, and assessing risks to keep the project on track. One such tool is the metric monitoring tool developed at the Department of Computer Science at Tampere University Finland specifically for student projects.

Project communication can take various forms to establish effective channels within the project team and stakeholders. It encompasses regular reporting and progress updates and facilitates collaboration and information sharing. Project teams have access to various communication tools, and the choice of tool depends on the project's specific needs and the team's experience. Popular communication tools include Slack, Microsoft Teams, Discord, Telegram, Zoom, WhatsApp, and Google Meet. These platforms are utilised to organise various types of project meetings. Furthermore, Task boards and Trello are commonly employed within the project management category. Trello software serves multiple purposes, including selecting and moving cards from the product backlog to the sprint backlog during sprint planning, enabling the creation of a focused and manageable set of tasks for the upcoming sprint. It also facilitates collaboration and communication among team members and stakeholders. Email is predominantly used for communication with

clients and sometimes within project teams. Regarding file sharing, Google Drive, OneDrive, and Word documents are frequently utilised.
Furthermore, during the project implementation, project teams define risk management to identify potential risks that may impact the project's success and develop strategies to mitigate or respond to them. It often includes risk identification, assessment, prioritisation, and implementation of risk mitigation plans. Risk management is defined at the beginning of the project while thinking about different scenarios which would happen throughout the project that would impede project success. Risks identified by team members are sometimes identified individually and combined in a brainstorming session. Risks are associated with project estimation based on the probability of the risk occurring and how much impact the risk would make on the project.

6.5 Managing and Leading Project Teams

Managing and leading project teams involves various responsibilities, such as assigning roles and responsibilities, resolving conflicts, motivating team members, and fostering a positive team culture. The agile software development process is widely adopted by teams in student projects due to its inherent flexibility, especially when combined with the Scrum variant. In my experience as the author of this thesis, I participated in student projects with a team consisting of two project managers, two developers, one tester, and one UX designer. Despite being project managers, we assumed roles as the Scrum master and the product owner, respectively. As the Scrum master, I was responsible for project activities, including adherence to the Scrum process, removing obstacles, facilitating events, and ensuring effective communication. While serving as a project manager, I actively participated in other project tasks, as the workload was distributed equally among team members with firm deadlines.
The second project manager took on the product owner role and was responsible for requirement elicitation and prioritisation. In Scrum, requirements elicitation is accomplished through the product backlog, which contains a comprehensive list of work items for the project. The product owner and the development team collaborated in analysing, specifying, and validating the requirements. At the beginning of each development iteration, the team and product owner jointly decided on the backlog items to be implemented based on the team's performance. After the iteration, the product owner conducted product validation, including assessing the product's behaviour.

Quality Management ensures that project deliverables meet the required standards and specifications. It involves defining quality objectives, implementing quality assurance processes, and conducting quality control activities.
Stakeholder Management ensures that industry clients, who are the source of the requirements in student projects, are identified, and their requirements are elicited. Project teams must proactively identify and engage with stakeholders, understanding their needs and expectations and managing their involvement throughout the project lifecycle. Finally, project closure formalises project completion, conducting project reviews, documenting lessons learned, and celebrating project achievements. It involves ensuring that all project deliverables are completed and handed over appropriately.

6.6 Requirements Elicitation in Student Projects

The project team meets with the client at a visiting meeting to build a relationship and acquire information. During this meeting, the team familiarises themselves with the client involved in the product development process. Additionally, the team makes necessary arrangements to elicit requirements from the client and discuss his availability in meetings and throughout the development process.
This was followed by a project plan inspection meeting, a collaborative session in which the project team, supervisor, clients, and relevant parties come together to review and assess the project plan. The objective of the meeting is to ensure that the project plan is comprehensive, realistic, and aligned with the project goals and objectives. The primary purpose of the project plan inspection meeting is to identify potential issues, gaps, or areas for improvement within the project plan before the execution phase. The topics discussed during the meeting include plan review, stakeholder input, risk assessment, alignment of objectives, quality assurance, documentation, and updates that must be addressed before the project's execution phase. Identifying and interacting with industry clients is common among project teams when eliciting requirements. The involvement of clients plays a vital role in the success of a project. Also, during the eTandem project, our team used interviews to elicit requirements. The selection of elicitation techniques often depends on the project's unique characteristics. Each project team chooses techniques that align with their specific project requirements. While interviews are popular among many teams, it is essential to note that other project teams may opt for techniques other than interviews. The guidelines explicitly state that project teams can choose any tool suitable for their projects, including requirement elicitation techniques. Additionally, research suggests that

interviews are particularly effective in agile development, as direct communication with customers helps prevent misunderstandings.

Furthermore, our team prioritised requirements in order to meet client needs. The reasons for prioritising requirements may be based on their value, aiming to deliver maximum business value and the highest-value features early in development. Project teams often use Trello software to manage the product backlog. The backlog items of interest are selected based on their value, priority, and necessity.

These items are then converted into tasks and estimated, resulting in the creation of the sprint backlog. The sprint backlog comprises the set of backlog items that will be completed during the upcoming sprint.

A sprint represents an actual iteration in the scrum process, typically lasting 2 to 4 weeks. While daily scrum meetings are critical to the scrum, many project teams skip them due to conflicts with students' class schedules. Instead, these teams often schedule flexible weekly meetings to discuss project progress. However, project teams that conduct daily scrum meetings typically allocate around fifty minutes for the team to synchronise. Project accomplishments since the last meeting are discussed and shared among team members during these meetings. A to-do list is created for the upcoming meetings, which includes discussing obstacles analysis and providing solutions.

Another vital meeting, the client review meeting, is conducted after each iteration. This meeting involves the project team, industry client, and project supervisor. The purpose of the meeting is to showcase the new software and gather feedback that can inform the next sprint. This feedback loop continues throughout the project's lifecycle.

At the end of each 2- to 4-week cycle, a final review meeting and retrospective take place. The sprint review is typically a two-hour session packed with various activities. During this meeting, the product owner evaluates the achievements of the specific sprint, and the team engages in discussions regarding any issues encountered and resolved. Furthermore, a demonstration of the deliverable is typically provided.

Immediately following the sprint review, the retrospective meeting takes place. In this session, the project team deliberately reflects on their performance and seeks ways to improve their agile practices in subsequent sprints. Each team member is encouraged to identify specific actions that the team should continue, stop, and start doing, fostering a culture of continuous improvement. As the project progresses and understanding deepens, new requirements may emerge. Regular prioritisation throughout the development lifecycle ensures that priorities remain current and aligned with the project objectives.

Models capture essential elements such as user interfaces, data flows, system components, and process workflows of a system. The project team employs modelling techniques to transform requirements into visual representations that reflect the desired product. This step is vital because it determines whether the system will succeed or fail in meeting the client's needs. Various tools are used by the project team during the design process, facilitating constant interaction, particularly between the UX designer and the client. The client's validation of the design in progress is essential, and any clarification required by the designer is addressed through direct contact.

Throughout the implementation of the eTandem design, the designer maintained communication with the clients through various channels, including emails, telephone calls, Microsoft Teams, and Slack. The choice of communication tool depended on the type of information needed by the design team and the client's preferences.

Our project team utilised different tools to facilitate the design process, starting with creating quick sketch models of the eTandem app using wireframes and prototypes. Wireframes and prototypes allowed the team to represent the eTandem user interface and functionality visually. This enabled the project team and clients to validate requirements early in development and ensure alignment between expectations and system design.

7 RESULT

This chapter summarises the results obtained and addresses the research questions. The challenges faced in eliciting requirements in student projects are presented in Table 8, categorised into four main categories: requirements, communication, stakeholders and developers. The challenges are further classified under the respective criteria. It was found that requirements-related issues play a significant role in the elicitation process for student projects. These challenges have been identified through previous studies and are crucial to address to overcome problems during requirements elicitation in student projects.

Research Question 1: What are the challenges of eliciting requirements in student projects?

The challenges in eliciting requirements in student projects often arise due to the complexity and precision required in information needs, as well as the skill set of the developers, which may be lower than expected. Effective communication among the development team is crucial for successful requirements elicitation. However, challenges may arise regarding presenting information, language barriers, and cultural variations due to the diverse composition of students involved in the projects (Dar et al., 2018b; Sajjad & Hanif, 2010).

Research Question 2: What elicitation techniques are suitable for student projects?

To address the second research question, this study investigated the 17 most common elicitation techniques used in student projects, as presented in Table 5, Table 6, and Table 7. These tables comprehensively describe each technique, including its contribution to the project, strengths and weaknesses, and the current state of practice. Additionally, a case study was conducted to explore the application of these techniques in the context of student projects.

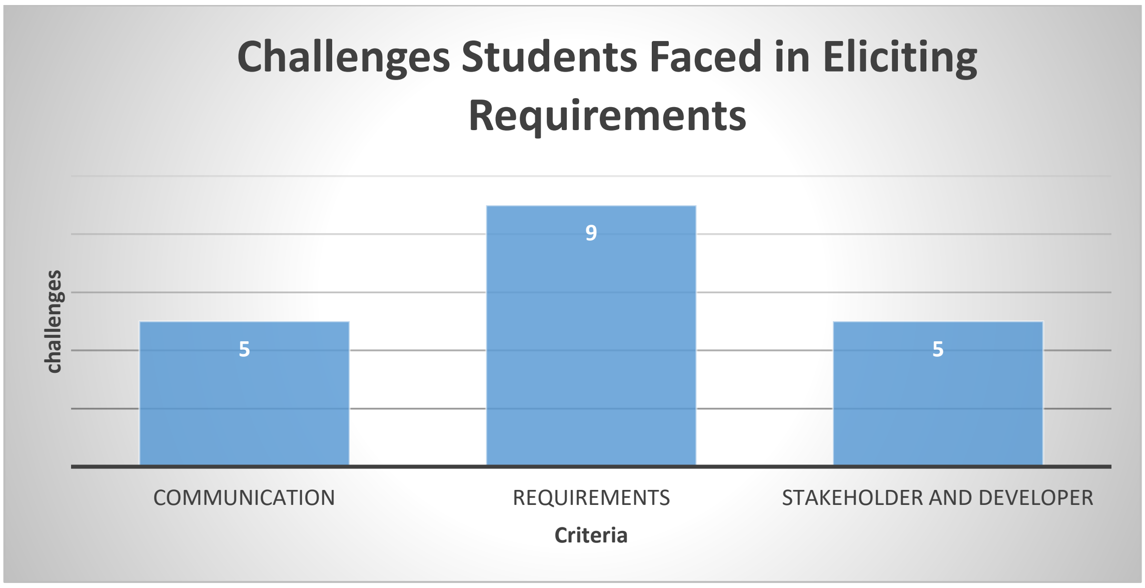
The data used in this study was collected during the software engineering project in the fall of 2019. Project teams demonstrate diversity in their elicitation techniques, influenced by their experience and specific project requirements. The selection of an elicitation technique is often guided by its compatibility with the employed development process. For example, project teams prefer the interview technique in agile development processes due to its interactive and collaborative nature, facilitating effective client interaction. Furthermore, in some instances, project teams may opt to combine multiple techniques based on the nature and complexity of the project.

7.1 Challenges Students Faced in Eliciting Requirements.

The primary source of requirements for student projects is the client. Project teams employ different approaches to elicit requirements. Some teams receive features directly from the client, which means that requirements have already been elicited upfront, and there is no need for further requirement elicitation. However, these teams may seek clarification from the client if any features are unclear. On the other hand, some teams actively engage in requirement elicitation by collecting requirements directly from the client.

In the personal report, each student was asked whether they had participated in requirement elicitation and if they encountered any challenges during the process. Project team members provided individual responses in text format, which were then collated and mapped into categories. The results are presented in Figure 1. Based on the chart, the challenges were categorised into criteria, with the most significant challenges being related to requirements, followed by communication, stakeholder, and developer. These four challenges form the main criteria that must be addressed: requirements, communication, stakeholder involvement, and developer and personalities involved.

Figure 1. Challenges in Eliciting Requirements in Student Projects



The requirements quality attributes include prioritising, scheduling, traceability, ambiguity, unclear information, unstable requirements, and scope change. These challenges are specifically related to managing and addressing requirements throughout the project lifecycle. The communication quality attributes encompass presentation skills, medium of communication, a tool of communication, cultural and language barriers, and lack of communication skills. These challenges pertain to effective communication and collaboration among team members and stakeholders. The stakeholder and developer quality attributes involve stakeholder identification, student dropout, ambiguous conflicts among stakeholders, and sharing users in the requirements elicitation process. Finally, the personality quality attributes involve challenges related to a lack of cooperation and participation, a loss of control over work burden, and an inability to deal with conflicts.

7.2 Correlation with Project Reports

The challenges reported by project teams in their project reports align with the main challenges identified in the literature review in Chapter 5. The project teams' experiences confirm the prevalence of challenges related to requirements, communication, stakeholder involvement, and developers and personalities in the context of requirement elicitation.

The challenges of prioritising requirements arise when the client desires to implement all requirements, but the project team must decide which to prioritise. This challenge often stems from limited resources and time constraints. It requires careful consideration of the project objectives, stakeholder needs, and available resources to address the most critical requirements.

Problems in planning and estimating the project scope can lead to challenges in scheduling. In student projects, where team members may need more experience and unfamiliarity with the content and resources, accurately estimating project timelines becomes challenging. Lack of proper scheduling can result in delays, poor resource allocation, and difficulties meeting project milestones.

When requirements are written at a general level or need more detail, it can lead to incomplete specifications. This challenge hinders the developer’s understanding of what is required and can result in misinterpretations and errors during implementation. It is essential to ensure that requirements are clearly and precisely defined to avoid ambiguity and confusion.

Student projects often involve intricate functionality, integration with existing systems, or advanced technical constraints. Dealing with such complexities presents challenges in understanding the domain, effective collaboration among team members, and breaking down complex tasks into manageable components. When there is a lack of clarity in various interpretations of requirements, ambiguity results. Communication challenges and clear information contribute to clarity in requirements elicitation. Resolving ambiguity is crucial to ensure all stakeholders understand the requirements, minimising misunderstandings, and preventing rework during development.

Unclear information adds to the challenge of requirements elicitation by creating confusion and hindering effective communication. When the project team needs more clarity in the information provided by clients or stakeholders, it becomes easier to understand their needs and expectations fully. Clear and precise information guides the requirements elicitation process and ensures successful project delivery.

Unstable requirements and scope changes can be disruptive, leading to rework, delays, and challenges in managing project resources. Effective change management practices, such as regular communication, documentation, and stakeholder involvement, are crucial to address these challenges and maintain project stability.

To mitigate these challenges, project teams must adopt practical requirements engineering practices. This includes establishing clear communication channels, conducting thorough analysis and documentation of requirements, involving stakeholders throughout the process, managing change effectively, and promoting collaboration and cooperation among team members. By addressing these challenges proactively, student projects can enhance the requirements elicitation process, ensure a shared understanding among stakeholders, and improve project outcomes.

In the middle of each project, the students were asked to identify the requirements elicitation methods used in their respective projects. With numerous project teams participating, each team listed their specific elicitation techniques, and these answers were recorded in text format. This research investigates the 17 most common elicitation techniques employed in student projects, as presented in Table 5, Table 6 and Table 7. These tables comprehensively describe each technique, including its contribution to the project, strengths and weaknesses, and the current state of practice.

Project teams exhibit diversity in their elicitation techniques, influenced by their experience and project requirements. Additionally, selecting an elicitation technique may be guided by its compatibility with the development process employed. For instance, in agile development processes, project teams often favour the interview technique due to its interactive and collaborative nature, facilitating effective client interaction. Additionally, in some instances, project teams may choose to combine multiple techniques, depending on the nature and complexity of the project.

8 CONCLUSION

Software development projects in industry or student settings are complex endeavours. Among the challenges experienced in software projects, requirement engineering stands out as a crucial issue that can significantly affect a project's success or failure. The requirement engineering process involves complex and error-prone activities that must be undertaken at the project's outset. One of these critical activities is the identification of stakeholders and their requirements before commencing software development. However, identifying the stakeholders and their requirements poses a daunting task to project teams. The challenges in eliciting requirements in student projects often arise due to the complexity and precision required in information needs, as well as the skill set of the developers, which may be lower than expected. Effective communication among the development team is crucial for successful requirements elicitation. However, challenges may arise in presenting information, language barriers, and cultural variations due to the diverse composition of students involved in the projects.

Despite these challenges, it is noteworthy that students in their projects navigate and overcome the difficulties associated with requirement elicitation. This observation underscores the students' ability to understand and effectively address challenges, leading to remarkable project success. Requirements elicitation and mobile app development have received considerable attention in literature and industry, equipping students with valuable knowledge and skills that can benefit them both during their academic studies and in their professional work.

This study aimed to investigate requirements elicitation techniques in student projects. The study sought to identify suitable methods for requirements elicitation, explore the challenges encountered in the process, investigate the 17 most common elicitation techniques employed in student projects, and provide a comprehensive description of each technique, including its contribution to the software projects, strengths and weaknesses, and the current state of practice. Additionally, a case study was conducted to explore the application of these techniques in the context of student projects. The data was collected during the software engineering project in the fall of 2019.

The findings of this study reveal that challenges in student projects primarily revolve around requirements, communication, stakeholders, and team dynamics. The most commonly used requirement elicitation techniques in student projects are interviews. Others include workshops, prototypes and questionnaires, brainstorming, focus groups, use cases, Joint Application Development (JAD), requirements reuse, document analysis, and card sorting.

Project teams exhibit diversity in their elicitation techniques, influenced by their experience and project requirements. Additionally, selecting an elicitation technique may be guided by its compatibility with the development process employed. For instance, in Agile development processes, project teams often favour the interview technique due to its interactive and collaborative nature, facilitating effectiveness.

This study has contributed to identifying numerous requirements elicitation techniques suitable for student projects, shedding light on the associated challenges. This knowledge will have long-term benefits for students as they can apply these techniques in various capacities throughout their careers. Furthermore, by gaining insights into the challenges of eliciting requirements in student projects, students can avoid potential pitfalls and mitigate costs that may arise during project execution.

Further research should be conducted to explore the impact of students' autonomy in selecting elicitation techniques in software projects. This would provide valuable insights into the effectiveness of different approaches and inform decision-making processes for instructors and staff.

REFERENCES

ACM Computing Curricula Task Force (Ed.). (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, Inc. https://doi.org/10.1145/2534860

Alam, S., Nazir, S., Asim, S., & Amr, Dr. (2017). Impact and Challenges of Requirement Engineering in Agile Methodologies: A Systematic Review. *International Journal of Advanced Computer Science and Applications*, *8*(4). https://doi.org/10.14569/IJACSA.2017.080455

Aldabbus, D. S. (2018). PROJECT-BASED LEARNING: IMPLEMENTATION & CHALLENGES. *International Journal of Education*.

Al-Zawahreh, H., & Almakadmeh, K. (2015). Procedural Model of Requirements Elicitation Techniques. *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*, 1–6. https://doi.org/10.1145/2816839.2816902

Ang, J. K., Leong, S. B., Lee, C. F., & Yusof, U. K. (2011). Requirement engineering techniques in developing expert systems. *2011 IEEE Symposium on Computers & Informatics*, 640–645. https://doi.org/10.1109/ISCI.2011.5958991

Babb, J., Longenecker, H. E., Baugh, J., & Feinstein, D. (2014). *Confronting the Issues of Programming In Information Systems Curricula: The Goal is Success*.

Systematic Literature Review. *International Journal of Information Technology Project Management*, *12*(3), 1–18. https://doi.org/10.4018/IJITPM.2021070101

Bahurmuz, N., Alnajim, R., Al-Mutairi, R., Al-Shingiti, Z., Saleem, F., & Fakieh, B. (2021b). Requirements Elicitation Techniques in Mobile Applications: A

Systematic Literature Review. *International Journal of Information Technology Project Management*, *12*(3), 1–18. https://doi.org/10.4018/IJITPM.2021070101

Behdinan, K., Pop-Iliev, R., & Foster, J. (2015). WHAT CONSTITUTES A MULTIDISCIPLINARY CAPSTONE DESIGN COURSE? BEST PRACTICES, SUCCESSES AND CHALLENGES. *Proceedings of the Canadian Engineering Education Association (CEEA)*. https://doi.org/10.24908/pceea.v0i0.5940

Berntsson Svensson, R., Gorschek, T., Regnell, B., Torkar, R., Shahrokni, A., & Feldt, R. (2012). Quality Requirements in Industrial Practice—An Extended Interview Study at Eleven Companies. *IEEE Transactions on Software Engineering*, *38*(4), 923–935. https://doi.org/10.1109/TSE.2011.47

Berre, A. J., Huang, S., Murad, H., & Alibakhsh, H. (2018). Teaching modelling for requirements engineering and model-driven software development courses. *Computer Science Education*, *28*(1), 42–64. https://doi.org/10.1080/08993408.2018.1479090

Bjarnason, E., Wnuk, K., & Regnell, B. (2011). Requirements are slipping through the gaps &#x2014; A case study on causes & effects of communication gaps in large-scale software development. *2011 IEEE 19th International Requirements Engineering Conference*, 37–46. https://doi.org/10.1109/RE.2011.6051639

Boehm, B. W. (1991). Software risk management: Principles and practices. *IEEE Software*, *8*(1), 32–41. https://doi.org/10.1109/52.62930

Bowen, G. A. (2009). Document Analysis as a Qualitative Research Method. *Qualitative Research Journal*, *9*(2), 27–40. https://doi.org/10.3316/QRJ0902027

Canedo, E. D., Santos, G. A., & Leite, L. L. (2018). An Assessment of the Teaching-Learning Methodologies Used in the Introductory Programming Courses at a

Brazilian University. *Informatics in Education*, *17*(1), 45–59. https://doi.org/10.15388/infedu.2018.03

Carrizo, D., Dieste, O., & Juristo, N. (2014a). Systematizing requirements elicitation technique selection. *Information and Software Technology*, *56*(6), 644–669. https://doi.org/10.1016/j.infsof.2014.01.009

Cheng, Y.-P., & Lin, J. M.-C. (2010). A Constrained and Guided Approach for Managing Software Engineering Course Projects. *IEEE Transactions on Education*, *53*(3), 430–436. https://doi.org/10.1109/TE.2009.2026738

Cico, O., Jaccheri, L., Nguyen-Duc, A., & Zhang, H. (2021). Exploring the intersection between software industry and Software Engineering education—A systematic mapping of Software Engineering Trends. *Journal of Systems and Software*, *172*, 110736. https://doi.org/10.1016/j.jss.2020.110736

Clear, T., Co-Chair, W., Goldweber, M., Leidig, P. M., & Scott, K. (2001). *Resources for Instructors of Capstone Courses in Computing*. 93–113.

Corbridge, C., Rugg, G., Major, N. P., Shadbolt, N. R., & Burton, A. M. (1994). Laddering: Technique and tool use in knowledge acquisition. *Knowledge Acquisition*, *6*(3), 315–341. https://doi.org/10.1006/knac.1994.1016

Coughlan, J., & Macredie, R. D. (2002a). Effective Communication in Requirements Elicitation: A Comparison of Methodologies. *Requirements Engineering*, *7*(2), 47–60. https://doi.org/10.1007/s007660200004

Dar, H., Lali, M. I., Ashraf, H., Ramzan, M., Amjad, T., & Shahzad, B. (2018). A Systematic Study on Software Requirements Elicitation Techniques and its Challenges in Mobile Application Development. *IEEE Access*, *6*, 63859–63867. https://doi.org/10.1109/ACCESS.2018.2874981

Darin, T. G. R., Carneiro, N., Paiva, J. O. V., Santos, I. S., & Andrade, R. M. C. (2020). Contextual Requirements Elicitation through the Combination of Interviews, Scenarios and Visual Artifacts. *19th Brazilian Symposium on Software Quality*, 1–10. https://doi.org/10.1145/3439961.3439990

Davis, A., Dieste, O., Hickey, A., Juristo, N., & Moreno, A. M. (2006). Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review. *14th IEEE International Requirements Engineering Conference (RE'06)*, pp. 179–188. https://doi.org/10.1109/RE.2006.17

Davis, A. M. (1992). Operational prototyping: A new development approach. *IEEE Software*, *9*(5), 70–78. https://doi.org/10.1109/52.156899

Dehlinger, J., & Dixon, J. (2011). *Mobile Application Software Engineering: Challenges and Research Directions*. 5.

Diefenbach, S., Christoforakos, L., Maisch, B., & Kohler, K. (2019). The State of Prototyping Practice in the Industrial Setting: Potential, Challenges and Implications. *Proceedings of the Design Society: International Conference on Engineering Design*, *1*(1), 1703–1712. https://doi.org/10.1017/dsi.2019.176

Dyba, T., Maiden, N., & Glass, R. (2014). The Reflective Software Engineer: Reflective Practice. *IEEE Software*, *31*(4), 32–36. https://doi.org/10.1109/MS.2014.97

Eveleens, J. L., & Verhoef, C. (2010). The rise and fall of the Chaos report figures. *IEEE Software*, *27*(1), 30–36. https://doi.org/10.1109/MS.2009.154

Fan, X. (2018). Orchestrating Agile Sprint Reviews in Undergraduate Capstone Projects. *2018 IEEE Frontiers in Education Conference (FIE)*, 1–8. https://doi.org/10.1109/FIE.2018.8658435

Fernandes, S. R. G. (2014). Preparing Graduates for Professional Practice: Findings from a Case Study of Project-based Learning (PBL). *Procedia - Social and Behavioral Sciences*, *139*, 219–226. https://doi.org/10.1016/j.sbspro.2014.08.064

Fricker, S. A., & Schneider, K. (Eds.). (2015). *Requirements Engineering: Foundation for Software Quality: 21st International Working Conference, REFSQ 2015, Essen, Germany, March 23-26, 2015. Proceedings* (Vol. 9013). Springer International Publishing. https://doi.org/10.1007/978-3-319-16101-3

Fuentes-Fernández, R., Gómez-Sanz, J. J., & Pavón, J. (2010). Understanding the human context in requirements elicitation. *Requirements Engineering*, *15*(3), 267–283. https://doi.org/10.1007/s00766-009-0087-7

García-López, D., Segura-Morales, M., & Loza-Aguirre, E. (2020). Improving the quality and quantity of functional and non-functional requirements obtained during requirements elicitation stage for the development of e-commerce mobile applications: An alternative reference process model. *IET Software*, *14*(2), 148–158. https://doi.org/10.1049/iet-sen.2018.5443

Glass, R. L., & Vessey, I. (1995). Contemporary application-domain taxonomies. *IEEE Software*, *12*(4), 63–76. https://doi.org/10.1109/52.391837

Grunert, K. G., & Bech-Larsen, T. (2005). Explaining choice option attractiveness by beliefs elicited by the laddering method. *Journal of Economic Psychology*, *26*(2), 223–241. https://doi.org/10.1016/j.joep.2004.04.002

Herbert, N. (2018). Reflections on 17 years of ICT Capstone Project Coordination: Effective Strategies for Managing Clients, Teams and Assessment. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 215–220. https://doi.org/10.1145/3159450.3159584

Hickey, A. M., & Davis, A. M. (2003). Elicitation technique selection: How do experts do it? *Journal of Lightwave Technology*, 169–178. https://doi.org/10.1109/ICRE.2003.1232748

Holtzblatt, K., & Beyer, H. (1993). Making customer-centered design work for teams. *Communications of the ACM*, *36*(10), 92–103. https://doi.org/10.1145/163430.164050

Irshad, M., Petersen, K., & Poulding, S. (2018). A systematic literature review of software requirements reuse approaches. *Information and Software Technology*, pp. *93*, 223–245. https://doi.org/10.1016/j.infsof.2017.09.009

Jaramillo Franco, A. (2015). Requirements elicitation approaches: A systematic review. *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, 520–521. https://doi.org/10.1109/RCIS.2015.7128917

Jensen, L. S., Özkil, A. G., & Mortensen, N. H. (n.d.). *PROTOTYPES IN ENGINEERING DESIGN: DEFINITIONS AND STRATEGIES*.

Jiang, L., Eberlein, A., Far, B. H., & Mousavi, M. (2008). A methodology for the selection of requirements engineering techniques. *Software & Systems Modeling*, *7*(3), 303–328. https://doi.org/10.1007/s10270-007-0055-y

Kaleel, S. B., & Harishankar, S. (n.d.). *Applying Agile Methodology in Mobile Software Engineering: Android Application Development and its Challenges*.

Kasirun, Z. M., & Salim, S. S. (2008). Focus Group Discussion Model for Requirements Elicitation Activity. *2008 International Conference on Computer and Electrical Engineering*, 101–105. https://doi.org/10.1109/ICCEE.2008.65

Kaul, S., & Stone, W. (2015). Learning Outcomes of a Junior-Level Project-Based Learning (PBL) Course: Preparation for Capstone. *2015 ASEE Annual Conference and Exposition Proceedings*, 26.1074.1-26.1074.14. https://doi.org/10.18260/p.24411

Knobloch, J., Kaltenbach, J., & Bruegge, B. (2018). Increasing student engagement in higher education using a context-aware Q&A teaching framework. *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, 136–145. https://doi.org/10.1145/3183377.3183389

Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. *Improving Schools*, *19*(3), 267–277. https://doi.org/10.1177/1365480216659733

Kulkarni, R., & Padmanabham, D. P. (2012a). *Using Contextual Inquiry as a subset of Requirement Gathering Process*. *3*.

Laplante, P. A. (2017). *Requirements engineering for software and systems*. Auerbach Publications.

Laporti, V., Borges, M. R. S., & Braganholo, V. (2009). Athena: A collaborative approach to requirements elicitation. *Computers in Industry*, *60*(6), 367–380. https://doi.org/10.1016/j.compind.2009.02.011

Leffingwell, D., & Widrig, D. (2000). *Managing software requirements: A unified approach*. Addison-Wesley.

Li Jiang, Eberlein, A., & Far, B. H. (2004). A methodology for requirements engineering process development. *Proceedings. 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2004.*, 263–272. https://doi.org/10.1109/ECBS.2004.1316708

Liu, L., Li, T., & Peng, F. (2010). Why Requirements Engineering Fails: A Survey Report from China. *2010 18th IEEE International Requirements Engineering Conference*, 317–322. https://doi.org/10.1109/RE.2010.45

Maiden, N. A. M., & Rugg, G. (1996). ACRE: Selecting methods for requirements acquisition. *Software Engineering Journal*, *11*(3), 183. https://doi.org/10.1049/sej.1996.0024

Majanoja, A.-M., & Vasankari, T. (2018). Reflections on Teaching Software Engineering Capstone Course: *Proceedings of the 10th International Conference on Computer Supported Education*, 68–77. https://doi.org/10.5220/0006665600680077

Major, C. H., & Palmer, B. (n.d.). *Academic Exchange Quarterly Spring 2001: Volume 5, Issue*.

Mäkiaho, P., Poranen, T., & Zhang, Z. (2017). Requirements Management in Students' Software Development Projects. *Proceedings of the 18th International Conference on Computer Systems and Technologies*, 203–210. https://doi.org/10.1145/3134302.3134340

Mann, S., & Smith, L. (n.d.). Role of the development methodology and prototyping within capstone projects. *2004*, pp. 119–128.

Marques, M. (2015). A Prescriptive Software Process for Academic Scenarios. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, pp. 265–266. https://doi.org/10.1145/2787622.2787743

Marques, M., Ochoa, S. F., Bastarrica, M. C., & Gutierrez, F. J. (2018). Enhancing the Student Learning Experience in Software Engineering Project Courses. *IEEE Transactions on Education*, *61*(1), 63–73. https://doi.org/10.1109/TE.2017.2742989

Marques, M. R., Quispe, A., & Ochoa, S. F. (2014a). A systematic mapping study on practical approaches to teaching software engineering. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 1–8. https://doi.org/10.1109/FIE.2014.7044277

Morais, P., Ferreira, M. J., & Veloso, B. (2021a). Improving Student Engagement With Project-Based Learning: A Case Study in Software Engineering. *IEEE Revista Iberoamericana de Tecnologias Del Aprendizaje*, *16*(1), 21–28. https://doi.org/10.1109/RITA.2021.3052677

Mushtaq, J. (2016). *Different Requirements Gathering Techniques and Issues*. *7*(9).

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: A roadmap. *Proceedings of the Conference on the Future of Software Engineering*, 35–46.

Ouhbi, S., & Pombo, N. (2020). Software Engineering Education: Challenges and Perspectives. *2020 IEEE Global Engineering Education Conference (EDUCON)*, 202–209. https://doi.org/10.1109/EDUCON45650.2020.9125353

Pacheco, C., García, I., & Reyes, M. (2018a). Requirements elicitation techniques: A systematic literature review based on the maturity of the techniques. *IET Software*, *12*(4), 365–378. https://doi.org/10.1049/iet-sen.2017.0144

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, 308–313. https://doi.org/10.1109/ENABL.2003.1231428

Palomares, C., Franch, X., Quer, C., Chatzipetrou, P., López, L., & Gorschek, T. (2021). The state-of-practice in requirements elicitation: An extended interview study at 12 companies. *Requirements Engineering*, *26*(2), 273–299. https://doi.org/10.1007/s00766-020-00345-x

Pitula, K. (2011). *On requirements elicitation for software projects in ICT for development*. Library and Archives Canada = Bibliothèque et Archives Canada.

Project Management Institute (Ed.). (2013). *A guide to the project management body of knowledge (PMBOK guide)* (Fifth edition). Project Management Institute, Inc.

Quintanilla Portugal, R. L., Engiel, P., Pivatelli, J., & do Prado Leite, J. C. S. (2016). Facing the challenges of teaching requirements engineering. *Proceedings of the 38th International Conference on Software Engineering Companion*, 461–470. https://doi.org/10.1145/2889160.2889200

Regnell, B., Kimbler, K., & Wesslen, A. (1995). Improving the use case driven approach to requirements engineering. *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*, 40–47. https://doi.org/10.1109/ISRE.1995.512544

Sabariah, M. K., Santosa, P. I., & Ferdiana, R. (2018). Selecting elicitation technique on requirements elicitation process: A case study on education application for children. *IOP Conference Series: Materials Science and Engineering*, *434*, 012056. https://doi.org/10.1088/1757-899X/434/1/012056

Sajjad, U., & Hanif, M. Q. (n.d.). *Issues and Challenges of Requirement Elicitation in Large Web Projects*. 60.

Shams-Ul-Arif, M., Khan, M. Q., & Gahyyur, S. A. K. (n.d.). *REQUIREMENTS ENGINEERING PROCESSES, TOOLS/TECHNOLOGIES, & METHODOLOGIES.*

Sharma, S., & Pandey, S. K. (2014). Requirements elicitation: Issues and challenges. *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 151–155. https://doi.org/10.1109/IndiaCom.2014.6828119

Sivaloganathan, S. (2004). *Influencing factors from the literature for engineering education*.

Sommerville, I. (2007). *Software engineering* (8th ed). Addison-Wesley.

Stewart, D. W., & Myers, J. H. (1998). Segmentation and Positioning for Strategic Marketing Decisions. *Journal of Marketing Research*, *35*(1), 128. https://doi.org/10.2307/3151936

Svensson, R. B., Host, M., & Regnell, B. (2010). Managing Quality Requirements: A Systematic Review. *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, 261–268. https://doi.org/10.1109/SEAA.2010.55

Tenhunen, S., Männistö, T., Luukkainen, M., & Ihantola, P. (2023). *A systematic literature review of capstone courses in software engineering* (arXiv:2301.03554). arXiv. http://arxiv.org/abs/2301.03554

Ylönen, I. (2021). *User requirements elicitation method comparison for a system upgrade.*

Ul Amin, T., Shahzad, B., Fazal-e-Amin, & Shoaib, M. (2021). Economical Requirements Elicitation Techniques During COVID-19: A Systematic Literature Review. *Computers, Materials & Continua*, *67*(2), 2665–2680. https://doi.org/10.32604/cmc.2021.013263

Valencia, D., Vizcaino, A., Soto, J. P., & Piattini, M. (2016). A Serious Game to Improve Students' Skills in Global Software Development: *Proceedings of the 8th International Conference on Computer Supported Education*, 470–475. https://doi.org/10.5220/0005895904700475

Vanhanen, J., Lehtinen, T. O. A., & Lassenius, C. (2018a). Software engineering problems and their relationship to perceived learning and customer satisfaction on a software capstone project. *Journal of Systems and Software*, *137*, 50–66. https://doi.org/10.1016/j.jss.2017.11.021

Weissberger, I. (2015). INCORPORATING SOFTWARE MAINTENANCE IN A SENIOR CAPSTONE PROJECT. *International Journal of Cyber Society and Education*, *8*(1), 31–38. https://doi.org/10.7903/ijcse.1238

Yousuf, M., & M.Asger, M. A. (2015). Comparison of Various Requirements Elicitation Techniques. *International Journal of Computer Applications*, *116*(4), 8–15. https://doi.org/10.5120/20322-2408

Yu, L. (Ed.). (2014). *Overcoming Challenges in Software Engineering Education: Delivering Non-Technical Knowledge and Skills*. IGI Global. https://doi.org/10.4018/978-1-4666-5800-4

Zhang, X., Dai, H., Hu, T., & Li, X. (2010). *SOFTWARE DEVELOPMENT METHODOLOGIES, TRENDS, AND IMPLICATIONS*. 7.

Zhang, Z. (2007b). Effective requirements development-A- A comparison of requirements elicitation techniques. *Software Quality Management XV: Software Quality in the Knowledge Society, E. Berki, J. Nummenmaa, I. Sunley, M. Ross and G. Staples (Ed.) British Computer Society*, pp. 225–240.

Zowghi, D., & Coulin, C. (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In A. Aurum & C. Wohlin (Eds.), *Engineering and Managing Software Requirements* (pp. 19–46). Springer-Verlag. https://doi.org/10.1007/3-540-28244-0_2