# Leadership Teaching in Agile Software Engineering: A Systematic Mapping

Nicolas Nascimento[a], Afonso Sales[b] and Rafael Chanin[c]
*PUCRS, School of Technology, Porto Alegre, RS, Brazil*

Keywords: Leadership Teaching, Active Learning, Software Development.

Abstract: The software industry is characterized by an environment of uncertainty, high volatility, and constant change. This context has shaped the industry, its components, and actors, generating methodologies capable of meeting both market expectations and software development requirements. Among these methodologies, agile has been the most widely adopted — it provides teams with an overarching set of practices to manage software development project requirements while maintaining flexibility to incorporate changes that the business environment presents. However, professionals equipped with the necessary skills to work in these Agile teams, often referred to as "soft skills", pose a challenge for universities to teach. From this set of skills, leadership, in particular, has recently garnered attention from the software engineering education community but remains an open research opportunity. In this context, this work aims at creating a body of knowledge regarding leadership teaching in agile software engineering. To achieve this goal, we conducted a systematic mapping. From a selection of 27 studies, our results provide indications that: (i) leadership in software engineering education is typically defined as part of teamwork, shared among team members, associated with Scrum, and applied to provide students with experience in group representation; and (ii) it is usually taught through Scrum, involving active learning methodologies (such as problem-based learning) and employing real projects either sourced from external partners or designed to solve real-world problems.

## 1 INTRODUCTION

Great uncertainty, high volatility, and constant change are inherent traits of software development. This environment has shaped the industry, its components and actors, generating methodologies that are able to cope both with market expectations and software development requirements (Fowler et al., 2001). Agile (via its derived frameworks, such as Scrum (Schwaber and Sutherland, 2011) and XP (Beck, 2000)) is the one which has been adopted the most. By providing teams with an overarching set of practices to handle the many assignments a software development project requires while remaining flexible and adaptable enough to incorporate changes that business demands (Dingsøyr et al., 2012).

Professionals that work in these agile teams are expected to not only technically excel, but also to possess abilities to handle conflict, perform on-demand adaptations and communicate properly and effectively. This set of skills is usually denominated *"soft skills"* (*"people skills"*, *"social skills"*, *"generic competencies"*, or *"human factor"*) (Matturro et al., 2015). The need for software developers to possess this set of skills is evident provided their routines usually include interactions with customers, teammates, stakeholders and leadership, where being able to problem-solve, negotiate and resolve conflicts is crucial (Ahmed et al., 2015).

From this set of soft skills, leadership, in particular, is a soft skill that has recently been studied by the software engineering research community. For instance, the relevance of leadership in software development (Faraj and Sambamurthy, 2006; Matturro et al., 2015), styles of leadership (transformational, transactional, among others) and their impacts on the software development life-cycle (Athukorala et al., 2016; da Silva et al., 2016; Faraj and Sambamurthy, 2006; Van Kelle et al., 2015), developing leadership in virtual and globally distributed teams (Furumo et al., 2012; Sangwan and Ros, 2008; Hidayati et al., 2020), and leadership emergence and its antecedents in software engineering (Przybilla et al.,

[a] https://orcid.org/0000-0002-0080-8822
[b] https://orcid.org/0000-0001-6962-3706
[c] https://orcid.org/0000-0002-6293-7419

Nascimento, N., Sales, A. and Chanin, R.
Leadership Teaching in Agile Software Engineering: A Systematic Mapping.
DOI: 10.5220/0013196600003932
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Computer Supported Education (CSEDU 2025) - Volume 2*, pages 469-480
ISBN: 978-989-758-746-7; ISSN: 2184-5026
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

2019; Przybilla et al., 2020) are some of the angles currently being investigated by the community. Moreover, leadership in software engineering is still a topic not fully understood, with studies in the area not providing a unified view of leadership, although hierarchy and management aversiveness are commonly reported (Modi and Strode, 2020).

There are some studies that touch relevant topics on the field, such as leadership teaching and training for software engineering students. Some of this studies, for example, investigate leadership in distance learning and as a component of teamwork (Marquez et al., 2022; Vivian et al., 2013; Noguera et al., 2018), teaching responsible leadership (Goyal et al., 2022), using capstone projects (Schneider et al., 2020; Paiva and Carvalho, 2018; Li et al., 2023) and even incorporating entrepreneurship in teaching (Moreno et al., 2022; Tenhunen et al., 2023) and, thus, the challenge of understanding and adopting the proper way to teach this skill remains for educators.

This study seeks to develop a comprehensive understanding of leadership teaching in agile software engineering. To achieve this, we conducted a systematic mapping to examine how leadership is defined and implemented in educational settings. Our findings indicate that leadership in software engineering education is commonly seen as a component of teamwork, distributed among team members, and closely tied to Scrum practices. It is predominantly taught through active learning methodologies and real-world projects, either sourced from external partners or designed to address practical challenges.

## 2 BACKGROUND

### 2.1 Leadership

There are many definitions of leadership. In our case, to better frame leadership, we have chosen to ground ourselves on the concept of path-goal theory of leadership. This concept refers to body of knowledge that describes the effectiveness of leaders as a consequence of their ability to positively impact peers' motivation, ability to perform effectively and satisfactions (House and Mitchell, 1975). Further, this concept has been empirically studied to have positive correlations to leadership in software projects, such as open-source (OSS) (Li et al., 2012) and has been studied in education in various settings, such as distance learning (Dewan and Dewan, 2010; Bickle, 2017) in industry organizations (Malik et al., 2014).Further, leadership, in a general sense, revolves around two different styles of leadership, which are mostly opposed. These are transactional and transformational leadership (Bass et al., 2003). Each of these style presents different aspects on how guidance of the leader should be provided towards his/her followers.

#### 2.1.1 Transactional Leadership

Transactional leadership is a leadership style that revolves around followers being in accordance or agreeing with the leader either to be rewarded or to avoid any corrective action. Prizes and rewards are awarded based on the compliance of the followers to the expectations of the leader (Podsakoff et al., 1984). Examining from a social context, this style of leadership has been stated to be observed in a "well-ordered society" (Bass and Bass Bernard, 1985).

#### 2.1.2 Transformational Leadership

On the other hand, transformational leadership achieves followers performance enhancement by setting higher expectations and by increasing their willingness to embrace more challenging duties. Further, it emphasizes a greater flexibility of the leader in which better understanding of the organization's challenges are achieved and the solving of complex problems is performed through cooperation between leaders and followers (Bass et al., 2003). In addition, leaders aid followers at performing leadership duties. In this style, followers present personal and socially identification towards the missions of the organization and build identification and motivation on the followers through personal and social identification. From a social context, this style of leadership is stated *"to emerge in times of distress and change"* (Bass and Bass Bernard, 1985).

### 2.2 Agile Software Development

In modern software development, change is a constant and it is often caused by external and uncontrollable factors (Barry et al., 2002). As markets and economies quickly and unpredictably change, traditional software development practices, tool and techniques become difficult to apply and to follow appropriately. In addition, these changes impact on the software development project, which commonly grows both in scope and cost, a phenomenon known as *Scope Creep* (Barry et al., 2002; Melegati et al., 2019). This results in high costs for the development, maintenance and update of software products, thus reducing the competitiveness of software development companies. In this context, as faster and more flexible software development techniques became more necessary, in 2001, the *"Manifesto for Agile Software Development"* (Fowler et al., 2001) was created.

## 3 RESEARCH METHOD

According to standard guidelines for performing this research method (Budgen et al., 2008; Petersen et al., 2008; Kitchenham et al., 2011), a systematic mapping review (SMR) is relevant in software engineering as it *"provides a structure of the type of research reports and results that have been published by categorizing them."* Furthermore, SMR provides an overview of the knowledge in an area, usually aided by a visual summary and can benefit researchers by establishing baselines for further research activities.

Thus, we have followed these standard guidelines and performed our search. At this point, it is important to mention that the research question used for the systematic mapping focused on the concept of how leadership is taught in software engineering education environments more generally, not constrained to only active learning environments. This decision was made to maximize the potential studies and findings from this systematic mapping so that we could create a body of knowledge regarding leadership teaching in software engineering education by characterizing and defining it appropriately.

As stated previously, the research conducted aims at characterizing and defining leadership teaching in software engineering education. As such, the research questions to be addressed in this study are:

- (RQ1) *"How is leadership teaching defined in agile software engineering education environments?"*
- (RQ2) *"How is leadership taught in agile software engineering education environments?"*

In order to answer these questions, a systematic mapping was conducted. The review design was based on some of the most relevant studies in the area (Budgen et al., 2008; Petersen et al., 2008; Kitchenham et al., 2011).

### 3.1 Data Source and Search Strategy

As a way of finding studies with high relevance to the research questions proposed, we defined our search string following the guidelines proposed by (Kitchenham, 2004). As such, the search string addresses the *population*, *intervention* and *outcome* expected. We have chosen exclude *Comparison* and *Context* as the research conducted followed the principle of exploratory research. Table 1 summarizes the search string used. It is important to mention that search string could be adapted based on the limitations of the database, such as limitation in the number of search items. In this case, the last item from the outcome portion of the string was to be removed.

Table 1: Search string.

| | |
|---|---|
| Population | (Agile Software Engineering *OR* Agile Software Development *OR* Agile Software) *AND* |
| Intervention | (Leadership) *AND* |
| Outcome | (Teaching *OR* Educating *OR* Training *OR* Education) |

As for the search strategy, we have also followed the guidelines proposed by Kitchenham (Kitchenham and Charters, 2007). Table 2 presents the summary of the applied search strategy. All available databases were selected with the exception of Citeseer library, Inspec, due to difficulties using these platforms. The minimum publication year was not set. The selection criteria for the search was defined based on the goal of the study. Studies not written in English or not published in any journal, conference, workshop or symposia were not considered. Regarding the number of pages, which usually reflects the depth of the analysis conducted by the authors, we have chosen to accept a minimum of 6 pages, as our intention was to capture even results from preliminary studies.

Table 2: Search strategy.

| | |
|---|---|
| Databases searched | IEEExplore; ACM; Scopus; El Compendex; Science@Direct |
| Selection criteria | available online; 6 pages minimum written in English; up to 2023 in: Journals/Conferences/Workshops/Symposia |
| Search applied to | Title; Abstract; Keywords |

From this point, we have applied our search string and used the search criteria previously specified (Table 2) to all the specified databases. Regarding organization of the results, we have created a spreadsheet which contained the necessary information, meaning that we could apply the established inclusion and exclusion criteria.

Each element in the spreadsheet contained metadata about the retrieved studies. Table 3 presents the attributes assigned to each study. In addition to the standard information of the studies, we have added three additional attributes, which were related to the possibility of exclusion of a given study. These attributes were assigned based on other attributes from the study, such as abstract and keywords.

The goals of these attributes was to better categorize studies and serve as our exclusion criteria, thus answering three additional questions: *"Is this study duplicated?"* to verify whether the study is duplicated in the studies' database, *"Is this study relevant?"* to certify that the study is relevant to the subject of this study after reading the study once and *"Does this study fit in the criteria?"* to verify whether the study meets the Selection Criteria mentioned in Table 2.

Table 3: Metadata information of each study

| **Info. retrieved** | **Explanation** |
|---|---|
| Database | Identifier of the database |
| Title | Study title |
| Authors | List of all authors |
| Type of forum | Journal/conference/workshop/symposium |
| Abstract | Study abstract |
| Keywords | Study keywords |
| Control 1 | Duplicate |
| Control 2 | Does not fit into criteria |
| Control 3 | Is relevant |

In this sense, Table 4 presents the studies retrieved from each base. Following the standard guidelines, we have read every title, abstract and keywords of the studies, answering the control questions (exclusion criteria) for them. If the study was duplicated, did not fit inclusion criteria or was not relevant for the review topic, the study was excluded. The reading process was initially conducted by one researcher, then verified independently by one other author.

Table 4: Search results.

| **Base** | **Studies Found** | **Selected Studies (Abstract + Keywords)** | **Selected Studies (Full Read)** |
|---|---|---|---|
| ACM DL | 449 | 43 | 13 |
| IEEExplore | 864 | 21 | 5 |
| Science@Direct | 528 | 20 | 2 |
| El Compendex | 25 | 10 | 1 |
| Scopus | 1594 | 38 | 6 |
| Total | 3460 | 132 | 27 |

From the total of 3460 studies found, our research selected 132 studies which would be fully analyzed. After fully reading these 132 studies (in a similar manner to the previous step where one of the authors read all studies and had other authors double-check them), we further reduced our list of studies to 27. It is important to note that the complete analysis of studies gave us better understanding of their research focus and thus enabled us to exclude 105 studies which were not related to our research questions.

### 3.2 Data Extraction and Classification

From the initial metadata assigned to the 27 selected studies in the spreadsheet, we have also added some additional attributes:

- Contribution Facet: type of contribution. Based on a work from (Shaw, 2003);
- Research Method: the research method applied (case study, survey *etc*);
- Research Type: type of research (based on work from (Wieringa et al., 2005));
- Study Quality: a grade from 0 to 10, based on the work from (Salleh et al., 2011). Details on Table 5;
- Contribution: the research contribution from the study.

### 3.3 Classification Scheme

Regarding study quality, we have assigned a score for each study, based on the work from (Salleh et al., 2011). In this sense, eight (8) questions were used to provide a grade to the work under analysis. Each question could be either completely answered, meaning the work will be assigned the complete score for the question, partially answered, meaning that the work will be assigned half the points for the question, and not answered, meaning that the work will be assigned zero points for the question.

We have chosen to better classify studies which were closely related to our research questions, thus each of these questions are worth 2 points. Once a study is graded, it is assigned a quality category, meaning that the study possesses:

- High quality: 8 to 10 points;
- Medium quality: between 5 and 8 points;
- Low quality: 0 to 5 points.

Based on the work from (Chanin et al., 2018a), we have created a completed classification scheme and present it in Table 5.

## 4 RESULTS

Based on the 27 selected studies, we have performed a classification based on the scheme presented in Table 5 and the results are presented in Table 6. Figure 1 presents the distribution of the analyzed work based on the year of publication.

Figure 2 summarizes the results from the systematic mapping from the facets of research type, focus and contribution of the 27 studies. The first publication found is relatively old, published in 2005, which indicates a spark of interest, but the figure indicates that starting 2018, the amount of studies stayed at a decent level (3.8 on average per year since 2018) which could be an indicative that the theme has become more relevant for the research community.

Table 5: Classification Scheme.

| Category | Description |
|---|---|
| ***Research Method Facet*** | |
| Case Study | Report from a specific situation being studied (Kitchenham et al., 1995). |
| Empirical Study | Study based on empirical evidence (Perry et al., 2000). |
| Experimental Study | Study in which an intervention is introduced to observe effects (Sjøberg et al., 2005). |
| Survey | Process to collect data, analyze it and report results (Pfleeger and Kitchenham, 2001). |
| ***Research Type Facet*** | |
| Evaluation Research | Evaluation of a method or technique in practice. |
| Experience Study | Personal experience of the author depicting how something has been done in practice. |
| Opinion Study | Personal opinion on a certain technique. |
| Philosophical Study | New way of looking at an existing context. |
| Solution Proposal | The proposition of a solution to a problem. |
| Validation Research | New techniques being implemented in experiments, simulations or in practice. |
| ***Focus Facet*** | |
| Classroom | The focus is on classroom education strategies for leadership teaching in SE. |
| Real Projects | The focus is leadership teaching in SE on real-world project execution. |
| Synthesis | Focus is a review of primary (or secondary) studies on leadership teaching in SE. |
| ***Contribution Facet*** | |
| Advice/Implication | Recommendations based on personal opinions. |
| Framework/Method | Framework/Method used to teach (or to learn). |
| Guidelines | Advices based on the research results. |
| Lessons Learned | Actionable advices derived from the obtained research results. |
| Model | Representation of a given context based on a conceptualized process. |
| Tool | Tools used to teach (or to learn). |
| ***Study Quality Facet*** | |
| References | Are the references adequate and well-cited? (1 point) |
| Goal | Is the goal clearly stated? (1 point) |
| Sample Observation | Data collection and sample strategy was carried out correctly? (1 point) |
| Research Method | The analysis methodology was well applied? (1 point) |
| Clear Description | Is the context of the study clearly described? (1 point) |
| Findings | Are findings credible? (1 point) |
| RQ1 | Does the study answer RQ1? (2 points) |
| RQ2 | Does the study answer RQ2? (2 points) |

Table 6: Overview of results.

| Authors [year] | Research method | Research type | Focus | Contribution | Study Quality |
|---|---|---|---|---|---|
| (Noguera et al., 2018) | Mixed | Experience Study | Teaching | Lessons Learned | 10 |
| (Khakurel and Porras, 2020) | Empirical Study | Experience Study | Real Projects | Lessons Learned | 10 |
| (Paasivaara, 2021) | Case Study | Experience Study | Real Projects | Lessons Learned | 9 |
| (Vivian et al., 2013) | Survey | Evaluation Research | Teaching | Guidelines | 9 |
| (Christensen and Paasivaara, 2022) | Empirical Study | Experience Study | Real Projects | Lessons Learned | 9 |
| (Fontão et al., 2019) | Mixed | Evaluation Research | Teaching | Lessons Learned | 9 |
| (Li et al., 2023) | Survey | Experience Study | Teaching | Lessons Learned | 9 |
| (Ceh-Varela et al., 2023) | Empirical Study | Experience Study | Teaching | Guidelines | 9 |
| (Libreros et al., 2020) | Experimental Study | Solution Proposal | Teaching | Framework/Method | 9 |
| (Heggen and Myers, 2018) | Case Study | Evaluation Research | Real Projects | Framework/Method | 8 |
| (Ha et al., 2019) | Mixed | Evaluation Research | Teaching | Lessons Learned | 7 |
| (Schneider et al., 2020) | Case Study | Validation Research | Teaching | Model | 7 |
| (Tenhunen et al., 2023) | Empirical Study | Solution Proposal | Real Projects | Lessons Learned | 7 |
| (Hogan and Thomas, 2005) | Survey | Evaluation Research | Real Projects | Guidelines | 7 |
| (Johnson et al., 2020) | Survey | Validation Research | Teaching | Framework/Method | 6 |
| (Kapitsaki and Loizou, 2018) | Survey | Experience Study | Real Projects | Lessons Learned | 6 |
| (Poženel, 2013) | Case Study | Evaluation Research | Teaching | Framework/Method | 6 |
| (Watson and Cutting, 2022) | Empirical Study | Evaluation Research | Real Projects | Guidelines | 6 |
| (Sundaram, 2023) | Survey | Experience Study | Teaching | Framework/Method | 6 |
| (Boiangiu and Stănică, 2019) | Empirical Study | Experience Study | Teaching | Model | 5 |
| (Paiva and Carvalho, 2018) | Empirical Study | Experience Study | Teaching | Framework/Method | 5 |
| (Budu, 2018) | Empirical Study | Validation Research | Teaching | Lessons Learned | 5 |
| (Peters and Moreno, 2015) | Empirical Study | Philosophical Study | Synthesis | Guidelines | 4 |
| (Kawano et al., 2019) | Empirical Study | Opinion Study | Teaching | Framework/Method | 4 |
| (Moh'd A, 2021) | Empirical Study | Philosophical Study | Teaching | Guidelines | 3 |
| (Moreno et al., 2022) | Empirical Study | Experience Study | Teaching | Lessons Learned | 2 |
| (Escudeiro et al., 2020) | Empirical Study | Experience Study | Real Projects | Framework/Method | 1 |

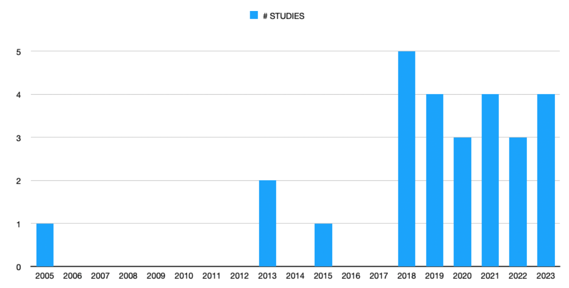

Figure 1: Distribution of selected studies by year.

In a general sense, majority of the studies found were classified as *"High"* with regard to our *"study quality"* facet. This indicates that the teaching of leadership in software engineering is a topic which is being undertaken by the scientific community and robust studies are addressing it. In addition, specifically regarding the Focus facet, we can clearly see that there are studies many studies being conducted with regard to teaching leadership in software engineering. This is an indicative that the software engineering education research community is interested in the topic and conducting efforts to teach leaderships skills in software engineering.

After fully reading and comprehending the selected studies, we have proceed to extract data to answer our proposed research questions. This discussion is presented next.

## 4.1 RQ1 - How Is Leadership Teaching Defined in Agile Software Engineering Education Environments?

There are many definitions to what could be considered leadership teaching the studies we have found. In general, we have found 4 categories of definition in the studies analyzed. Those define leadership as:

### 4.1.1 Part of Teamwork and Shared Among Team Members

(Noguera et al., 2018), based on (Eubanks et al., 2016), defined leadership as encompassing *"task management"* and *"team development"*.The study emphasizes teamwork, with leadership as an important component of teamwork. Specifically, the study proposed two roles in an agile software development educational context, Project Manager and *"Work-Cycle"* manager, where each one of these two roles is responsible for management and leadership tasks.

(Vivian et al., 2013) use the definition of leadership as being an emergent trait of self-organizing teams when conducting *"teamwork"*. This definition is based on the concept of *"Team Leadership"* from (Dickinson and McIntyre, 1997) and states leadership as a component of teamwork which *"involves providing direction, structure, and support for other team members. It does not necessarily refer to a single individual with formal authority over others; several members can show team leadership"*.

Similarly, (Poženel, 2013) defines leadership using the concept of shared leadership from (Moe et al., 2009) where among many specifics, important decision making is shared among all team members. Excessive centralization is discouraged (*e.g.*, decision making made by a single team member) as well as excessive decentralization (*e.g.*, tasks not being attributed to specific team members).

(Hogan and Thomas, 2005) acknowledge the existence of formal and informal roles regarding leadership, in which a formal role refers to when the role is explicitly assigned to a team member (*e.g.*, SM) and informal refers to when there is not explicit assignment of roles, but a team member exhibits leadership behavior and performs leadership tasks (this formal/informal dichotomy could be an indicative of the concept of shared leadership found in the other studies).

### 4.1.2 Associated with Scrum

(Paasivaara, 2021), in a study applying Scrum in education, has defined that the Scrum Master (SM) is a leading team member in the initial phases of the development, after which leadership can become shared among other team members. Furthermore, the study indicates that a shared leadership is an indication of a mature team. The definition of leadership as a role also appears in (Fontão et al., 2019), in which authors describes leadership as being a role of the SM in a Scrum team and further defining the SM as a *"technical leadership"*.

Furthermore, in (Li et al., 2023) which also add that some personality traits such as openness and agreeableness may be relevant for a leading SM. (Libreros et al., 2020) report a study in which Team Leader Rotation was adopted and Scrum teams had a team member denominated *"team leader"* (different from the SM and/or PO). As the study was conducted in a Scrum context, the team leaders in the study were acting as a complementary role to the SMs and the POs.

### 4.1.3 Representing the Group in Academic Settings

(Budu, 2018) used a definition of a member of the team who had the responsibility to present results for

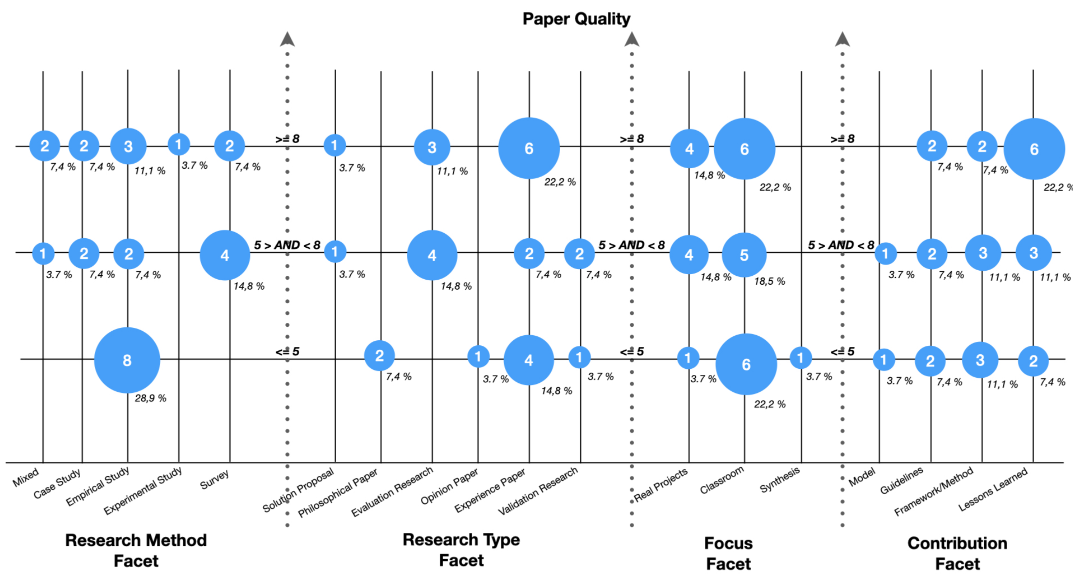

Figure 2: Summary of the studies. X axis represents facets of the studies (research type, focus and contribution) and the Y axis represents study quality

their working group to the class. This definition implies a very academic setting. (Khakurel and Porras, 2020) define leadership as a trait of being more *"responsible"*, managing the project and handling customer interactions. In the study, the role adopted by students participating the study who reported on leadership was of a project manager.

#### 4.1.4 Not Explicitly Defined

(Christensen and Paasivaara, 2022) created a definition based both on the work from (Matturro et al., 2019) and the context of their study. This concept can be summarized in *"the ability to lead and supervise others"*. The study was conducted in a course which students were intended to learn soft skills, among of which was leadership.

(Sundaram, 2023) defines leadership in the study by comparing the difference between a *"leader"* and a *"manager"*. A manager being the role which tends to *"push"* the team and focuses on the work being delivered, while the leader is stated to *"guide"* team members and to put more emphasis on aiding team members themselves.

(Kawano et al., 2019) applies the definition of leader for the *"next generation software development"*. This concept is not further specified, so it somewhat vague. In addition to this, the author also defines leadership as when a student is interested in leading junior colleagues.

(Moh'd A, 2021) defines leadership as being able to both manage and lead, in the context of a professional denominated software project manager. This definition is further specified as encompassing providing supervision, motivation, monitoring and keeping track of the project.

### 4.2 RQ2 - How Is Leadership Taught in Agile Software Engineering Education Environments?

Results indicate convergence on how leadership is taught in software engineering education. We have found 4 manners in which leadership is taught.

#### 4.2.1 Using Scrum

Majority of the studies that we have analyzed incorporate Scrum as their software development methodology and use its roles (Product Owner, Scrum Master etc.) to provide students with the opportunity to develop *"soft skills"*. Among these soft skills, there is leadership.

For example, (Noguera et al., 2018) have taught leadership in software engineering by incorporating agile in education, specifically, an adaption of the Scrum framework (Schwaber and Sutherland, 2011). (Paasivaara, 2021) has used professional agile coaches and communities of practice (CoP). CoP are exchange sessions that the students who are learning to be SM participate and exchange ideas. The projects developed in the are provided by real industry companies and external agile coaches help the assigned SM (who is a student of the course). (Christensen and Paasivaara, 2022) teach soft skills through a distributed software development course which applied a Casptone project. It used Scrum as its development framework. The projects were provided by real Danish companies. (Heggen and Myers, 2018) have taught through practical experience in real software projects that address community and/or university departments' needs. All development is performed using Scrum as the primary framework. (Fontão et al., 2019) applied project-based learning (PBL) in a Capstone project in combination with Scrum. (Li et al., 2023) also uses Scrum and proposes a multi-team Capstone project to provide real experience to students, where a project is developed by more than one team. (Libreros et al., 2020) have taught adopting team leader rotation in agile software development. Scrum is applied as the chosen development process. (Johnson et al., 2020) have taught through a course that simulates real-world environment. Scrum is applied as the chosen development process. (Paiva and Carvalho, 2018) have taught using Capstone Projects and Scrum software development. (Poženel, 2013) have assessed student who worked on large capstone courses. Scrum is applied as the chosen development process. (Sundaram, 2023) have taught using traditional teaching, but Scrum is studied and simulated by students.

#### 4.2.2 Using Active Learning Methodologies

Another finding is that leadership teaching is usually performed using active learning strategies (*e.g.*, PBL).

For example, (Ceh-Varela et al., 2023) used project-based learning in the development of a software project. The software development methodology applied by students is defined as *"relaxed plan-based model"*, a combination of cascade, spiral and prototype model. (Hogan and Thomas, 2005) have taught using problem based learning in addition to real projects that connect students to the industry. Agile software development is applied. Furthermore, (Fontão et al., 2019) applied project based learning in a Capstone project in combination with Scrum.

#### 4.2.3 Using Real Projects Either Originating from External Partners or Which Propose to Solve Real-World Problems

Real projects, either proposed by industry partners or solving real-world problems, being incorporated in the teaching process was also reported in our findings. For instance, (Khakurel and Porras, 2020) have taught through Capstone projects, with real projects from Finish companies. (Schneider et al., 2020) have taught through Capstone projects, adopting industry practices in the development process. The model adopted is inspired by Spotify's *"Tribes and Squads"* (Alqudah and Razali, 2016). (Watson and Cutting, 2022) have taught using capstone project. Projects could be from the industry Agile software development is incentivized. (Tenhunen et al., 2023) have taught through real world projects. This was achieved via a *"Software Development Academy"* (SDA), a startup concept inside the university with internal projects developed by student. Students are paid to participate in the SDA. (Kapitsaki and Loizou, 2018) have taught by working on Real project and in teams that mix undergraduate and postgraduate students. The integration of Minimum Viable Products and continuous experimentation in real-world projects has shown to be an effective strategy for teaching practical software engineering skills (Melegati et al., 2020).

#### 4.2.4 Using Other Learning Methodologies and/or Teaching Strategies

Two of the studies we have found to use other learning methodologies and/or teaching strategies methodologies for teaching. (Boiangiu and Stănică, 2019) present a conceptual model that can be instantiated depending on the *"type"* of general-purpose education envisioned. (Ha et al., 2019) used a concept named *"Conceive-Design-Implement-Operate"* approach (CDIO), with two levels. Level 1 addresses simple projects and fundamental knowledge, while level 2 addresses complex projects and advanced knowledge. (Vivian et al., 2013) have performed two online collaborative sessions where students had to solve a *"difficult"* problem by creating a document which contained answers. From the results and overall presentation of the ideas of the studies we found during the analysis step, we have proceeded to discussion on our findings.

## 5 DISCUSSION

In this section, we discuss some of the results obtained from our mapping from the perspective of our research questions and the general implications of these findings.

From the analyzed data, it was possible to find different definitions for the teaching of leadership in software engineering education. Many of the authors have defined leadership as a component that is integrated into teamwork and thus is shared among team members (Noguera et al., 2018; Vivian et al., 2013; Poženel, 2013; Hogan and Thomas, 2005).

Furthermore, Scrum has been a base for defining leadership *"roles"* in software engineering education.

Although using Scrum is expected as it the most used agile framework in the industry 63% of agile teams (of Agile, 2023), this could indicate that the roles provided by Scrum (*e.g.*, Scrum Master, Product Owner) serve as a comprehensive set of skills for students to develop leadership abilities. For example, practices such as Behavior-Driven Development (BDD) have been shown to enhance team collaboration and leadership development when integrated with agile frameworks like Scrum (Nascimento et al., 2020).

Our data also revealed that being a *"leader"* from a software engineering education perspective can also be associated with performing group presentations and interactions with external actors (Budu, 2018; Khakurel and Porras, 2020). This is a traditional approach to group education and revolves around assigning responsibility to a team member as a manner of teaching this skill to the assignee.

Another interesting finding was that in software engineering education, it is not unusual to define leadership more generically, such as *"being able to lead a team"*, with specifying the details of what exactly this implies (Christensen and Paasivaara, 2022; Sundaram, 2023; Kawano et al., 2019; Moh'd A, 2021). Further, provided that other authors propose leadership as a component of teamwork, this finding could be a symptom of the difficulty of isolating leadership from teamwork in software engineering education.

In terms of how to teach leadership in software engineering education, Scrum appeared as the most adopted development methodology (Noguera et al., 2018; Schwaber and Sutherland, 2011; Paasivaara, 2021; Christensen and Paasivaara, 2022; Heggen and Myers, 2018; Fontão et al., 2019; Li et al., 2023; Libreros et al., 2020; Johnson et al., 2020; Paiva and Carvalho, 2018; Poženel, 2013; Sundaram, 2023).

Finally, regarding active learning, many of the studies adopt active learning as the preferred teaching strategy. Collaborative approaches, such as Challenge-Based Learning (CBL), have proven effective in fostering leadership and teamwork skills within software engineering education (Chanin et al., 2018b). As these methodologies are student-centered, it is not surprising that they are adopted to provide students with practical experience with leadership.

## 6 LIMITATIONS

This study contributes insights into leadership teaching in agile software engineering. However, it is necessary to recognize its limitations regarding reliability, construction/internal/external validity, which may influence the interpretation and generalization of the findings (Runeson and Höst, 2009).

Construction Validity: The operationalization of key concepts may not fully capture the complex and multifaceted nature of leadership in software engineering. We minimize this by looking into the most commonly adopted digital libraries.

Internal Validity: While the systematic approach aimed to minimize biases and errors in selecting and synthesizing studies, it is possible that researchers might have been influenced to find leadership concepts in the analyzed studies. To mitigate this, initially one of the researchers conducted the analysis steps separately and independently. After results were obtained, an alignment meeting was conducted to discuss doubts and reach consensus.

External Validity: Generalization of our findings is constrained by the focus on agile software engineering education. This limitation restricts the applicability of our results to educational contexts in which agile software engineering is applied.

Reliability: The reproducibility of this study's findings may be influenced by the dynamic nature of software engineering and leadership practices, which are continually evolving. To mitigate this, we have provided all the parameters applied in our mapping study so that other researchers are able to replicate it.

## 7 CONCLUSION

This study presented the results from a systematic mapping about teaching leadership in software engineering education. The results were analyzed from the perspective of two researched questions.

Upon meticulous evaluation of 27 studies selected for their relevance to the proposed research topic, we discerned preliminary indicators highlighting key facets of leadership instruction within software engineering education. Firstly, it is discernible that leadership is commonly defined as an integral component of teamwork, distributed among team members and associated with Scrum, thereby providing students with practical experiences of group representation. Secondly, it is mostly taught using Scrum, involved in active learning strategies, such as problem-based learning, and incorporating real-world projects, either provided by external partners or aimed at addressing real-world challenges.

As for future work, we intend to propose a leadership teaching framework that incorporates these findings and perform a case study in software engineering education environment.

## ACKNOWLEDGMENT

This study was partially supported by the Ministry of Science, Technology, and Innovations from Brazil, with resources from Law No. 8.248, dated October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex.

## REFERENCES

Ahmed, F., Capretz, L. F., Bouktif, S., and Campbell, P. (2015). Soft skills and software development: A reflection from the software industry. *arXiv preprint arXiv:1507.06873*.

Alqudah, M. and Razali, R. (2016). A review of scaling agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology*, 6(6):828–837.

Athukorala, C., Perera, I., and Meedeniya, D. (2016). The impact of transformational and transactional leadership styles on knowledge creation in sri lankan software industry. In *2016 Moratuwa Engineering Research Conference (MERCon)*, pages 309–314. IEEE.

Barry, E. J., Mukhopadhyay, T., and Slaughter, S. A. (2002). Software project duration and effort: an empirical study. *Journal of Information Technology and Management*, 3(1-2):113–136.

Bass, B. M., Avolio, B. J., Jung, D. I., and Berson, Y. (2003). Predicting unit performance by assessing transformational and transactional leadership. *Journal of applied psychology*, 88(2):207.

Bass, B. M. and Bass Bernard, M. (1985). *Leadership and performance beyond expectations*. Free press New York.

Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley Longman.

Bickle, J. T. (2017). Developing remote training consultants as leaders—dialogic/network application of path-goal leadership theory in leadership development. *Performance Improvement*, 56(9):32–39.

Boiangiu, C.-A. and Stănică, I.-C. (2019). The mosaics model of educational approaches for teaching the practice of software project management. *Education Sciences*, 9(1):26.

Budgen, D., Turner, M., Brereton, P., and Kitchenham, B. A. (2008). Using mapping studies in software engineering. In *Ppig*, volume 8, pages 195–204.

Budu, J. (2018). Applying agile principles in teaching undergraduate information technology project management. *Int. Journal of Information and Communication Technology Education*, 14(3):29–40.

Ceh-Varela, E., Canto-Bonilla, C., and Duni, D. (2023). Application of project-based learning to a software engineering course in a hybrid class environment. *Information and Software Technology*, 158:107189.

Chanin, R., Sales, A., Pompermaier, L., and Prikladnicki, R. (2018a). A systematic mapping study on software startups education. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering*, pages 163–168.

Chanin, R., Sales, A., Santos, A. R., Pompermaier, L. B., and Prikladnicki, R. (2018b). A collaborative approach to teaching software startups: findings from a study using challenge based learning. In Sharp, H., de Souza, C. R. B., Graziotin, D., Levy, M., and Socha, D., editors, *Proc. of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pages 9–12. ACM.

Christensen, E. L. and Paasivaara, M. (2022). Learning soft skills through distributed software development. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*, pages 93–103.

da Silva, F. Q., Monteiro, C. V., dos Santos, I. E., and Capretz, L. F. (2016). How software development group leaders influence team members' innovative behavior. *IEEE Software*, 33(5):106–109.

Dewan, S. and Dewan, D. (2010). Distance education teacher as a leader: Learning from the path goal leadership theory. *Journal of Online Learning and Teaching*, 6(3):673–685.

Dickinson, T. L. and McIntyre, R. M. (1997). A conceptual framework for teamwork measurement. *Team performance assessment and measurement*, pages 19–43.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development.

Escudeiro, N., Barata, A., Escudeiro, P., Welzer, T., Almeida, R., and Papadourakis, G. (2020). Blended academic international mobility: tearing down barriers to mobility in a sustainable way. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 1434–1443. IEEE.

Eubanks, D. L., Palanski, M., Olabisi, J., Joinson, A., and Dove, J. (2016). Team dynamics in virtual, partially distributed teams: Optimal role fulfillment. *Computers in Human Behavior*, 61:556–568.

Faraj, S. and Sambamurthy, V. (2006). Leadership of information systems development projects. *IEEE Transactions on engineering management*, 53(2):238–249.

Fontão, A., Gadelha, B., and Júnior, A. C. (2019). Balancing theory and practice in software engineering education–a pbl, toolset based approach. In *IEEE Frontiers in Education Conference*, pages 1–8. IEEE.

Fowler, M., Highsmith, J., et al. (2001). The agile manifesto. *Journal of Software Development*, 9(8):28–35.

Furumo, K., de Pillis, E., and Buxton, M. (2012). The impact of leadership on participation and trust in virtual teams. In *Proceedings of the 50th annual conference on Computers and People Research*, pages 123–126.

Goyal, D., Cortinovis, R., and Capretz, L. F. (2022). A framework for class activities to cultivate responsible leadership in software engineering students. In *Proc. of the 15th Int. Conf. on Cooperative and Human Aspects of Software Engineering*, pages 96–101.

Ha, N.-H., Nayyar, A., Nguyen, D.-M., and Liu, C.-A. (2019). Enhancing students' soft skills by implementing cdio-based integration teaching mode. In *The 15th International CDIO Conference*, page 569.

Heggen, S. and Myers, C. (2018). Hiring millennial students as software engineers: a study in developing self-confidence and marketable skills. In *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*, pages 32–39.

Hidayati, A., Budiardjo, E. K., and Purwandari, B. (2020). Hard and soft skills for scrum global software development teams. In *Proceedings of the 3rd International Conference on Software Engineering and Information Management*, pages 110–114.

Hogan, J. M. and Thomas, R. (2005). Developing the software engineering team. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42*, pages 203–210.

House, R. J. and Mitchell, T. R. (1975). *Path goal theory of leadership*. Faculty of Management Studies, University of Toronto.

Johnson, W. G., Sunderraman, R., and Bourgeois, A. G. (2020). Teaching strategies in software engineering towards industry interview preparedness. In *Proceedings of the 9th Computer Science Education Research Conference*, pages 1–11.

Kapitsaki, G. M. and Loizou, S. K. (2018). Bringing together undergraduate and postgraduate students in software engineering team project: Experiences and lessons. In *Proc. of the 23rd Annual ACM Conference on ITiCSE*, pages 320–325.

Kawano, A., Motoyama, Y., and Aoyama, M. (2019). A lx (learner experience)-based evaluation method of the education and training programs for professional software engineers. In *Proceedings of the 2019 7th Int. Conf. on Inf. and Ed. Technology*, pages 151–159.

Khakurel, J. and Porras, J. (2020). The effect of real-world capstone project in an acquisition of soft skills among software engineering students. In *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*, pages 1–9. IEEE.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Tech report, Keele University and Durham University.

Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case studies for method and tool evaluation. *Journal of Software*, 12(4):52–62.

Kitchenham, B. A., Budgen, D., and Brereton, O. P. (2011). Using mapping studies as the basis for further research–a participant-observer case study. *Information and Software Technology*, 53(6):638–651.

Li, Y., Tan, C.-H., and Teo, H.-H. (2012). Leadership characteristics and developers' motivation in open source software development. *Information & Management*, 49(5):257–267.

Li, Z. S., Arony, N. N., Devathasan, K., and Damian, D. (2023). " software is the easy part of software engineering"–lessons and experiences from a large-scale, multi-team capstone course. *arXiv preprint arXiv:2302.05536*.

Libreros, J., Viveros, I., Trujillo, M., Gaona, M., and Cuadrado, D. (2020). Improving soft skills in agile software development by team leader rotation. In *International Congress on Information and Communication Technology*, pages 186–194. Springer.

Malik, S. H., Aziz, S., and Hassan, H. (2014). Leadership behavior and acceptance of leaders by subordinates: Application of path goal theory in telecom sector. *Int. Journal of Trade, Economics and Finance*, 5(2):170.

Marquez, J. M. D., Aguja, S. E., and Prudente, M. S. (2022). Evaluation of student leadership development amidst online distance learning set-up. In *Proceedings of the 2022 13th Int. Conf. on E-Education, E-Business, E-Management, and E-Learning*, pages 118–123.

Matturro, G., Raschetti, F., and Fontán, C. (2015). Soft skills in software development teams: A survey of the points of view of team leaders and team members. In *2015 IEEE/ACM 8th Int. Work. on Cooperative and Human Aspects of Soft. Eng.*, pages 101–104. IEEE.

Matturro, G., Raschetti, F., and Fontán, C. (2019). A systematic mapping study on soft skills in software engineering. *J. Univers. Comput. Sci.*, 25(1):16–41.

Melegati, J., Chanin, R., Sales, A., Prikladnicki, R., and Wang, X. (2020). MVP and experimentation in software startups: a qualitative survey. In *46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, Portoroz, Slovenia, August 26-28, 2020*, pages 322–325. IEEE.

Melegati, J., Chanin, R., Wang, X., Sales, A., and Prikladnicki, R. (2019). Enablers and inhibitors of experimentation in early-stage software startups. In Franch, X., Männistö, T., and Martínez-Fernández, S., editors, *Product-Focused Software Process Improvement - 20th International Conference, PROFES 2019, Barcelona, Spain, November 27-29, 2019, Proceedings*, volume 11915 of *Lecture Notes in Computer Science*, pages 554–569. Springer.

Modi, S. and Strode, D. (2020). Leadership in agile software development: a systematic literature review. In *Proceedings of the Australasian Conference on Information Systems*, pages 1–12.

Moe, N. B., Dingsøyr, T., and Røyrvik, E. A. (2009). Putting agile teamwork to the test–an preliminary instrument for empirically assessing and improving agile software development. In *Agile Processes in Software Engineering and Extreme Programming: 10th Int. Conf., XP 2009, Pula, Sardinia, Italy, May 25-29, 2009. Proceedings 10*, pages 114–123. Springer.

Moh'd A, R. (2021). Shifting the paradigms from teaching project management to teaching software project management at jordan university of science and technology based on the ieee software engineering management knowledge area. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1072–1078. IEEE.

Moreno, E. D., Fernandes, J. M., Alves, V., Leon Olave, M. E., and Afonso, P. (2022). Transforming ideas and developing entrepreneurship skills in computing sciences and informatics engineering courses. In *Proceedings of the 11th Euro American Conference on Telematics and Information Systems*, pages 1–6.

Nascimento, N., Santos, A. R., Sales, A., and Chanin, R. (2020). Behavior-driven development: A case study on its impacts on agile development teams. In *ICSE '20: 42nd International Conference on Software Engineering, Workshops, Seoul, Republic of Korea, 27 June - 19 July, 2020*, pages 109–116. ACM.

Noguera, I., Guerrero-Roldán, A.-E., and Masó, R. (2018). Collaborative agile learning in online environments: Strategies for improving team regulation and project management. *Computers & Education*, 116:110–129.

of Agile, S. (2023). The 17th state of agile report. Technical report, https://info.digital.ai/rs/981-LQX-968/images/RE-SA-17th-Annual-State-Of-Agile-Report.pdf?version=0.

Paasivaara, M. (2021). Teaching the scrum master role using professional agile coaches and communities of practice. In *2021 IEEE/ACM 43rd Int. Conf. on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 30–39. IEEE.

Paiva, S. C. and Carvalho, D. B. F. (2018). Software creation workshop: A capstone course for business-oriented software engineering teaching. In *Proc. of the XXXII SBES*, pages 280–288.

Perry, D., Porter, A., and Votta, L. (2000). Empirical studies of software engineering: a roadmap. In *Proceedings of the 1st Conference on the Future of Software Engineering*, pages 345–355.

Peters, L. and Moreno, A. M. (2015). Educating software engineering managers-revisited what software project managers need to know today. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 353–359. IEEE.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *12th international conference on evaluation and assessment in software engineering (EASE)*. BCS Learning & Development.

Pfleeger, S. L. and Kitchenham, B. (2001). Principles of survey research: part 1: turning lemons into lemonade. *Journal of Software Engineering Notes*, 26(6):16–18.

Podsakoff, P. M., Todor, W. D., Grover, R. A., and Huber, V. L. (1984). Situational moderators of leader reward and punishment behaviors: Fact or fiction? *Organiz. behavior and human performance*, 34(1):21–63.

Poženel, M. (2013). Assessing teamwork in a software engineering capstone course. *World Transactions on Engineering and Technology Education*, 11(1):6–12.

Przybilla, L., Präg, A., Wiesche, M., and Krcmar, H. (2020). A conceptual model of antecedents of emergent leadership in agile teams. In *Proceedings of the 2020 on Computers and People Research Conference*, pages 164–165.

Przybilla, L., Wiesche, M., and Krcmar, H. (2019). Emergent leadership in agile teams–an initial exploration. In *Proc. of the 2019 on Computers and People Research Conference*, pages 176–179.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164.

Salleh, N., Mendes, E., and Grundy, J. (2011). Empirical studies of pair programming for cs/se teaching in higher education: a systematic literature review. *Journal of Transactions on Software Engineering*, 37(4):509–525.

Sangwan, R. S. and Ros, J. (2008). Architecture leadership and management in globally distributed software development. In *Proc. of the 1st Int. Work. on Leadership and Manag. in Soft. Architecture*, pages 17–22.

Schneider, J.-G., Eklund, P. W., Lee, K., Chen, F., Cain, A., and Abdelrazek, M. (2020). Adopting industry agile practices in large-scale capstone education. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, pages 119–129.

Schwaber, K. and Sutherland, J. (2011). *The scrum guide*. Scrum Alliance.

Shaw, M. (2003). Writing good software engineering research papers: minitutorial. In *Proc. of the 25th Int. Conf. on Software Engineering*, pages 726–736.

Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K., and Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *Journal of Transactions on Software Engineering*, 31(9):733–753.

Sundaram, R. (2023). Incorporation of significant project experiences within the undergraduate engineering curriculum. In *2023 ASEE Annual Conf. & Exposition*.

Tenhunen, S., Männistö, T., Ihantola, P., Kousa, J., and Luukkainen, M. (2023). Software startup within a university–producing industry-ready graduates. *arXiv preprint arXiv:2301.07020*.

Van Kelle, E., Visser, J., Plaat, A., and van der Wijst, P. (2015). An empirical study into social success factors for agile software development. In *2015 IEEE/ACM 8th Int. Work. on Cooperative and Human Aspects of Soft. Eng.*, pages 77–80. IEEE.

Vivian, R., Falkner, K., and Falkner, N. (2013). Analysing computer science students' teamwork role adoption in an online self-organised teamwork activity. In *Proc.s of the 13th Koli Calling Int. Conf. on Computing Education Research*, pages 105–114.

Watson, E. M. and Cutting, D. (2022). Engagement contexts of software engineering education projects. In *31st Annual Conf. of the European Ass. for Ed. in Elect. and Inf. Eng. (EAEEIE)*, pages 1–6. IEEE.

Wieringa, R., Maiden, N., Mead, N., and Rolland, C. (2005). Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Journal of Requirements Engineering*, 11(1):102–107.