IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 27, 2021, accepted September 8, 2021, date of publication September 16, 2021,
date of current version September 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3113328

Software Engineering in Small Software Companies: Consolidating and Integrating Empirical Literature Into a Process Tool Adoption Framework

MICHEAL TUAPE[1], (Member, IEEE), VICTORIA T. HASHEELA-MUFETI[2], ANNA KAYANDA[3], JARI PORRAS[1], (Member, IEEE), AND JUSSI KASURINEN[1]

[1]Department of Software Engineering, Lappeenranta-Lahti University of Technology, 53850 Lappeenranta, Finland
[2]Department of Computing, Mathematical and Statistical Sciences, University of Namibia, Windhoek 10005, Namibia
[3]Information Systems Department, College of Business Education, Dar es Salaam 0255, Tanzania

Corresponding author: Micheal Tuape (micheal.tuape@lut.fi)

ABSTRACT Small software companies face numerous challenges of complexity, unstructured software development processes and scarce resources. This notwithstanding, the companies have dominated the software market by 80 percent. The practice and products of these companies are still persistently marred by quality issues arising from the processes, with evidence indicating that process tools do not fit the unique contexts in which they operate. Significant strides have been made to transform software development practice; however, the challenges are still apparent. Hence the need to establish how knowledge areas are applied in process practice, understand the context of software development and its implication in practice, how process tools are utilised in practice and evaluate the quality of research in software literature. The researchers undertook a systematic mapping study to determine the state of practice in the empirical literature on software engineering of SSCs by examining and classifying 1096 publications. Other than the finding that research quality was low and affecting generalisation and transferability, the results also revealed exciting findings, which we finally consolidated and integrated to develop two contributions (i) a software development process adoption theoretical framework that provides essential insights into understanding software development and (ii) a 3-point guideline for research quality. By solving the adoption of process tools in software development, this paper presents one of the most significant contributions to transforming practice in software development and research in small software companies.

INDEX TERMS Small software companies, software development practice, software process tools systematic mapping studies.

I. INTRODUCTION

Software has become entrenched in human life that society is increasingly dependent on software-intensive systems. Software facilitates a plethora of human activities such as business processes, governance, medicine, security, entertainment, and social interaction. Recent development in technology and growth in the software industry has been championed by Small Software Companies (SSCs), making up over 90% of the companies in the software industry [1]. Although software is a crucial driver to today's global economy [2] and SSCs contribute significantly to this, these companies' failure rate and inferior quality products are a point of concern [3].

The SSCs are business entities involved in producing software products, typically employing less than 50 employees, and their aim is to create one or a few software products for their customers [4]. In other definitions, the annual turnover is taken as an aspect to consider in defining SSCs. However, the threshold differs depending on economies. Owing to small sizes and character of SSCs, these companies are flexible in their operations, which is to a certain extent an advantage for them. This definition also fits very small

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

entities (VSE), though with fewer employees and start-ups at the initial stage of formation as part of the SSC. The European Union defines a small company as an enterprise employing less than 50 persons with a 10 million Euro annual turnover [5], [6]. In this study, we define SSCs as enterprises with less than 50 employees that build software products, including building and maintaining software solutions, web applications, corporate systems, and business intelligence tools. We, therefore, consider the three contexts to delimit SSCs from the context that does not fit our definition for this study.

In the past decade, research on software development by SSCs seems to have gained traction. Despite this attention, the studies do not sufficiently cover software engineering processes and practices, as Paternoster *et al.* [7] elaborates. To overcome the challenges of software production by SSCs, researchers developed several process tools like frameworks and standards [8]–[10] to improve software development processes. However, the SSCs find adopting these frameworks rather difficult, as cited by Alexanda *et al.* [11]. This has not solved the problem because the frameworks are used minimally. Researchers suggest that only about 7% of SSCs in practice have adopted software process improvement (SPI) standards and models [12]. Additionally, software development practice in SSCs faces a myriad of challenges [13], [14] to the extent that about 50–60 percent of software projects either partially or totally fail. However, other studies report a higher failure rate; for example, in 2018, the project failure rate was reported at 70 percent [15]. According to the project management statistics [16], the proportion of challenged projects has increased to 43 percent resulting from a change in the organisation's priorities, inaccurate requirements gathering, change in project objectives, and inadequate vision or goal. Other authors [3], [15], [17], [15] list scope creep as one of the most prevalent factors responsible for project failure.

The challenges have not stopped the SSCs from considerable progress by significantly dominating the highly competitive market, although sustaining this business environment with a good customer relationship is complex [18]. The companies have adapted to the rapid development of cheaper software products [19] to meet the market's volatile demand, which has affected the quality of the products.

Notably, SSCs play an essential role in the global economy [20] because of their ability to capture the markets that larger companies are incapable of reaching or could have rejected [4]. However, it is paramount that efficient software development processes [4] are used for the SSCs to attain a competitive advantage.

The success of software development in SSCs is dependent on the complexity of the system built, business risk, and the number of people involved in building the system in question, as cited by Wasserman [21]. The SSCs develop products under challenging environments of time pressure and limited resources while constantly searching for sustainable and scalable business models [22]–[24]. The size and flexibility may be an advantage to accommodate constant changes, and perhaps this explains the increased preference for agile methodologies, which are perceived as the most viable approach for SSCs. The SSCs are conveniently attracted to agile methods to benefit from shorter development schedules and greater delivery flexibility [18], [25], [26].

There is evidence of growth in the significance and number of SSCs. Recent statistics indicate that the percentage of people directly employed by small companies has risen to over 85 percent [5], of which software companies take up a considerable majority [27]. Notably, whereas evidence in literature presents SSCs as most prone to difficulty while producing software, the rapid growth of SSCs is accompanied by stiff competition that breeds good industry practice, which must be harnessed and tapped with an aim to improve the general software development practices.

It remains unclear why SSCs are tangled in this dilemma, even though software engineering has several knowledge areas listed in the software body of knowledge intentioned to guide processes and practice. Additionally, researchers have tried to develop tools such as frameworks and standards to support the processes of software practice evidenced in [28]–[32]. However, this attention seems not to have delivered the much-needed transformation in software processes for better software products as the SSCs require sometimes leading to frustration.

Imagine the frustration of applying software process tools in vain because of the complexity due to the operational context of an organisation. It becomes unfortunate because one must abandon the process tool, yet delivery pressure and time constraints are at the project doorsteps. This leads to the unestablished process that has been existent in software practice, especially for small companies that continuously try, in vain, to find a fitting process tools, as alluded to by [22], [33].

This has led to the development of several tools to streamline processes suggested by different researchers, for example, [18], [34], [35] for requirements and for [36], [37] project management. Solutions for process adoption are fragmented in the different empirical literature on software processes and hence the need to consolidate and integrate the fragmented findings for comprehensive theory development in software processes adoption for SSCs.

Theories are known to be important for theorising synthesising, preserving and communicating empirical knowledge. Notably, that the software industry lacks theories about software artifacts [38], and research is predominately prescriptive and method-focused [39]. This has led to the production of thousands of software development process tools like methods and models that remain unutilised.

The purpose of this study was to review software development literature specific to SSCs and published in the last 30 years to consolidate and integrate the findings fragmented from the empirical literature on software processes. To do this, we conducted a systematic mapping study to identify the software engineering gaps in research in relation to SSCs to improve software practice in SSCs and

propose a theoretical framework with the consolidated findings. We therefore investigated:

i Which knowledge areas have been used in practice of SSCs as reported in literature?
ii What terminologies are used to refer to the companies in the selected articles?
iii What software development frameworks/standards were used by the companies are reported in literature?

The rest of the paper proceeds as follows: Section 2 presents the related work; Section 3 is a description of the methodology used in this SMS, section 4 presents the results from the mapping study, followed by a discussion of the research questions presented in section 5 and lastly the conclusions of the study in section 6.

II. RELATED WORK

Although not explicitly studying SSCs, different researchers have expressed concern over the gaps in the literature on software development practice. For example, the different researchers in [2], [7], [40] explored the gaps in the literature using SMS to look at software engineering in start-ups, while others have explored gaps in software engineering in SSCs through Systematic Literature Reviews [1], [4] Studying these gaps helps researchers appreciate how research has transformed software engineering practices and is helpful to map existing studies.

Paternoster *et al.* [7] conducted a SMS to develop a classification schema, in which they ranked the selected primary studies according to their rigor and relevance; they also analysed and reported software development work practices in start-ups. This study aimed at structuring and analysing the literature on software development in software start-ups. They also determined the potential for technology transfer and identified software development work practices reported by practitioners. The researchers considered 43 primary studies to synthesise the available evidence on software development in start-ups. Their work found 16 studies entirely dedicated to software development in start-ups, of which nine studies exhibited high scientific rigor and relevance.

In a similar study by Berg *et al.* [2] in which 74 primary papers from 1994 to 2017, were assessed and compared to findings from previous mapping studies, a classification schema was developed, and the primary studies ranked according to their rigor. Their work discovered that most research is conducted within the SWEBOK knowledge areas of software engineering process, management, construction, design, and requirements, with evidence of a shift of focus towards process and management areas. The researchers noted that the primary papers published between 2013 and 2017 were of higher rigor, when compared to those published between 1994 and 2013. In addition, there was evidence of inconsistency in the characterisation of software start-up companies and recommended an alternative classification for use in future start-up research.

In another study, Klotins *et al.* [40] conducted a SMS on software engineering start-ups where they paid specific attention to identifying knowledge areas covered by software start-up literature of the 14 selected primary studies from 1994 to 2014. Their findings from the 11 knowledge areas covered reveal that inadequate research in the software development practice of SSCs is a contributing factor to the high failure rates. The same authors also highlight the challenges of software development in start-ups and add that the failure to engineer quality software products and inadequacies in applied engineering practices is not fully explored and yet this is another significant contributing factor for the high failure rates.

In addition, other literature studies have been conducted on SSCs [1], [4]; although these studies were not SMSs, the researchers showed software practice in SSCs, and in both cases, the studies pay attention to the challenges affecting the development practice. Tuape and Ayalew [1] 2019 conducted an SLR on SSCs from 1988 to 2018; this study reported the factors affecting software development in SSCs. Whereas they found that limited studies had been conducted on SSCs specifically, the authors report factors affecting software development processes. These factors include organisational, business, governance, human and technical factors, which converge with the SLR of Tripathi *et al.* [4]. Tripathi *et al.* in 2016, conducted a study in which they selected 41 primary studies from papers published between 2004 and 2014. Other than the challenges like process adoption, limited documentation, limited technical knowledge and capacity, gaps in communication, limited understanding, and commitment to quality, which seem to have a point of similarity with most empirical literature, this study also reported six process areas covered in the literature, which translates into eight knowledge areas.

All the studies significantly highlight the challenges in attaining quality software products by the SSCs. Interestingly, no significant attention is given to addressing the context of software development in attaining quality products and improving the process. However, adaptability of the process tools such as methods, standards and models is highlighted as a major challenge in transforming software development especially for SSCs.

III. METHODOLOGY

A systematic mapping study is a secondary study method that builds a classification scheme and structure in the research field of interest. This method was initially used in medicine; however, researchers have adopted SMS in software engineering in the recent past. According to Petersen *et al.* [41], software engineering researchers started adopting SMS when Bailey *et al.* [42] first reported a review of 138 papers in their study of evidence related to object-oriented design.

In this paper, we conducted the SMS following the guidelines of Petersen *et al.* [41]. Significant to this guide is the use of a study protocol that ensures that personal bias is eliminated. In dealing with human bias or what is also referred to as subjective vagueness, some authors critic the use of statistical techniques, arguing that statistics present

a limitation in dealing with the subjective vagueness and human biases, alternatively suggesting fuzzy mathematics as a remedy to such uncertainties in comparison to statistics [43].

While the processes can create room for subjectivity, caution was accordingly taken through strict adherence to the study protocol and applied qualitative methods as a supplementary approach. Additionally, the data collected in this study are not subjective data and cannot therefore have any form of subjective vagueness as similarly observed in other mapping studies [2], [7], [40] that have used statistical meta-analysis.

The use of fuzzy mathematics in [44] and [45] is observed as a measure to mitigate subjectiveness and personal bias where the data collected are subjective, although the latter is a systematic literature review that also used the Kitchenham guidelines [46] in which the use of statistical meta-analysis is advised.

We selected 77 primary articles published in 4 databases over 30 years from 1990 to 2021. The steps undertaken in this study are illustrated in Figure 1 and explained in the different subsections; subsection A covers the SMS planning, the study design is covered in subsection B, the search extraction is discussed in subsection C and subsection D discusses the process of reporting the SMS.

A. PLANNING

During the planning phase, the researchers defined the need for the study; established a research protocol to ensure that the research questions are developed as planned, the planning of a search strategy for the study, and the methods to extract data and report the results were drawn.

1) PROTOCOL DEVELOPMENT

The systematic mapping study protocol is a step-by-step guide for conducting the study that describes the rationale and planned strategy. The protocol was prepared before the review started to guide the study, methods, and steps used in the study. The protocol was also necessary to reduce the possibility of any bias from the researchers. In this study, a protocol was developed, the senior researchers reviewed and approved it before the commencement of the study.

2) RESEARCH QUESTION (SCOPE)

These questions help develop a scope for the study; research questions are the core of the systematic mapping study; the questions streamline the study's overall purpose. In addition, it helps focus the study, determining the method and strategy to use while guiding all inquiry, analysis, and reporting stages.

This study was driven by the goal to understand how software engineering in SSCs is supported. To pursue this goal, we sought answers to the following research questions:

a: RQ1: HOW HAS SOFTWARE PRACTICE IN SMALL SOFTWARE COMPANIES UTILISED THE SOFTWARE ENGINEERING KNOWLEDGE AREAS IN THE ISO/IEC TR 19759: 2015?

Knowledge areas are a vital realm of knowledge with which all software engineers should be acquainted. This research question focuses on evaluating the extent to which the SSCS utilise software engineering knowledge areas in practice. This will help establish which knowledge areas have been used most in software development by the SSCs and highlight those that have not been used in practice. For example, Berg *et al.* also used knowledge areas [2] that systemised and prioritised software engineering processes, resulting in successful software execution.

b: RQ2: WHAT TERMINOLOGIES ARE USED TO REFER TO THE COMPANIES THAT HAVE BEEN STUDIED IN THE LITERATURE?

This question focuses on the terminologies used in literature to refer to SSCs, premising on the lack of a proper classification taxonomy for SSCs, leading to the researcher's usage of different terminologies inconsistently to refer to the software companies. Current literature refers to these companies in some cases as software start-ups and in other cases as small and medium enterprises. Paternoster *et al.* [7] also cite similar situations regarding many unclear classifications of small companies.

c: RQ3: WHICH SOFTWARE DEVELOPMENT FRAMEWORKS/STANDARDS DO THE COMPANIES USE?

This question draws the attention of the researchers to investigate the software frameworks are reported as used by SSCs in literature. Frameworks/standards are used to improve the efficiency of processes used in creating and maintaining software; Therefore, they are expected to be helpful during software development. However, the literature indicates that SSCs have failed to adopt frameworks, hence compromising the ability to improve developer productivity, quality, reliability, and robustness of software products. Understanding the most used frameworks will enable further investigation into the insufficiency of the frameworks.

B. STUDY DESIGN

1) SEARCH STRATEGY

Initial pilot searches were conducted to choose suitable keywords for the construction of the search string. This process was used to determine a sound approach in selecting the primary studies from the databases giving optimal results and enabling the choice of the four databases used in the study.

2) KEYWORDS

Identifying keywords is important in developing a schema to guide selecting relevant papers for review. The keywords used to generate the search string guaranteed that the relevant

TABLE 1. Search string used in the study.

Keywords	*Synonyms*
Software Engineering	development, requirements, design, construction, testing, management, maintenance, configuration management, process, models and methods, quality, engineering professional practice
Small Software Companies	start-up companies, Small and Medium Enterprises, Very Small Entities,

```
("Software") AND ("Development" OR "Requirements" OR "Design" OR "Construction" OR "Testing" OR "Maintenance" OR "Processes" OR "Configuration Management" OR "Organization" OR "Models" OR "Methods" OR "Quality" OR "Professional Practice") AND ("SME" OR "SSCs" OR "Very Small Entities" OR "Start-ups")
```

Listing 1. Search string used in the study.

papers are considered in the study. The trial searches were conducted to identify the keywords most used in studies on software engineering in SSCs, VSE, SMEs, and software start-ups. The keywords of the articles were identified and validated with the knowledge areas in the software body of knowledge presented in Table 1.

3) DEVELOPING THE SEARCH STRING

From SWEBOK ISO/IEC TR 1975:2015, we adopt the critical knowledge areas in software engineering to use as synonyms for software engineering. Twelve knowledge areas identified from the keywords were selected as most prevalent in the literature. These include software requirements, software design, software construction, software maintenance, software testing, software configuration management, software management, software process, software models and methods, and software quality. This is seen in similar SMSs in software engineering by other researchers Klotins *et al.* [40]. These formed the first part of the search string, the terminology software was then coined with the Boolean "AND" to the terminology identified from the keywords to synonymise engineering, and we used the Boolean OR as presented in Listing 1.

The first part of the search string connected to the second part of the string with the Boolean AND while the terms "small software companies" was synonymised with software start-up, SMEs, and VSEs all connected with the Boolean OR most used by researchers to refer to the same thing. Although not precisely researching SSCs but a segment of it, other researchers, Klotins *et al.* [40], use VSEs as synonyms of software start-ups.

C. SEARCH EXECUTION

Databases were selected following the guidance of Petersen *et al.* [41], based on the ability of the database to handle complex search queries and the history of usage

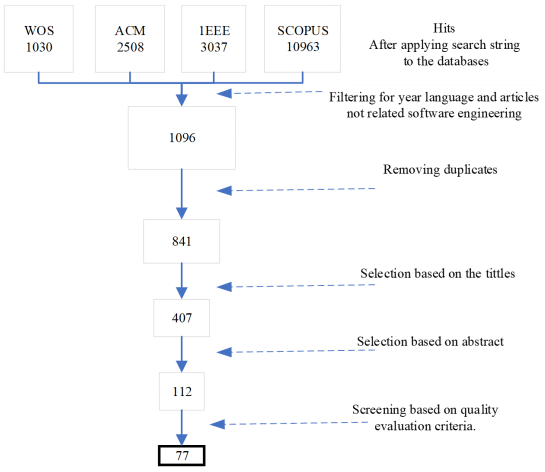


FIGURE 1. Steps of the study selection process.

TABLE 2. Inclusion and exclusion criteria.

Inclusion criteria	*Exclusion criteria*
The study was conducted within the last 30 years, 1990-2021	The paper was published in languages other than English.
The study is an empirical study (Primary study)	The paper is not peer-reviewed.
The Paper is about Software Engineering and any of the knowledge areas as software engineering processes of interest in the study.	Papers considered not to cover software engineering and the related knowledge area in SSCs.
The papers are about SSCs and their variants of VSE, start-ups, and SMEs.	The papers are not available as a complete article.
	Duplicate papers that appear in at least two of the databases considered for this study.

by researchers in software engineering related systematic mapping studies. Typical examples where these databases are used are Paternoster *et al.* [7], Neto *et al.* [47], and Gupta *et al.* [48], who have used all the four applied in this study, among others, and Berg *et al.* [2], use 2 of the databases.

1) STUDY SELECTION

The primary articles to consider for this study were selected through 6 stages illustrated in Figure 1. First, the search string returned 16536 hits; a filter for the year, language, relevance and if the study was a primary study returned 1096; filter for duplicate studies returned 841 non-duplicates based on the criteria in Table 2; after reading the titles, 407 articles were returned; a read of the abstract 112 articles were selected presented in Table 3. The selected 112 articles were then subjected to the quality evaluation process in Table 4.

2) DATA EXTRACTION

a: GENERAL DATA

For the general data, we collected details of the authors, the papers, the title of the article, year of publication, name of database and abstract of the study.

TABLE 3. Stages of screening of the selected primary studies.

Database	Hits	Filters	Titles	Abstracts	Quality
ISI Web of Science,	1030	216	117	57	36
ACM Digital Library	2508	136	36	7	6
IEEE Xplore,	3037	132	54	13	8
Scopus	10963	612	200	35	27
Total	16538	1096	407	112	77

b: SPECIFIC DATA

Specific data collected included knowledge areas used in the software development process described in the empirical literature, the research contribution described in the studies, category of the research described in the empirical literature, terminologies used to describe the software company in industry as reported in research, and software development frameworks used by the SSCs as reported in research.

c: MEANS OF DATA EXTRACTION

Data extraction was done with the aid of basic tools like google forms for collecting the data and eventually extracted in an excel sheet. The excel sheet supports the process using color coding tools for the initial exclusion based on the exclusion criteria.

3) QUALITY ASSESSMENT

a: RIGOR AND RELEVANCE

The selection of the primary studies was based on the evaluation for scientific rigor and industrial relevance, consistent with the proposal of Ivarsson and Gorschek [49]. The selected articles were arrived at through the quality evaluation criteria assessing for rigor and relevance using a rubric scale shown in Table 4. In this criterion, the 112 candidate papers screened using abstracts were subjected to an evaluation that realised 77 primary papers. This meant that the selected study must impact the industry; it was incumbent upon the authors of the paper in question to provide tangible evidence of the advantages of using the ideas of the research done. Ivarsson and Gorschek propose a systematic and validity model in which they guide the evaluation of software engineering papers for rigor and scientific relevance. This model provides for rules and a mechanism of applying metrics for measuring rigor and relevance; the model also splits the two components of rigor and relevance into different features and measures how they are reflected in the studies. Table 4 (a) and (b) describe the features and metrics for evaluating rigor and relevance adopted from the Ivarsson model.

Rigor is the precision, exactness of the study's research method, and how the study is presented. It influences the way practitioners perceived the results of the study and help to determine relevance. On the other hand, relevance is the realism of the environment in which the research is conducted and how the study is responsive to challenges in software engineering.

TABLE 4. A rubric scale is applied for the evaluation of rigor and relevance.

(a)

Aspect	Strong description (1)	Medium description (0,5)	Weak description (0)
Context described (Mandatory as strong)	the description is to the satisfaction clarity to compare with the context of the study of which the article is a selected candidate paper.	Context is mentioned, however, not described for adequate understanding and comparison with context under study.	No specific description of context is identified in the candidate paper.
Study design described	A description of the study design is evaluated for the variables measured, the sample selection, and the study's controls.	A limited description of the study design	There is no description of a study design for the study.
Validity	A satisfactory discussion of the threats to validity and how they have been mitigated	Mention and description of the threats to validity with less satisfactory mitigative measures	No mention of the likely threats to validity posed to the study.

(b)

	Contribution to relevance (1)	No contribution to the relevance (0)
Subjects	The subjects considered while evaluating the study are actual industry practitioners.	The evaluation was done using subjects who are not practitioners.
Research method	The research method used in the study including case studies, descriptive studies, surveys, grounded theory, experiments, exploratory studies, conceptual analysis, design science, ethnography, observations, means-end analysis, and mapping method	The research method used in the study does not involve actual industry settings Lab experiments, conceptual analysis (Mathematical)
Context (Mandatory as relevant)	Evaluation of the study based on the context of software companies under this study. (companies with less than 50 persons and are either VSEs, SSCs, Start-ups or SME)	Studies are evaluating companies with more than 50 staff. Start-up studies evaluating start-ups with more than 50 staff, or SME evaluating companies, some of which have more than 50

This quality criterion for rigor and relevance, applied through this process, was undertaken by three researchers using the metric and criterion in Table 4 (a) and (b) for rigor and relevance. Each component had to fulfil at least one mandatory feature attracting the respective scores and an extra score for the other features.

TABLE 5. Research contribution with its description as adopted from Shaw [50] and research classification schema as proposed by Wieringa *et al.* [51].

Contribution	*Description*
(a) Research contribution	
Models	This is the representation of an observed reality by concepts or related concepts after a conceptualisation.
Theories	these are constructs of cause-effect relationships of the determined result.
Frameworks/Methods	These are models related to constructing software or managing software development processes.
Guidelines	These are lists of advice, synthesis of the obtained research result.
Lessons learnt	These are sets of the outcome directly analysed from the obtained research result.
Advise/Implications	These are discursive and generic recommendations, deemed from opinion.
Tools	Technologies, programs or applications used to create, debug, maintain or support software development processes.
(b)Research classification	
Validation Research	A new investigated technique that has not been implemented in practice.
Evaluation Research	A Methodology implemented in practice followed an evaluation of methodology showing how the study was conducted and the consequences on the implementation (pros and the cons).
Solution Proposal	A solution to a problem is introduced and explained in detail. A description of the benefits and an analysis of its applicability are required.
Philosophical Papers	A new point of view or even a controversial approach is defined.
Opinion Papers	Personal opinions are the main contributions of these papers. There is no scientific approach to the results.
Personal Experience Papers	These papers explain what and how something is done in practice. Of course, it has to be the personal experience of the author.

D. REPORTING OF THE SYSTEMATIC MAPPING STUDY

1) DATA RETRIEVAL AND CLASSIFICATION

We tabulated the results from the study in terms of knowledge areas, the context of the companies (what terminology was used to refer to the company) and software development frameworks identified by the empirical literature. We also used classification facets of the research contribution proposed by Shaw [50] and research classification schema defined by Wieringa *et al.* [51] shown in Table 5. as useful in determining the quality of the selected articles.

2) ANALYSIS

The data is tabulated showing the primary studies spread over the years of publication. To answer research question 1, we categorised the selected primary studies on SSCs using the knowledge area of the ISO/IEC TR 1957:2015. We also mapped the research contribution over the years of study. In RQ. 2, we categorised the terminologies used to describe the software companies reported in the primary study. Finally, to answer RQ. 3, we categorised and mapped the software development frameworks/standards utilised by the software companies cited in the empirical literature.

Qualitative and quantitative methods of analysis were applied to the data and presented using graphs, charts, and matrix bubble charts to illustrate our findings, which ultimately consolidated and integrated the fragmented findings from the empirical literature on software processes.

IV. RESULTS

This section presents the results of the general findings, answering the research questions according to the study's overall objective. The results are presented in 3 subsections: Subsection A presents the general finding of the SMS; Subsection B presents the utilised knowledge areas by the companies cited with their respective research contribution and categories aspects, the terminologies used to refer to the companies and the companies utilisation of software development process frameworks/standards. Finally, sub-section C presents the evaluation of rigor and relevance.

A. GENERAL DATA

Figure 2 shows the number of studies published on practices of small companies in the period 1990-2021; of the 77 studies selected, 76 are concentrated in the last 20 years as the first ten years presents with only 1 study. The last 5 years between 2016 and 2021 present with over 60 percent of all the studies. The years 2016 and 2017 are the years with the highest frequencies, while no studies are presented for the years 1990-1994, 1996-1999 and 2013.

Research contribution aspects described in Table 5(a) presents the value research adds to the software engineering practice. This contribution has an order of importance and is classified as a weak and strong contribution. The weak contribution includes advice implication, guidelines, tools, and lessons learnt, while vital contributions are framework, theory, and models. Figure 3 shows the frequency distribution of the 77 primary articles selected to the seven research contribution aspects. For example, lesson learnt (32), frameworks/standards (13) and models (10) presenting 42, 17 and 12 percent, respectively. The other four research contribution aspects: guidelines, advice implications, tools, and theories share the remaining 29 percent with 4, 5, 6 and 7, respectively.

Table 5(b) presents the research category, distinguishing between the different types of studies. This is an abstraction from the exact research methodology as adopted from the work of Wieringa *et al.* [51]. The research categories identified in the selected articles are presented in Figure 4 with the following frequency distribution; validation (11),

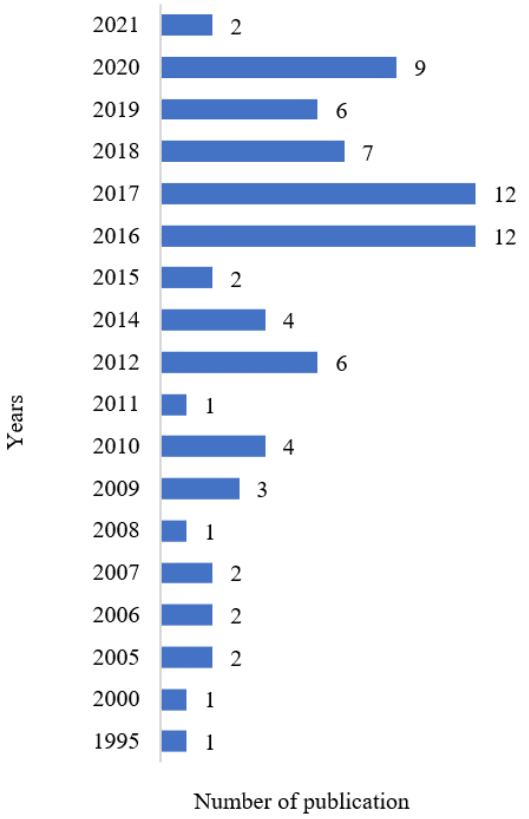


FIGURE 2. Publication frequency, 1990-2021.

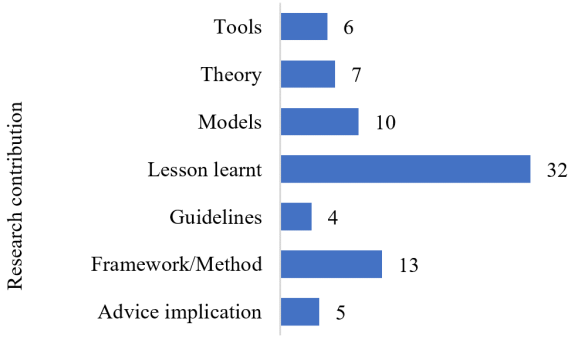


FIGURE 3. Frequency distribution of research contribution.

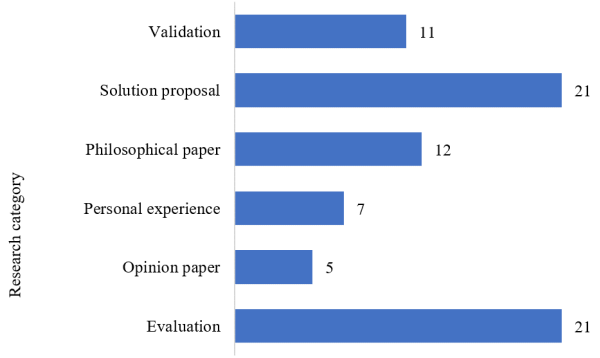


FIGURE 4. Frequency distribution of research category.

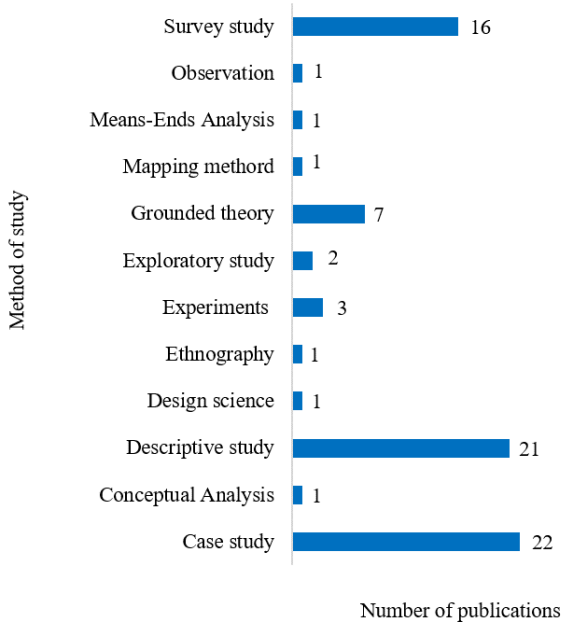


FIGURE 5. Study methodology frequency.

solution proposal (21), philosophical papers (12), personal experience (7), opinion papers (5) and evaluation (21).

Figure 5 presents the frequency distribution of the research methods identified in the primary studies selected for this SMS. The results show case studies (22), descriptive studies (21), surveys (17), grounded theory (7), experiments (3), and exploratory studies (2). The rest included conceptual analysis, design science, ethnography, observations, means-end analysis, and mapping method, each with one publication.

B. KNOWLEDGE AREA, CHARACTERISTICS, AND FRAMEWORK

The specific points of interest to this study, namely knowledge area, the context of the companies and the framework/standards are presented in Table 6 (a), (b), (c). In addition, each of the areas studied to answer the research questions is presented with the respective category, the frequency, and the article's references that cover the specific facet under a point of interest.

Figure 6 (a), (b) and (c) further presents the above data as bar charts to aid visualisation. The charts illustrate the years with the respective articles and the facets covered with percentages of the studies in the specific facet in that particular year.

The general landscape of results from the selected studies over the years from which the articles were published is presented in Figure 6 with the knowledge area, terminologies used to refer to the software companies, and the utilised framework/standards denoted as 6 (a), (b) and (c) respectively. An overall increase in research is evidently observed after 2010, and most of the research activity in all the aspects presented as getting visible in empirical literature

TABLE 6. Studies covering respective Knowledge areas, classification and frameworks used. (n=77).

No	Category	Frequency	Reference
(a) Knowledge area			
1	Development Design and Construction	33	[52]–[57][23][22][58]–[67][3][27], [68]–[81]
2	Project Management	16	[82]–[97]
3	Quality	7	[98]–[104]
4	Processes	5	[105]–[109]
5	Models and methods	5	[110]–[114]
6	Maintenance	4	[115]–[118]
7	Requirement engineering	3	[35][34],[18]
8	Testing	3	[119]–[121]
9	Management	1	[122]
(b) Naming of the companies			
1	VSE	27	[3], [27], [59], [61], [66], [68]–[70], [74], [77], [79], [84]–[89], [100], [104], [108], [109], [111–[114]
2	Start-up	17	[18], [34], [35], [65], [67], [71], [75], [76], [78], [81], [95]–[97], [106], [116]–[118]
3	SSCs	17	[22], [23], [52], [55], [56], [58], [62], [64], [72], [80], [93], [94], [98], [103], [110], [119], [122]
4	SME	16	[53], [54], [60], [63], [73], [82], [83], [90]–[92], [101], [102], [105], [107], [115], [120], [121]
(c) Framework/standards			
1	None	36	[18], [22], [31], [34], [35], [60], [63], [65], [77], [75], [78], [80]–[84], [90], [91], [93]–[99], [101], [106], [115]–[123]
2	ISO/IEC 29110	25	[3], [27], [58], [59], [61], [64], [66], [68], [69], [74], [76], [77], [79], [85]–[89], [104], [108], [109], [111]–[114]
3	CMMI	7	[53]–[55], [102], [105], [107], [110]
4	ISO/IEC 15504	3	[23], [62], [100]
5	ISO/IEC 12207	3	[56], [57], [92]
6	ISO/IEC 25010	1	[70]
7	ISO 9000	1	[103]
8	CMM	1	[52]

significantly after 2016. Other than process frameworks, which were evenly distributed in the first 10 years, the other areas of interest had mostly two facets dominate the first 10 years. The knowledge areas and the use of different terminologies are random because of increase in research, while the process frameworks are first seen as random and later tended to be dominated by the ISO/ IEC 29110 after its introduction in 2010.

1) KNOWLEDGE AREA

A close look at the knowledge areas of software engineering used by the companies as reported in the selected articles indicates that general software development and project management are most prominent in the knowledge area frequency distribution, as presented in Figure 7.

The data detail specifically indicates the most covered knowledge area of software engineering by the articles selected in the study. Figure. 7, illustrates 3 most dominant knowledge areas presenting up to 46 studies identified on the knowledge areas of software development design and construction (33), project management (16), and quality (7). The remaining 31 studies covered the knowledge areas of process (5), models and methods (5), maintenance (4), testing (3), requirements (3) and management (1) with the respective number of studies. Although the knowledge areas that ultimately are observed at a point in time are presented decimally in the empirical literature, the initial 10 years show that these knowledge areas are not reflected in the empirical studies around this time. Figure 6(a) shows the studies in the years between 2016 and 2021 present interest in the knowledge area of quality, requirements, models and methods, and maintenance. Although the numbers are minimal, the growing interest of researchers is important for the industry and for the transformation of software development practice in SSCs.

2) TERMINOLOGIES USED TO REFER TO THE SOFTWARE COMPANIES

The terminologies used to refer to the companies covered in this systematic mapping study is illustrated in Figure 8. The majority 27 studies presented the companies as VSEs; the remaining 50 studies are fairly distributed with almost equal numbers amongst the 3 other terminologies referring to small companies, namely, start-ups (17), SSCs (17) and SMEs (16). The two other terminologies, SME and the SSCs are reflected in the selected studies although the first 20 years portray minimal research. However, the overall outlook of the terminologies used to refer to the companies as covered in the selected years is presented in Figure 6(b), in which VSE and start-up depicted a significant increase from the year 2010.

3) FRAMEWORKS/STANDARDS

The frequency of publication in relation to frameworks/standards covered in the systematic mapping study presents; 36 articles identified as not mentioning any process frameworks/standards used in the SSCs studied. The remaining 41 articles used frameworks/standards with ISO/IEC 29110 being dominant, indicating 25 studies reporting reference to this framework/standard in software practice.

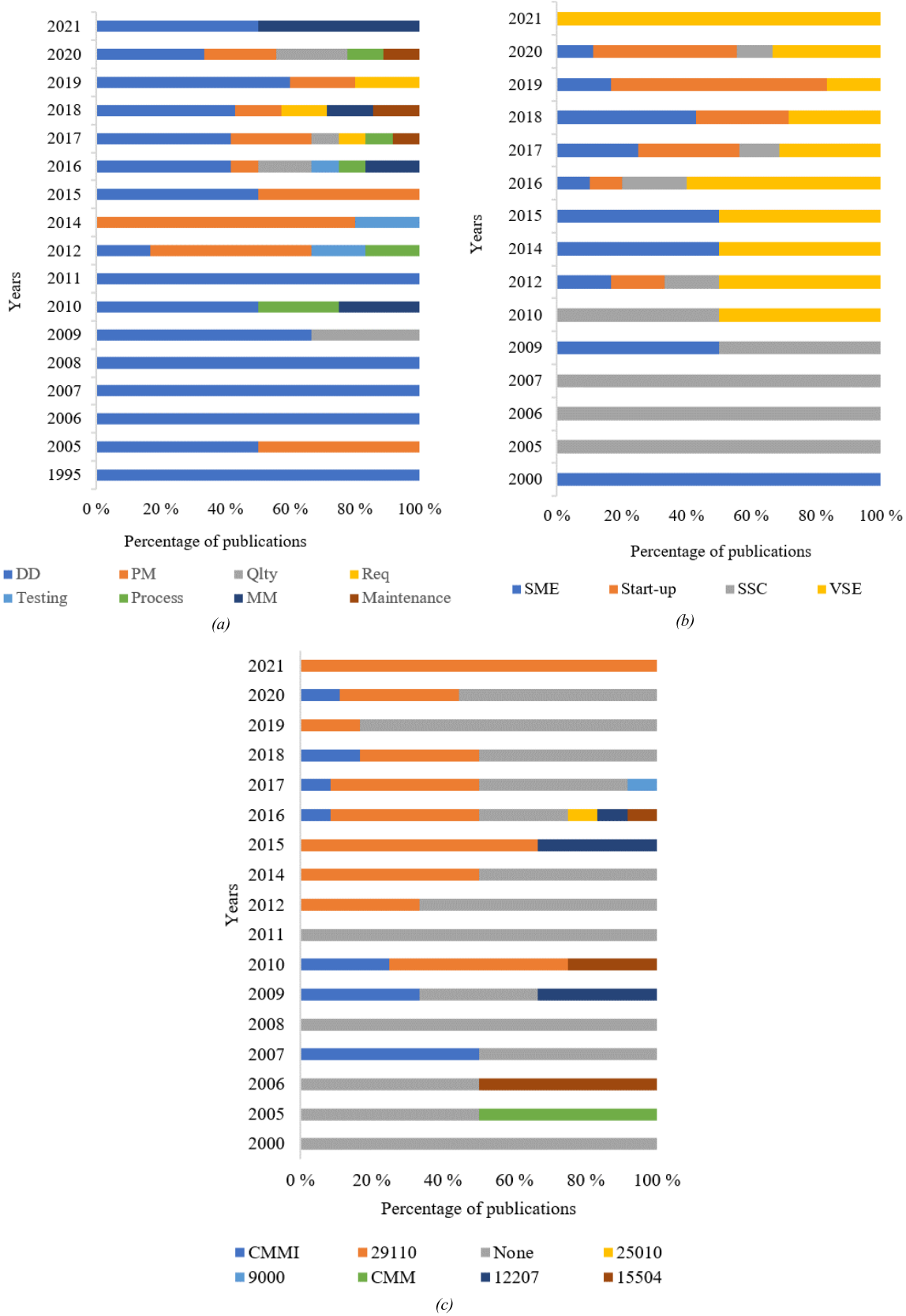


FIGURE 6. (a)The key knowledge areas covered in the primary studies, (b) names used to refer to the companies and (c) frameworks/standards used by the companies reported in the studies over the years.

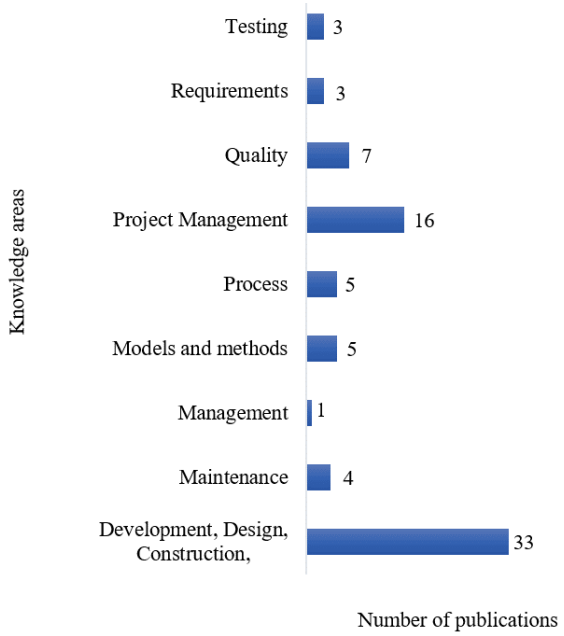


FIGURE 7. Knowledge area frequency.

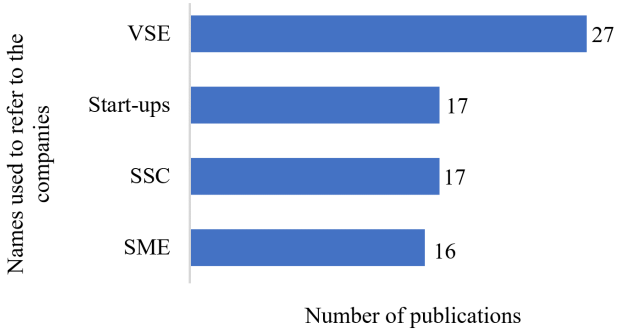


FIGURE 8. The frequency of the terminologies used in referring to small software companies in the systematic mapping study (n = 77).

The CMMI followed, indicating 8 articles reporting its usage in SSCs; the remaining 3 frameworks/standards were recognised as cited in 7 articles with ISO/IEC 15504(3), with ISO/IEC 12207(3), and with ISO 9000 being documented in 1 article as presented in Figure 9.

C. RIGOR AND RELEVANCE

Table 7. presents the evaluation scores of the rigor and relevance of the selected studies after the application of the rubric scale evaluation criteria in Table 3 (a) and 3 (b). The scores attained from the 4 features of rigor generate totals 1.5, 2, 2.5, and 3 with frequencies 3, 54, 5, and 15, respectively.

In the case of relevance, total scores of 2, 3, and 4 with frequencies of 14, 49 and 15 accordingly.

An evaluation of rigor and relevance of the selected primary articles is illustrated using the bubble plots. To evaluate rigor, we map the scores of the rigor evaluation to the research contribution and research category in Figures 10 and 11,

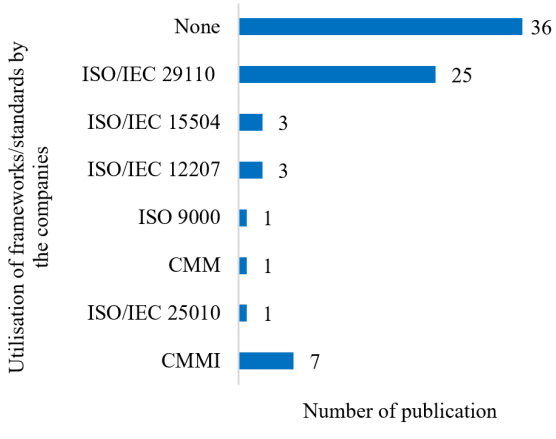


FIGURE 9. Frameworks/standards frequency.

TABLE 7. Evaluation scores for rigor and relevance.

	Total scores from evaluation	*Frequency*	*Total*
Rigor			
	1.5	3	
	2	54	
	2.5	5	
	3	15	
			77
Relevance			
	2	13	
	3	49	
	4	15	
			77

respectively. Then to evaluate relevance mapping scores of relevance evaluation to research contribution and research category is presented in Figures 12 and 13.

The results indicate that most papers are lessons learnt which are considered to have low rigor. This is consistent with the findings of other researchers like Klotins *et al.* [40], who reveal that most of the papers have high relevance, although the same authors also evaluate rigor of their studies and find more of experience reports that are also considered to have low rigor.

V. ANALYSIS AND DISCUSSION

In this section, we synthesise the extracted data to consolidate and integrate the fragmented findings from the empirical literature on software processes, looking specifically at the knowledge areas covered in practice as reported in the research, the terms used to refer to the software companies, and the frameworks/standards used while developing software in the companies covered in the selected studies. We also mapped each of the facets to the research contribution methods of study and research category. The analysis and

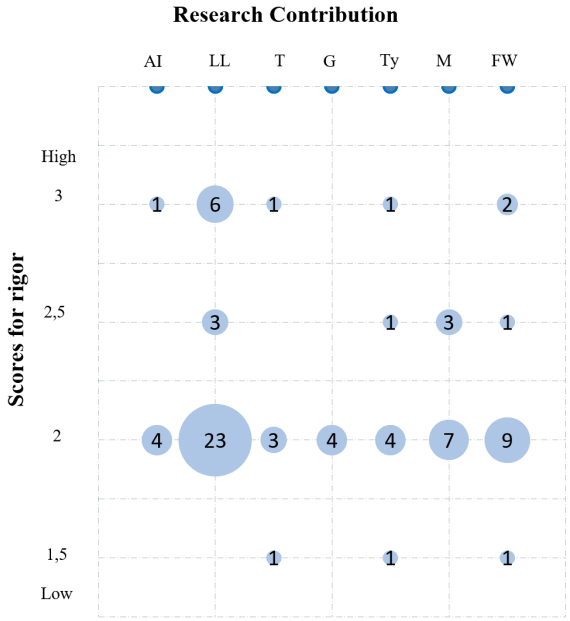


FIGURE 10. Systematic mapping of rigor on research contribution.

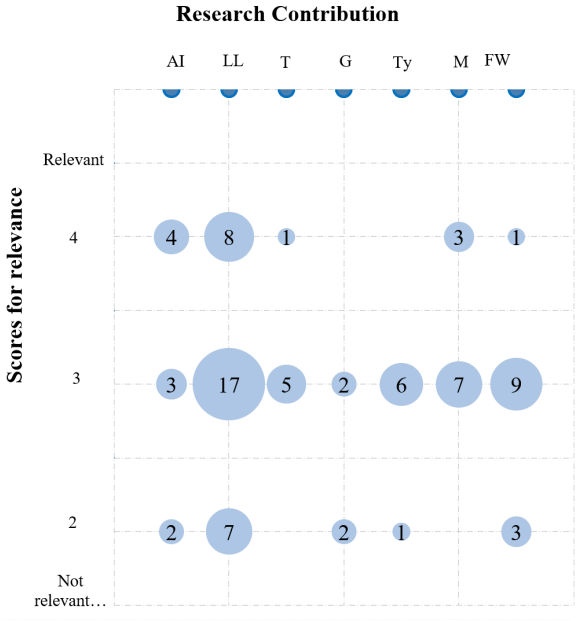


FIGURE 12. Systematic mapping of relevance on research contribution.

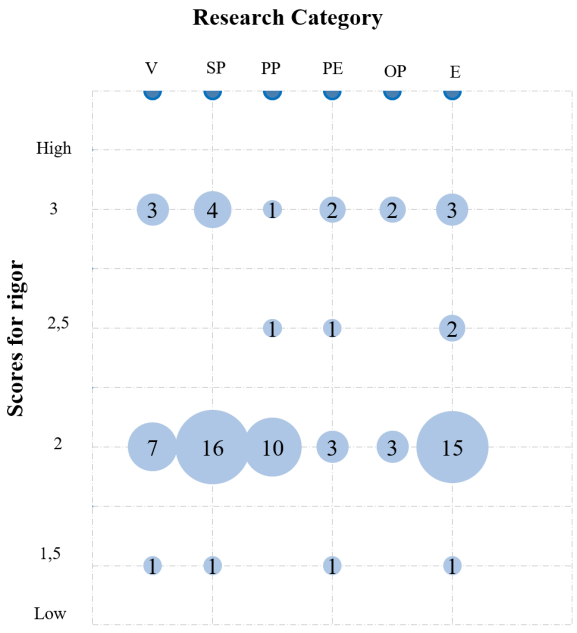


FIGURE 11. Systematic mapping of rigor on research category.

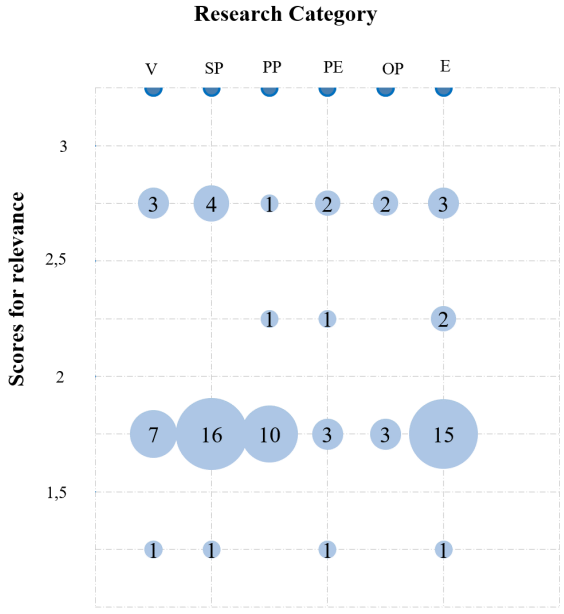


FIGURE 13. Systematic mapping of relevance on research category.

discussion covered in this section lead to appreciating the gaps and quality in research in the context of SSCs.

Generally, we visualise the findings from empirical literature of three decades in observations of each decade separately. The first ten, second ten and the rest of the ten years in which we observe 1, 19 and 79 percent of the studies respectively, with the last 5 years presenting with 62 percent. This is evidence that the years 2016 to 2021 had an increase in several studies compared to the prior 25 years. The years between 2016 and 2021 have presented with the highest percentages equally sharing 24 percent of the studies during the time. This is perhaps because new conferences on start-ups had been initiated. However, the trend is noticed from 2012 with the standardisation of the ISO/IEC 29110 for VSEs.

The analysis and discussion of this outcome answer the first, second, third research questions, the process tools adoption theoretical framework, discussion and analysis of rigor and relevance of the selected primary papers in the subsections A, B, C, D, E, respectively. Then subsection F presents the 3-point guideline for rigor and quality research,

and subsection G discusses how the study mitigated the threats to validity.

A. HOW HAS SOFTWARE PRACTICE IN SMALL SOFTWARE COMPANIES UTILISED THE SOFTWARE ENGINEERING KNOWLEDGE AREAS IN THE ISO/IEC TR 19759: 2015?

To answer this research question, we analysed and discussed the trend, examined how the knowledge areas have been translated to process tools to take care of the processes covered in the research. The articles considered in this study covered 9 knowledge areas for the 30 years; the first 20 years period of 1990 to 2011 are dominated by only two knowledge areas of software development and project management. However, in this same period, a few cases of three knowledge areas were observed in the literature in the last two years between 2009 and 2011. This indicates that researchers started paying attention to other knowledge areas in practice, specifically processes, quality, and methods and models.

In the subsequent ten years between 2012 and 2021, software development and project management are dominant, although a considerable rise in the other knowledge areas, including requirements engineering, software maintenance, software testing among software quality, processes, and models and methods.

Literature on software engineering practice in SSCs, [2], [40] suggest 9 and 11 knowledge areas, respectively, thus presenting a close similarity to the 9 knowledge areas identified by this study. Although both studies are not precisely on SSCs but rather a variant of it, we notice that development design, construction, and project management cover up to 64 percent of the articles in this study, with the former having 33 percent overall. This implies that the two areas have had the most research attention and would ordinarily mean the said areas have more attention in practice. However, it is imperative to note that the increase influences this results in several studies on the VSE because of mainly introducing the framework and standard ISO/IEC 29110, which covers these areas.

This development is an indication that this framework has made the VSE domain clear, organised, and simpler to understand. This was followed by an influx of research that also organised practice in the areas covered by the framework/standard. This is evidenced in literature with success factors with the VSE [69], [99], [114], [124] in which software development has become successful for general software and game development in different places, including Canada, Peru, Mexico and Finland.

Although the dominance of software development and project management as knowledge areas most covered in literature is very important for software development efficiency in SSCs, it is also significant that equal attention is given to the other knowledge areas, including requirements engineering, software testing, and software maintenance.

The trend of increased study in the other knowledge area, although observed to a minimal extent, probably explains the recent improvement noted in software development practice in SSCs. This same view is held in a number of recent studies [1], [4], [125]. In addition to this, many researchers also highlight the significance and call for increased attention on the other knowledge areas like requirements engineering, software quality, software testing, and software maintenance [126], [127]. Putting this together means the knowledge areas that seem prominent in empirical literature are effectively covered with subsequent process tools in practice, more so, evidence shows that the same process areas have sufficient attention in the literature and are better practised.

The entanglement of the SSCs in dilemmas of continued high failure and persistent production of inferior products gets significant explanations at this point. Similarly, the knowledge area like requirements engineering, software testing and maintenance are predicate to the process tools to utilise in the software development process. Unfortunately, they are not utilised and this will affect the basic understanding of applying them, knowing the challenges, what should be improved and hence the development of tools to support the processes. This, therefore, calls for efforts to factor investigating these knowledge areas deliberately and the practice in the corresponding practice to generate useful insights after all literature reveals that processes like requirements [18], [35], [128] and software testing [120], [129] are responsible for lots of challenges of software development process in practice of the SSCs. It would still be interesting to know if perhaps the practitioner's skill set is responsible for this maybe because of choice the practitioners tend to skew their skills and practice to other areas other than requirements, software testing and maintenance. This brings out interesting questions: for example, caring to know how much of requirements or testing in practice is enough because it seems like requirements are about the client being satisfied and the tests are about the errors not being found. Interestingly, ignorance of either a function by the client or limited test cases by both may affect requirements and testing.

B. HOW ARE COMPANIES REFERRED TO IN LITERATURE?

The main objective of this research question was to identify the terminologies used to refer to the companies in articles considered by this study. The identified terminologies used while referring to the companies included start-ups, SSCs, VSEs, and SMEs.

The smaller companies are known to have more challenges in producing quality software products due to inadequacies in the engineering practices [1], [8], [74], [76]. The sizes of these companies come with other inherent challenges affecting software development from the influencing factors like organization, business environment, governance, and technical factors [12], [32]. This, therefore, leads to the need in understanding context and its effect on the influencing factors. There seems to be an indication that researchers have not paid significant attention to size and context while developing tools and methods for software development [98]. This is evidenced by a lack of proper classification of small companies seen through author's usage of terminologies leading to con-

textual challenges in research and practice. The study mapped the usage of these terminologies in the empirical literature. The absence of a proper classification taxonomy has created an inconsistent usage of these terminologies while referring to the companies in the category of small companies. For example, the definition of software start-ups seems to overlap the description of VSEs and in some cases, the definition of SMEs [5] also seems to overlap the two. Similarly, some researchers also highlight the mix-up that would arise from using these terminologies [67]. Software engineering is a people-centred and intensive knowledge process with a specific methodology that requires process maturity [130], [131], least of which success may be challenging to attain.

The usage of the terminology in this study is fairly distributed to all the four terms commonly used to refer to the software companies. Although the VSEs are dominant with about 35 percent it is probably influenced by the ISO/IEC 29110 standard for VSEs introduced during the period of interest for this study. It is also clearly noticed that the term start-up is quite common, perhaps due to the increased attention on technology innovation that has seen the unprecedented promotion of start-ups in most economies as a source of employment creation owing to the flexibility they present [19]. This took the research community's attention with an introduction of a conference track dedicated to software start-ups, although unfortunately, the term has been used arbitrarily, that some companies are referred to as start-ups, yet they may not necessarily be start-ups. Starting a software company may not require much capital, which has attracted many entrepreneurs to venture into such setups. This is evidenced by the fact that most start-up owners are or employ less experienced people in software development. The start-ups, VSEs and SMEs, are ordinarily expected to present in different contexts, meaning the processes and methods of software development that apply to one may not apply. For example, using a general term such as SME creates unclear boundaries between small and medium companies. This also means that if a method is developed for SMEs, it could be applied to both small and medium companies, yet it cannot be the case in most cases. Overall, the arbitral usage of these terminologies may cause the arbitral application of process tools frameworks methods in software development, yet they may not apply in context. This, therefore, would require a proper classification taxonomy for small companies since the category of small companies involves the VSE, start-ups and those that remain unnamed.

C. WHAT IS THE PRACTICE OF UTILISATION OF SOFTWARE DEVELOPMENT FRAMEWORKS IN SMALL SOFTWARE COMPANIES?

Most of the studies do not cite utilisation of software development frameworks/standards in the companies researched, although 7 frameworks are cited in the primary studies, representing less than 50 percent. This perhaps confirms the inability of the smaller companies to adopt the existing frameworks, as cited by Alexandre *et al.* [11] and Anacleto *et al.* [12]. This finding means that more than 50 percent of the studies are not connected to any specific framework or standard. The ISO/IEC 29110 is very prominent as a framework for VSEs, and the other 6 are decimally covered in the primary studies, as illustrated in Figure 9.

The prominence of ISO/IEC 29110 seems to closely relate to this study's findings that the two knowledge areas of software development and project management are specifically tagged to this framework. This perhaps explains the finding that is followed by numerous studies in software engineering and project management areas. The other contexts, that do not fall under the realm of VSEs and are also small companies and may not necessarily be medium or larger software companies, remain unattended to as far as tools and frameworks are concerned. The effective utilisation of tools, methods, processes, and frameworks for small companies is largely dependent on the understanding the context in which the companies operate because context differs; therefore, the tools, methods, processes, and frameworks will differ. This now gives a proper explanation on why SSCs continue to register failure and produce low-quality products. Additionally, although the minimal usage of frameworks and standards like CMMI is thought to be primarily because of affordability to small enterprises [110], an argument also fronted by Singh and Gill [105], the issues seem not to be just about costs, but rather on the difficulty to fit the tools with the context of the companies. However, an opportunity of exploring the components that can be useful in improving quality still exists just like it is seen for standards like ISO IEC 15288 that defines the software lifecycle [132], whose models are adopted in ISO IEC 12207 to guide development and maintenance [37]. For small enterprises to improve on quality within the current environment, the challenges associated to software development practice in SSCs that are unique to them must be given special attention. Laporte and O'Connor [133], stress the fact that all organisations cannot be similar, further cautioning that if the developers of the software process models do not take into account the different operational contexts, the influence of SPI may be far from being achieved and it therefore remains theoretical than practical.

Understanding the company's context to propose appropriate tools and frameworks is, significant in improving software development practice by SSCs. Clarke *et al.* [134], reinforces this recommendation, adding that variation in software development contexts needs to be considered. Additionally, an adoption framework supported by a classification taxonomy for SSCs will organise and highlight the need for attention in the categories, which currently lack specific frameworks and tools to support software practices in SSCs.

D. THEORETICAL FRAMEWORK

1) OVERVIEW OF THEORETICAL FRAMEWORK

The study consolidates and integrates the fragmented findings from the empirical literature on software processes and highlights key points related to the utilization of software

processes tools in SSCs and the ability to attain quality products in the context in which software is developed. The highlights include: (i) insufficient use of the knowledge areas in software practice by SSCs - this means the processes of software development to ensure effective production of software are not being used; (ii) the ill-defined context within which SSCs operate, with an implication on the process tools – SSCs are expected to use common tools, breeding a complexity; and (iii) evidence of minimal usage of process tools which raises new concern on how quality software can be produced in an environment where the process tools are not usable.

2) CONCEPTS OF THE FRAMEWORK

In pursuance of the issues raised from the empirical literature, it is necessary for process theory to transform software engineering body of knowledge especially paying attention to SSCs. Additionally, research in software engineering has been predominately prescriptive and method-focused [39], producing thousands of software development methods that remain underutilized. In order to solve the challenges with software practice, practitioners and researchers should contribute to the body of knowledge [135], given that SSCs dominate the industry and are responsible for over 80% of software produced in the market.

Software engineering literature highlights the difficulty in adopting the process tools [11], [38], yet poor quality software and high project failure continue to raise concern. Whereas companies need the process tools to streamline the development processes, the available process tools are minimally utilized. Consequently, this paper proposes a software process tools adaptability framework to explain how process tools for software development are not utilisable by the SSCs.

This theoretical framework is based on a theory that posits that:

Software process tools can only be useful to small software companies if the context in which these companies operates is considered.

This has two implications: first, it implies that for the existing process tools to be usable by the SSCs, an adaptability mechanism must be put in place to streamline the process tool with the context of the company, which requires an in-depth understanding of the characteristics of the companies and a classification taxonomy. Secondly, the new process tools under construction must take care of the operational context of the SSCs, paying keen attention to the fact that the SSCs differ in character. This implies that an assessment for the character to ensure adaptability must be considered.

Through the study the researchers extracted four concepts to build the proposed process tools adoption framework. The concepts are adoption mechanism, process tools, influencing factors and software quality.

Table 8 shows the concepts, their respective descriptions, and the references of literature. The theoretical framework is developed from the concepts of development process tools, influencing factors and quality of software as constructs generated from the findings of the study. From the literature, we also identify assumptions used in software development practice in SSCs:

- Process tools influence the production of quality software and this means that the process tools have a relationship to the production of quality software.
- The influencing factors of software development (organization factors, the business environment, the governance factors, and the technical factors) within an organization are mediators to the relationship between the process tools and the production of quality software.

However, the process tools have a weak relationship to the production of quality software, and the weakness is explained by the inability of the SSCs to adopt the process tools hence failure to produce quality products. The weak relationship between the process tools and the production of quality software products during the development process is highlighted as the gap in research and practice, as evidenced by the continued poor quality of software products [46], [98], [76], [23]. The researchers propose the adoption mechanism to strengthen the relationship between the development process tools and quality software products. The role of the adoption mechanism is to ensure that the process tools are adaptable to the contexts of the companies for the relationship to be useful. This relationship needs to be mitigated with constructs that are moderators in the relationship.

TABLE 8. Concepts for the software process tools adoption framework.

Adoption mechanism	Understanding the context in which software development is practised has a relationship with utilising tools, process frameworks, methods, and standards. Evidence shows that the adoptability of the tools and frameworks is limited, which poses the development processes with contextual complexity.	[12], [38], [139]
Process tools	Process tools are found to be minimally used yet they are responsible for the foundation of guiding the complete software engineering process. This influences the quality of software produced through the process that are not guided using process tools	[12], [48]
Influencing factors	The knowledge areas define the processes, knowledge and skills that a software engineer should know to design quality software. Requirements, Testing, Maintenance are influenced by organization, business environment, governance, and the technical factors. These create limitations of the knowledge areas affects practice and process tool ultimately affecting the quality of software.	[1], [140], [141]
Quality software products	Process tools have a relationship with quality, but the tools must be adaptable to attain quality. How well a software product complies with or conforms to a given design, based on requirements or specifications and to meet customer satisfaction.	[1], [101], [142]–[144]

The theoretical framework explains the difficulty to attain quality products by SSCs, although software process tools seem important and are known or expected to be useful in transforming the processes and practices. The framework also predicts the likely increase in utility of process tools when an adoption mechanism has been applied making the process tool adoptable to the SSCs based on a classification taxonomy derived through the characteristics.

Process tools like methods, frameworks and standards are designed to support the different activities presumed to be vital in delivering quality software, the challenge, however is that the different context in which these companies operate make the practicality difficult in applying the tool to the extent that most companies end up not using the tools at all. This theoretical framework also leads to the predictability of adoptable software process tools, while developing the tools, effective consideration of the specific context of the companies to utilise them should be considered. This can be achieved using a classification taxonomy from which proper categorisation of the SSCs is done and the utilisation of the process tools is tailored based on the classification. This theoretical framework also becomes an eye-opener to researchers who will develop other tools specifically challenging the researchers to consider adaptability challenges arising from the difference in context in which companies operate and the varied characteristics of the companies which should be put into account. This theoretical framework is crucial to preserve and interconnect empirical knowledge and protect the software processes against fragmented empiricism and overemphasis on prescriptive knowledge that builds many tools.

Therefore, the researchers posit that for software companies to produce quality software while using process tools, the company's limitations, and strengths regarding the influencing factors in affecting the quality of software products under development should be considered. Secondly, software companies are not the same in character and the effectiveness of the tools used in the processes are dependent on how they fit into the context of the company's operations.

The relationships between the constructs and the assumptions are evaluated for weaknesses and it is from this that the gaps in literature are identified to form the constructs that need to be added to fill up the gap in theory.

The software process tool adoption framework is important in explaining the relationship between software process tools and software product quality. It also explains the inability of the SSCs to utilise the software process tools. This means that constructors of process tools will use the proposed framework to consider adaptability and operational context of the companies to build utilisable tools. The framework also creates an opportunity to predict the extent of quality attainable by a process tool built to specific context. The framework has an impact on practice by simplifying adaptability of the software process tools that has not been easy to use by the SSCs. Nonetheless, this benefit is attainable after factoring adoptability in the process tools either while developing the process tool or subjecting the existing process tool through an adaptability mechanism as suggested by the framework.

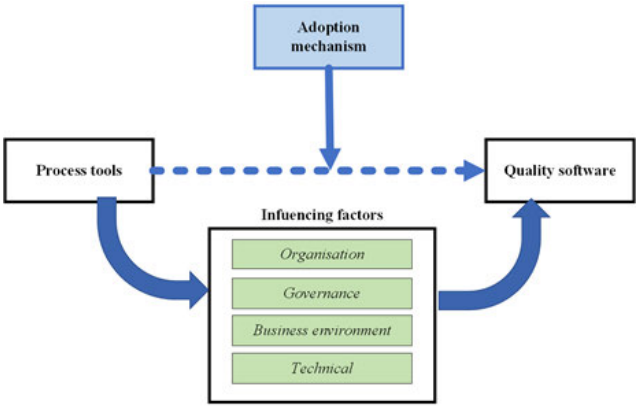


FIGURE 14. Process tools adoption framework.

This framework was constructed to integrate the fragmented findings of the empirical literature of software practice in SSCs although our findings show that lots of other knowledge areas are not sufficiently covered, presenting a possibility of bias. Other researchers share similar sentiments suggesting that academic literature has a bias towards formal methodologies [135]. This aspect was considered and mitigated, although the researchers recommend increasing empirical studies on knowledge areas to create a significant impact on improving software development process practice in SSCs.

Knowledge areas like requirements, software quality, software testing and software maintenance need to be factored in research to positively influence developing software products in SSCs an implication shared by other researchers in [4], [40]. This will also streamline successful and sustainable development practices, especially for the SSCs that are pivoting to medium and larger companies.

3) OTHER THEORIES

In comparison to other process theories, this theory provides *predictive* and *prescriptive* support for software engineering and also guides the choice of tools during software development getting away from the trial and error approach in SSCs only to be frustrated by failure to adopt to the tool because of contextual complexity [38].

Other researchers have proposed other theories; the Sensemaking–Coevolution–Implementation Theory [135] and the Function Behavior-Structure Framework [39], both adopted as software process theories. The former explains how cohesive software development teams in organizations create complex software systems. This theory is useful as a process theory; however, it falls short of taking into consideration the adaptation of the tools used in software practice. Similarly, the latter fronts a traditional view of the process of software development. This theory assumes that during problem framing, the artifact's structure is driven by its requirements, which are driven by goals, and that

TABLE 9. Conceptual evaluation of the software process tools adaptability framework.

	Evaluation criteria	*indicative parameters*
1	Feasibility Is there a fit between the theory and the problem?	The theory helps to mitigate inadequacies in software engineering by guiding the processes
2	Excitement Does theory promote new insight?	The theory is promoting the concept of context and adaptability of the process tools that are not easily utilisable by SSCs.
3	Context Is the theory appropriate in the context of the current problem?	The theory fits well in mitigating the challenges of software practice in SSCs.
4	Cost Is the theory affordable in terms of time and effort?	The theory is affordable in terms of effort and time to understand.
5	User friendly is the language of theory understandable and enlightening?	The theory uses simple and familiar words normally used by software practitioners.
6	Fruitful Is theory useful?	The theory is useful for developing adaptability of process tools and this can also guide the building of process tools to take adaptability into account.

the designers primarily evaluate their designs by predicting behaviour from design models. This theory also does not look at the context of the software company and the adaptability of process tools to create quality software.

E. RIGOR AND RELEVANCE

1) EVALUATION OF THE THEORETICAL FRAMEWORK

The assessment of methods and quality of research is vital to this study in demonstrating the extent of rigor and relevance of the selected articles in this study. Although the high relevance observed demonstrates how much evidence produced in the study has been adduced to working with the software companies themselves, this demonstrates how the study can influence the industry practice. On the other hand, rigor is reported as inadequate, and this is an area that requires substantial attention. This is similar to the findings of similar mapping studies of Klotins *et al.* [40] and Paternoster *et al.* [7]. This means that the description of context and study design in most of the studies has remained a challenge and, therefore a threat to generalizability and knowledge transfer.

Although the studies are conducted in real industry settings, this breeds a sense of realism and therefore making the study outcome transferable, it simply means that most of the studies included in the SMS are relevant. Unfortunately, the extent of rigor is what remained wanting. This, however, poses a threat to transferability, particularly in terms of context, study design, assessment of threats to validity of specific studies, and the extent to which these aspects of rigor are addressed in the selected studies expected. This is consistent with other findings from other studies [40] and [7].

We therefore present a set of guidelines to ensure that researchers have a reference point to guide their work to quality.

F. 3-POINT GUIDELINES FOR RIGOR AND QUALITY RESEARCH

As a remedy to the challenges unveiled, we develop a 3-point guideline as a recommendation to ensure that the research is both of quality and of high rigor, so that it is generalisable and easily transferable to industry. The guidelines cover aspects of methodology, the results, and the conclusion.

1) THE GUIDELINE ON METHODOLOGY

The study method and its processes should be described to detail the work's repeatability. The detailed description should also enable confirmability [141], A well-defined context and design of the study is very significant in supporting and promotion of research relevant to industry. It creates an avenue for applying the success reports of technologies and methods recorded in research in real-life industry software projects as highlighted by Ivarsson and Gorschek [49].

2) THE GUIDELINE ON PRESENTATION OF RESULTS

Results should be presented in a natural form while ensuring honesty and transparency [142]. It is important to avoid heavy undertones that are unnecessary Authors are encouraged to use robust descriptive language to provide sufficient contextual information that enables the reader to determine *credibility, transferability, dependability* and *confirmability* [143]. Communication to the reader should be clear and that the results must be based or reflect the information gathered from the participants. Efforts should be put in place to ensure that the results are not biased interpretations of the researcher [144]. Additionally, the geographical location of the study, characteristics of participants that have taken part in the study, and the specific time of data collection and analysis should be thoroughly described for the readers to appreciate the context of the study [145].

3) THE GUIDELINE ON THE CONCLUSION

The conclusion of the study should be arrived at based on the results and must enhance the study's contribution [146]. This should be reflected in the discussion and the paper's conclusion while answering questions like so what? Why do the results matter? What next? This needs to pay significant attention to both the conceptual and practical perspectives of the study. An avaricious narrative should be used while explaining results to enhance understanding of the research questions while relating the findings to each other [147].

The 3-point guideline set out statements on processes of writing the methodology, reporting the results, and writing the conclusions in a manuscript intentioned to determine a course of action while conducting research software engineering research. This guideline aims to streamline processes of writing to ensure quality of research.

The guideline is important because it has been derived from evidence in empirical literature that highlights the areas in research writing which require attention [40], [2], [148]; it takes care of the parts which are most wanting and has

a significant effect on the quality of research in software engineering. The guidelines cover only what has been identified as the areas that remain unclear in software engineering literature yet are important in reproducibility, generalisability and transferability.

Akin to this, researchers are advised to make an effort to collaborate with industry to ensure transfer and widespread use of research results in industry. This will also ensure that research results are evidence-based.

G. THREATS TO VALIDITY

While conducting systematic mapping studies, threats to validity must be considered. The descriptive, theoretical, generalizability, interpretive and repeatability present different levels of the threats to validity and mitigation measures have been put in place as discussed herein.

1) THEORETICAL VALIDITY

This is defined as the ability of a study to report what is intended for the study. Theoretical bias arises out of the challenges in the processes of identification and selection of the primary studies. The search strategy was designed to be as inclusive as possible in this study, although we ended up with 77 primary studies, out of 16538 search results from the 4 databases. However, this seems like a limitation of the search string because terms like ''software start-up'' and ''SMEs'' return mostly irrelevant results. To mitigate this threat, we had an opportunity to include qualifiers like ''software start-up'' and ''software SME''; however, many papers do not necessarily qualify the start-ups and the SMEs. To this effect, we enforced the inclusion and exclusion criteria in Table 2 to ensure that the empirical literature was precisely what we were looking for and had originated from a software engineering database.

Before arriving at the selected 77 primary studies, it was noticeable that using the term SME in the search phrase would pose invalidity to the selection, simply because many studies inconsistently used this term. Specific attention was given to each of the papers that reported on SME to strictly ensure that the paper had covered a considerable majority or all companies we considered small companies, particularly with 50 persons or less. Similarly, for start-ups, there are instances in literature where companies are classified as start-ups based on lack of resources and use immature processes, as cautioned by Paternoster *et al.* [7].

2) DESCRIPTIVE VALIDITY

The ability of the research study to describe the gathered information accurately ensures descriptive validity. Clearly defining and justifying the objectives in the study protocol paved way for a precise understanding of the data to be gathered in the study. Data extraction was carefully done using google forms to accurately record the information that we used to populate generated schema that led to answering the research questions. Additionally, experienced reviewers had to go through and approve the protocol, the final findings, and the reporting to ensure descriptive validity.

3) GENERALIZABILITY

The result obtained from this systematic mapping study is generalisable to practitioners and researchers in different knowledge areas of software engineering for SSCs. This is because the study ensured the coverage of a broad research area for a time interval of 30 years, which is long enough to capture most literature as far as software engineering in SSCs have been reported.

The researchers also considered the choice of databases to extract the papers as significant to ensure generalizability. The researchers chose four(4) databases: ISI web of science, ACM digital library, IEEE Xplore, and Scopus, believed to be among the most popular to the software engineering audience as guided by Kitchenham and Chanters [46] and are also commonly used by other researches in similar systematic mapping studies in [2], [7], [47], [48]. The coverage of a broad area in software engineering practice of SSCs over a significant period and the consideration of the most used databases makes the result of this study quite generalisable.

4) INTERPRETIVE VALIDITY

The ability of researchers to interpret the gathered data accurately without using their own perspective is referred to as interpretive validity. To ensure this, the conclusions in this research were based on the data we gathered and the diverse perspectives about the data, resulting in similar interpretations of the data collected. The joint involvement of all researchers with expertise in software engineering and empirical research (especially mapping studies and reviews) contributed to ensuring interpretive validity.

5) REPEATABILITY

This is the ability of a study to be undertaken by another researcher to reproduce closely related results. The mapping procedure is specifically documented and reported systematically to ensure that the study can be reproduced with similar results based on studied research papers reported in this paper. Furthermore, this will ensure that the other researchers can reproduce this mapping study under similar conditions: search string, and dates of the study period. However, due to issues with the abstracts of few studies and ambiguous use of terms, the repeatability with the same classification may vary marginally.

VI. CONCLUSION AND FUTURE WORK

A. CONCLUSION

This work is motivated by consolidating and integrating the fragmented findings in empirical literature of software process on SSCs. The paper makes two distinct contributions in the form of constructs that are induced from the existing empirical work, the main construct is based on the relationship between the three areas studied in the research questions

related to software development practice for SSCs, and the other construct is attributed to the challenges of quality and rigor of research in software engineering.

Overall, as SSCs undertake ambitious projects of quality software, practice in the SSCs need to be given the required attention [1] , [11]. A situation where process tools are available to have things done and not have the things done right cannot transform software practice. Calls for the transformation of software processes have led to a new dilemma of researchers developing lots of prospective process tools that are not utilisable by the SSCs [12], [39]. Due to the complexity that arises out of the efforts to solve the challenges of lack of utilization of the process tools by the SSCs, we find the justification of a new theoretical framework highlighting the relationship between the process tools and quality software products moderated by the adaptation mechanism, which is very important for utilization of the process tools especially to enhance practice. For the first time this theory helps in the explanation for the reasons for SSCs failure to utilise the numerous process tools which has been a continued concern both in practice and research as observed in [1], [11], [12], [98], [149].

The framework is evaluated conceptually against 6 evaluation criteria and compared with two process frameworks; the sensemaking–coevolution–implementation theory [135] and the function behavior-structure framework [39]. Although both are process theories, none tackles the challenges of adoptability of the process tools. Unlike our novel adoption theoretical framework, these are more of design process theory and do not look at the entire development process but rather the design process.

There is a need to explore a mechanism of proper classification of the SSCs, especially by undertaking more empirical studies on the characteristics of the SSCs. This should build meaningful consensus on the understanding of the different context of SSCs. Similarly, researchers need to pay attention to tools like process frameworks, standards, and methods especially those that tend towards flexibility and are comprehensive for the different contexts of SSCs.

Generally, the extent to which research on SSCs is transforming practice is still lacking. However, this transformation is possible through research, especially when the specific effort is put on the quality of the research to simplify and improve the research results' ability to be generalisable and transferrable to practitioners in industry.

B. FUTURE WORK

The future work will use the overview attained through this study to have a detailed review of software practice in SSCs in a multi-vocal literature review, in which grey literature will make a meaningful contribution to understanding software practice and particularly the gap that exists in practice as seen by industry and academia. This is intentioned to capture grey literature that is not recorded in academic literature.

We are currently investigating the experiences of software practitioners in using the different process tools by SSCs collaborating with software companies in 6 countries to appreciate the difference in contexts and the challenges of adoption of the tools due to context. The results from these ongoing studies combined with software process adoption theoretical framework to design and construct classification taxonomy and an adoption framework that the current process tools can use to ensure that the tools are used for software development by the SSCs.

REFERENCES

[1] M. Tuape and Y. Ayalew, "Factors affecting development process in small software companies," in *Proc. IEEE/ACM Symp. Softw. Eng. Afr. (SEiA)*, May 2019, pp. 16–23.
[2] V. Berg, J. Birkeland, A. Nguyen-Duc, I. O. Pappas, and L. Jaccheri, "Software startup engineering: A systematic mapping study," *J. Syst. Softw.*, vol. 144, pp. 255–274, Oct. 2018, doi: 10.1016/j.jss.2018.06.043.
[3] A. Majchrowski, C. Ponsard, S. Saadaoui, J. Flamand, and J.-C. Deprez, "Software development practices in small entities: An ISO29110-based survey," *J. Softw., Evol. Process*, vol. 28, no. 11, pp. 990–999, Nov. 2016.
[4] N. Tripathi, E. Annanperä, M. Oivo, and K. Liukkunen, "Exploring processes in small software companies: A systematic review," in *Software Process Improvement and Capability Determination*, vol. 609. Cham, Switzerland: Springer, Jun. 2016, pp. 150–165, doi: 10.1007/978-3-319-38980-6_12.
[5] E. Commission. (2020). *User Guide to the SME Definition*. Publications Office of the European Union. Accessed: May 30, 2021. [Online]. Available: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2003:124:0036:0041:en:PDF
[6] N. G. Lester, F. G. Wilkie, D. McFall, and M. P. Ware, "Investigating the role of CMMI with expanding company size for small-to medium-sized enterprises," *J. Softw. Maintenance Evol., Res. Pract.*, vol. 22, no. 1, pp. 17–31, Jan. 2010.
[7] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software development in startup companies: A systematic mapping study," *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1200–1218, Oct. 2014, doi: 10.1016/j.infsof.2014.04.014.
[8] R. V. O'Connor, "Developing software and systems engineering standards," in *Proc. 16th Int. Conf. Comput. Syst. Technol. (CompSysTech)*, 2015, pp. 13–21, doi: 10.1145/2812428.2812430.
[9] E. Mnkandla, "About software engineering frameworks and methodologies," in *Proc. AFRICON*, Sep. 2009, pp. 1–5.
[10] A. M. AL-Ashmori, B. B. Rad, and S. Ibrahim, "Software process improvement frameworks as alternative of CMMI for SMEs: A literature review," *J. Softw. Eng.*, vol. 11, no. 2, pp. 123–133, Apr. 2017.
[11] S. Alexandre, A. Renault, and N. Habra, "OWPL: A gradual approach for software process improvement in SMEs," in *Proc. 32nd EUROMICRO Conf. Softw. Eng. Adv. Appl. (EUROMICRO)*, 2006, pp. 328–335.
[12] A. Anacleto, C. G. von Wangenheim, C. F. Salviano, and R. Savi, "Experiences gained from applying ISO/IEC 15504 to small software companies in Brazil," in *Proc. 4th Int. SPICE Conf. Process Assessment Improvement*, Lisbon, Portugal, 2004, pp. 33–37.
[13] V. Claudia, M. Mirna, and M. Jezreel, "Characterization of software processes improvement needs in SMEs," in *Proc. Int. Conf. Mechatronics, Electron. Automot. Eng.*, Nov. 2013, pp. 223–228.
[14] M. Choras, T. Springer, R. Kozik, L. Lopez, S. Martinez-Fernandez, P. Ram, P. Rodriguez, and X. Franch, "Measuring and improving agile processes in a small-size software development company," *IEEE Access*, vol. 8, pp. 78452–78466, 2020, doi: 10.1109/ACCESS.2020.2990117.
[15] B. Komal, U. I. Janjua, F. Anwar, T. M. Madni, M. F. Cheema, M. N. Malik, and A. R. Shahid, "The impact of scope creep on project success: An empirical investigation," *IEEE Access*, vol. 8, pp. 125755–125775, 2020, doi: 10.1109/ACCESS.2020.3007098.
[16] T. Yaghoobi, "Prioritizing key success factors of software projects using fuzzy AHP," *J. Softw., Evol. Process*, vol. 30, no. 1, p. e1891, Jan. 2018.
[17] (1999). *The Software Engineering Process: Definition and Scope*. Accessed: Feb. 14, 2020. [Online]. Available: https://www.mendeley.com/research-papers/?query=10.1145/75110.75122
[18] C. Gralha, D. Damian, A. Wasserman, M. Goulão, and J. Araújo, "The evolution of requirements practices in software startups," in *Proc. 40th Int. Conf. Softw. Eng. (ICSE)*, May 2018, pp. 823–833, doi: 10.1145/3180155.3180158.

[19] N. Tripathi, M. Oivo, K. Liukkunen, and J. Markkula, "Startup ecosystem effect on minimum viable product development in software startups," *Inf. Softw. Technol.*, vol. 114, pp. 77–91, Oct. 2019, doi: 10.1016/j.infsof.2019.06.008.

[20] R. Anwar, M. Rehman, K. S. Wang, M. A. Hashmani, and A. Shamim, "Investigation of knowledge sharing behavior in global software development organizations using social cognitive theory," *IEEE Access*, vol. 7, pp. 71286–71298, 2019, doi: 10.1109/ACCESS.2019.2912657.

[21] A. I. Wasserman, "Low ceremony processes for short lifecycle projects," in *Managing Software Process Evolution*. Cham, Switzerland: Springer, 2016, pp. 1–13, doi: 10.1007/978-3-319-31545-4_1.

[22] C. G. V. Wangenheim, S. Weber, J. C. R. Hauck, and G. Trentin, "Experiences on establishing software processes in small companies," *Inf. Softw. Technol.*, vol. 48, no. 9, pp. 890–900, Sep. 2006.

[23] C. G. von Wangenheim, A. Anacleto, and C. F. Salviano, "Helping small companies assess software processes," *IEEE Softw.*, vol. 23, no. 1, pp. 91–98, Jan. 2006, doi: 10.1109/MS.2006.13.

[24] M. E. Fayad, M. Laitinen, and R. P. Ward, "Thinking objectively: Software engineering in the small," *Commun. ACM*, vol. 43, no. 3, pp. 115–118, Mar. 2000, doi: 10.1145/330534.330555.

[25] Y. Li, K.-C. Chang, H.-G. Chen, and J. J. Jiang, "Software development team flexibility antecedents," *J. Syst. Softw.*, vol. 83, no. 10, pp. 1726–1734, Oct. 2010, doi: 10.1016/j.jss.2010.04.077.

[26] I. F. da Silva, P. A. da Mota Silveira Neto, P. O'Leary, E. S. de Almeida, and S. R. D. L. Meira, "Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study," *J. Syst. Softw.*, vol. 88, pp. 189–206, Feb. 2014, doi: 10.1016/j.jss.2013.10.040.

[27] X. Larrucea, R. V. O'Connor, R. Colomo-Palacios, and C. Y. Laporte, "Software process improvement in very small organizations," *IEEE Softw.*, vol. 33, no. 2, pp. 85–89, Mar. 2016, doi: 10.1109/MS.2016.42.

[28] N. Tripathi, E. Klotins, R. Prikladnicki, M. Oivo, L. B. Pompermaier, A. S. Kudakacheril, M. Unterkalmsteiner, K. Liukkunen, and T. Gorschek, "An anatomy of requirements engineering in software startups using multi-vocal literature and case survey," *J. Syst. Softw.*, vol. 146, pp. 130–151, Dec. 2018, doi: 10.1016/j.jss.2018.08.059.

[29] E. Klotins, M. Unterkalmsteiner, P. Chatzipetrou, T. Gorschek, R. Prikladnicki, N. Tripathi, and L. B. Pompermaier, "Exploration of technical debt in start-ups," in *Proc. 40th Int. Conf. Softw. Eng., Softw. Eng. Pract.*, May 2018, pp. 75–84, doi: 10.1145/3183519.3183539.

[30] R. V. O'Connor, "Evaluating management sentiment towards ISO/IEC 29110 in very small software development companies," in *Software Process Improvement and Capability Determination*, vol. 290. Berlin, Germany: Springer, May 2012, pp. 277–281, doi: 10.1007/978-3-642-30439-2_31.

[31] M. Muñoz, A. Peña, J. Mejia, G. P. Gasca-Hurtado, M. C. Gómez-Alvarez, and C. Laporte, "A comparative analysis of the implementation of the software basic profile of ISO/IEC 29110 in thirteen teams that used predictive versus adaptive life cycles," in *Proc. Eur. Conf. Softw. Process Improvement*, 2019, pp. 179–191.

[32] C. Y. Laporte and J. M. Miranda, "Delivering software- and systems-engineering standards for small teams," *Computer*, vol. 53, no. 8, pp. 79–83, Aug. 2020, doi: 10.1109/MC.2020.2993331.

[33] A. Aldaeej, "Towards effective technical debt decision making in software startups," *SIGSOFT Softw. Eng. Notes*, vol. 44, no. 3, p. 22, 2019, doi: 10.1145/3356773.3356793.

[34] J. Melegati, A. Goldman, F. Kon, and X. Wang, "A model of requirements engineering in software startups," *Inf. Softw. Technol.*, vol. 109, pp. 92–107, May 2019.

[35] U. Rafiq, S. S. Bajwa, X. Wang, and I. Lunesu, "Requirements elicitation techniques applied in software startups," in *Proc. 43rd Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2017, pp. 141–144.

[36] A. Mishra and D. Mishra, "Software project management tools: A brief comparative view," *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 3, pp. 1–4, May 2013, doi: 10.1145/2464526.2464537.

[37] G. Marks, R. O'Connor, M. Yilmaz, and P. Clarke, "An ISO/IEC 12207 perspective on software development process adaptation," *Softw. Qual. Prof.*, vol. 20, no. 2, pp. 48–58, 2018.

[38] R. V. O'Connor and G. Coleman, "Ignoring 'best practice': Why Irish software SMEs are rejecting CMMI and ISO 9000," *Australas. J. Inf. Syst.*, vol. 16, no. 1, pp. 1–24, Nov. 2009.

[39] D. Truex, R. Baskerville, and J. Travis, "Amethodical systems development: The deferred meaning of systems development methods," *Accounting, Manage. Inf. Technol.*, vol. 10, no. 1, pp. 53–79, Jan. 2000.

[40] E. Klotins, M. Unterkalmsteiner, and T. Gorschek, "Software engineering knowledge areas in startup companies: A mapping study," in *Software Business*, vol. 210. Cham, Switzerland: Springer, Jun. 2015, pp. 245–257, doi: 10.1007/978-3-319-19593-3_22.

[41] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. 12th Int. Conf. Eval. and Assessment Softw. Eng. (EASE)*, Jun. 2008, pp. 1–10.

[42] J. Bailey, D. Budgen, M. Turner, B. Kitchenham, P. Brereton, and S. Linkman, "Evidence relating to object-oriented software design: A survey," in *Proc. 1st Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Sep. 2007, pp. 482–484.

[43] D. H. Lee and M. H. Kim, "Accommodating subjective vagueness through a fuzzy extension to the relational data model," *Inf. Syst.*, vol. 18, no. 6, pp. 363–374, Sep. 1993.

[44] S. Ali, H. Li, S. U. Khan, Y. Zhao, and L. Li, "Fuzzy multi attribute assessment model for software outsourcing partnership formation," *IEEE Access*, vol. 6, pp. 55431–55461, 2018, doi: 10.1109/ACCESS.2018.2871710.

[45] S. Ali, N. Ullah, M. F. Abrar, Z. Yang, and J. Huang, "Fuzzy multicriteria decision-making approach for measuring the possibility of cloud adoption for software testing," *Sci. Program.*, vol. 2020, Apr. 2020, Art. no. 6597316, doi: 10.1155/2020/6597316.

[46] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proc. 28th Int. Conf. Softw. Eng.*, May 2007, pp. 1051–1052.

[47] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. S. de Almeida, and S. R. de Lemos Meira, "A systematic mapping study of software product lines testing," *Inf. Softw. Technol.*, vol. 53, no. 5, pp. 407–423, May 2011.

[48] V. Gupta, J. M. Fernandez-Crehuet, T. Hanne, and R. Telesko, "Requirements engineering in software startups: A systematic mapping study," *Appl. Sci.*, vol. 10, no. 17, p. 6125, Sep. 2020.

[49] M. Ivarsson and T. Gorschek, "A method for evaluating rigor and industrial relevance of technology evaluations," *Empirical Softw. Eng.*, vol. 16, no. 3, pp. 365–395, Jun. 2011.

[50] M. Shaw, "Writing good software engineering research papers," in *Proc. 25th Int. Conf. Softw. Eng.*, 2003, pp. 726–736.

[51] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion," *Requir. Eng.*, vol. 11, no. 1, pp. 102–107, Mar. 2006.

[52] K. C. Dangle, P. Larsen, M. Shaw, and M. V. Zelkowitz, "Software process improvement in small organizations: A case study," *IEEE Softw.*, vol. 22, no. 6, pp. 68–75, Oct. 2005, doi: 10.1109/MS.2005.162.

[53] M. A. Almomani, S. Basri, A. K. B. Mahmood, and Y. M. Baashar, "An empirical analysis of software practices in Malaysian small and medium enterprises," in *Proc. 3rd Int. Conf. Comput. Inf. Sci. (ICCOINS)*, Aug. 2016, pp. 442–447.

[54] M. Ateeq, "The adoption of software process improvement in Saudi Arabian small and medium size software organizations: An exploratory study," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 3, pp. 195–201, 2018.

[55] F. Caffery, P. Taylor, and G. Coleman, "Adept: A unified assessment method for small software companies," *IEEE Softw.*, vol. 24, no. 1, pp. 24–31, Jan. 2007, doi: 10.1109/MS.2007.3.

[56] F. X. Bru, G. Frappin, L. Legrand, E. Merrer, S. Piteau, G. Salou, P. Saliou, and V. Ribaud, "Building an observatory of course-of-action in software engineering: Towards a link between ISO/IEC software engineering standards and a reflective practice," *Commun. Comput. Inf. Sci.*, vol. 42, pp. 185–200, Sep. 2009, doi: 10.1007/978-3-642-04133-4_16.

[57] T. Nonoyama, L. Wen, and T. Rout, "Current challenges and proposed software improvement process for VSEs in developing countries," in *Proc. Int. Conf. Softw. Process Improvement Capability Determination*, 2016, pp. 437–444.

[58] K. Suteeca and S. Ramingwong, "A framework to apply ISO/IEC29110 on SCRUM," in *Proc. Int. Comput. Sci. Eng. Conf. (ICSEC)*, Dec. 2016, pp. 1–5.

[59] M.-L. Sanchez-Gordon, R. V. O'Connor, and R. Colomo-Palacios, "Evaluating VSEs viewpoint and sentiment towards the ISO/IEC 29110 standard: A two country grounded theory study," in *Proc. Int. Conf. Softw. Process Improvement Capability Determination*, 2015, pp. 114–127.

[60] L. Rivas, M. Pérez, L. E. Mendoza, and A. Grimán, "Towards a selection model for software engineering tools in small and medium enterprises (SMEs)," in *Proc. 3rd Int. Conf. Softw. Eng. Adv.*, Sliema, Malta, Oct. 2008, pp. 264–269, doi: 10.1109/ICSEA.2008.51.

[61] V. Ribaud, P. Saliou, R. O'Connor, and C. Y. Laporte, "Software engineering support activities for very small entities," in *Systems, Software and Services Process Improvement*, vol. 99. Berlin, Germany: Springer, Sep. 2010, pp. 165–176, doi: 10.1007/978-3-642-15666-3_15.
[62] F. J. Pino, O. Pedreira, F. García, M. R. Luaces, and M. Piattini, "Using scrum to guide the execution of software process improvement in small organizations," *J. Syst. Softw.*, vol. 83, no. 10, pp. 1662–1677, Oct. 2010, doi: 10.1016/j.jss.2010.03.077.
[63] R. V. O'Connor, "Exploring the role of usability in the software process: A study of Irish software SMEs," in *Proc. Eur. Conf. Softw. Process Improvement*, 2009, pp. 161–172.
[64] S. F. Ochoa, R. Robbes, M. Marques, L. Silvestre, and A. Quispe, "What differentiates Chilean niche software companies: Business knowledge and reputation," *IEEE Softw.*, vol. 34, no. 3, pp. 96–103, May 2017.
[65] L. M. A. Nascimento and G. H. Travassos, "Software knowledge registration practices at software innovation startups: Results of an exploratory study," in *Proc. 31st Brazilian Symp. Softw. Eng. (SBES)*, 2017, pp. 234–243.
[66] M. Muñoz, A. Peña, J. Mejía, G. P. Gasca-Hurtado, M. C. Gómez-Alvarez, and C. Y. Laporte, "Analysis of 13 implementations of the software engineering management and engineering basic profile guide of ISO/IEC 29110 in very small entities using different life cycles," *J. Softw., Evol. Process*, vol. 32, no. 11, Nov. 2020, doi: 10.1002/smr.2300.
[67] M. E. Morales-Trujillo and G. A. García-Mireles, "Evolving with patterns: A 31-month startup experience report," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 1037–1047.
[68] S. Lucho, K. Melendez, and A. Dávila, "Analysis of environmental factors in the adoption of ISO/IEC 29110. Multiple case study," in *Proc. Int. Conf. Softw. Process Improvement*, 2017, pp. 82–93.
[69] M. Muñoz, J. Mejia, and C. Y. Laporte, "Reinforcing very small entities using agile methodologies with the ISO/IEC 29110," in *Proc. Int. Conf. Softw. Process Improvement*, 2018, pp. 88–98.
[70] G. A. García-Mireles, "Addressing product quality characteristics using the ISO/IEC 29110," in *Trends and Applications in Software Engineering*, vol. 405. Cham, Switzerland: Springer, 2016, pp. 25–34, doi: 10.1007/978-3-319-26285-7_3.
[71] R. Souza, K. Malta, and E. S. De Almeida, "Software engineering in startups: A single embedded case study," in *Proc. 1st IEEE/ACM Int. Workshop Softw. Eng. Startups (SoftStart)*, Buenos Aires, Argentina, May 2017, pp. 17–23, doi: 10.1109/SoftStart.2017.2.
[72] P. Borges, P. Monteiro, and R. J. Machado, "Tailoring RUP to small software development teams," in *Proc. 37th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, Aug. 2011, pp. 306–309.
[73] M. A. Almomani, S. Basri, and A. R. Gilal, "Empirical study of software process improvement in Malaysian small and medium enterprises: The human aspects," *J. Softw., Evol. Process*, vol. 30, no. 10, p. e1953, Oct. 2018, doi: 10.1002/smr.1953.
[74] C. Y. Laporte, M. Munoz, J. M. Miranda, and R. V. O'Connor, "Applying software engineering standards in very small entities: From startups to grownups," *IEEE Softw.*, vol. 35, no. 1, pp. 99–103, Jan. 2018.
[75] G. C. L. Leal, R. Prikladnicki, C. Ebert, R. Balancieri, and L. B. Pompermaier, "Practices and tools for software start-ups," *IEEE Softw.*, vol. 37, no. 1, pp. 72–77, Jan. 2020.
[76] E. Klotins, M. Unterkalmsteiner, and T. Gorschek, "Software engineering antipatterns in start-ups," *IEEE Softw.*, vol. 36, no. 2, pp. 118–126, Mar. 2019, doi: 10.1109/MS.2018.227105530.
[77] J. Kasurinen and K. Smolander, "Defining an iterative ISO/IEC 29110 deployment package for game developers," *Int. J. Inf. Technol. Syst. Approach*, vol. 10, no. 1, pp. 107–125, Jan. 2017.
[78] J. Melegati, R. Chanin, A. Sales, R. Prikladnicki, and X. Wang, "MVP and experimentation in software startups: A qualitative survey," in *Proc. 46th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2020, pp. 322–325.
[79] S. Galván-Cruz, M. Muñoz, J. Mejía, C. Y. Laporte, and M. Negrete, "Building a guideline to reinforce agile software development with the basic profile of ISO/IEC 29110 in very small entities," in *Proc. Int. Conf. Softw. Process Improvement*, 2020, pp. 20–37.
[80] E. Carmel and S. Becker, "A process model for packaged software development," *IEEE Trans. Eng. Manag.*, vol. 42, no. 1, pp. 50–61, Feb. 1995.
[81] T. Clark and P.-A. Muller, "Exploiting model driven technology: A tale of two startups," *Softw. Syst. Model.*, vol. 11, no. 4, pp. 481–493, Oct. 2012, doi: 10.1007/s10270-012-0260-1.
[82] A. M. Sharif and S. Basri, "Risk assessment factors for SME software development companies in Malaysia," in *Proc. Int. Conf. Comput. Inf. Sci. (ICCOINS)*, Jun. 2014, pp. 1–5.
[83] Y.-M. García, M. Muñoz, J. Mejia, G. P. G. Hurtado, and A. M. Medina, "Application of a risk management tool focused on helping to small and medium enterprises implementing the best practices in software development projects," in *Trends and Advances in Information Systems and Technologies*, vol. 746. Cham, Switzerland: Springer, Mar. 2018, pp. 429–440, doi: 10.1007/978-3-319-77712-2_41.
[84] S. Basri and R. V. Oconnor, "Software development team dynamics in SPI: A VSE context," in *Proc. Asia–Pacific Softw. Eng. Conf. (APSEC)*, vol. 2, Dec. 2012, pp. 1–8, doi: 10.1109/APSEC.2012.26.
[85] F. J. L.-L. Hinojo, "Agile, CMMI, RUP, ISO/IEC 12207…Is there a method in this madness?" *ACM SIGSOFT Softw. Eng. Notes*, vol. 39, no. 2, pp. 1–5, Mar. 2014, doi: 10.1145/2579281.2579299.
[86] X. Larrucea, I. Santamaria, and B. Fernandez-Gauna, "Managing security debt across PLC phases in a VSE context," *J. Softw., Evol. Process*, vol. 32, no. 3, p. e2214, Mar. 2020.
[87] R. V. O'Connor and C. Y. Laporte, "Software project management in very small entities with ISO/IEC 29110," *Commun. Comput. Inf. Sci.*, vol. 301, pp. 330–341, Jun. 2012, doi: 10.1007/978-3-642-31199-4_29.
[88] T. Nonoyama, L. Wen, T. Rout, and D. Tuffley, "Cultural issues and impacts of software process in very small entities (VSEs)," in *Software Process Improvement and Capability Determination*, vol. 770. Cham, Switzerland: Springer, Oct. 2017, pp. 70–81, doi: 10.1007/978-3-319-67383-7_6.
[89] A. Mesquida and A. Mas, "A project management improvement program according to ISO/IEC 29110 and PMBOK?" *J. Softw. E*, vol. 26, no. 9, pp. 846–854, 2014.
[90] A. Boden, G. Avram, L. Bannon, and V. Wulf, "Knowledge sharing practices and the impact of cultural factors: Reflections on two case studies of offshoring in SME," *J. Softw., Evol. Process*, vol. 24, no. 2, pp. 139–152, 2012.
[91] S. M. Neves, C. E. S. Da Silva, V. A. P. Salomon, and A. L. A. Santos, "Knowledge-based risk management: Survey on Brazilian software development enterprises," in *Advances in Information Systems and Technologies*, vol. 206. Berlin, Germany: Springer, 2013, pp. 55–65, doi: 10.1007/978-3-642-36981-0_6.
[92] M. Yilmaz, R. V. O'Connor, and P. Clarke, "Software development roles: A multi-project empirical investigation," *ACM SIGSOFT Softw. Eng. Notes*, vol. 40, no. 1, pp. 1–5, Feb. 2015, doi: 10.1145/2693208.2693239.
[93] M. Verlage and T. Kiesgen, "Five years of product line engineering in a small company," in *Proc. 27th Int. Conf. Softw. Eng. (ICSE)*, May 2005, pp. 534–543.
[94] I. Garcia, C. Pacheco, M. Arcilla, and N. Sanchez, "Project management in small-sized software enterprises: A metamodeling-based approach," in *Trends and Applications in Software Engineering*, vol. 405. Cham, Switzerland: Springer, 2016, pp. 3–13, doi: 10.1007/978-3-319-26285-7_1.
[95] K.-K. Kemell, A. Elonen, M. Suoranta, A. Nguyen-Duc, J. Garbajosa, R. Chanin, J. Melegati, U. Rafiq, A. Aldaeej, N. Assyne, A. Sales, S. Hyrynsalmi, J. Risku, H. Edison, and P. Abrahamsson, "Business model canvas should pay more attention to the software startup team," in *Proc. 46th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2020, pp. 342–345.
[96] A. Tereso, C. P. Leão, and T. Ribeiro, "Project management practices at Portuguese startups," in *Proc. World Conf. Inf. Syst. Technol.*, 2019, pp. 39–49.
[97] S. S. Bajwa, X. Wang, A. N. Duc, and P. Abrahamsson, "'Failures' to be celebrated: An analysis of major pivots of software startups," *Empirical Softw. Eng.*, vol. 22, no. 5, pp. 2373–2408, Oct. 2017, doi: 10.1007/s10664-016-9458-0.
[98] M. Tuape, P. Ntebane, and P. Majoo, "Does context matter? Assessing the current state of quality practice during software development in small software companies," in *Proc. Future Technol. Conf.*, 2020, pp. 341–356.
[99] M. Muñoz, J. Mejia, and C. Y. Laporte, "Implementing ISO/IEC 29110 to reinforce four very small entities of Mexico under an agile approach," *IET Softw.*, vol. 14, no. 2, pp. 75–81, Apr. 2020.
[100] J. Mejia, E. Muñoz, and M. Muñoz, "Reinforcing the applicability of multi-model environments for software process improvement using knowledge management," *Sci. Comput. Program.*, vol. 121, pp. 3–15, Jun. 2016.
[101] S. Basri, M. A. T. Almomani, A. A. Imam, T. Murugan, A. R. Gilal, and A. O. Balogun, "The organisational factors of software process improvement in small software industry: Comparative study," in *Emerging Trends in Intelligent Computing and Informatics*, vol. 1073. Cham, Switzerland: Springer, Sep. 2019, pp. 1132–1143, doi: 10.1007/978-3-030-33582-3_106.

[102] A. Tosun, A. Bener, and B. Turhan, "Implementation of a software quality improvement project in an SME: A before and after comparison," in *Proc. 35th Euromicro Conf. Softw. Eng. Adv. Appl.*, Patras, Greece, 2009, pp. 203–209, doi: 10.1109/SEAA.2009.52.

[103] I. Garcia, C. Pacheco, J. A. Calvo-Manzano, and H. Hernández-Moreno, "Implementing the Ki Wo Tsukau model to strengthen the commitment of small-sized software enterprises in software process improvement initiatives," in *Proc. Int. Conf. Softw. Process Improvement*, 2016, pp. 3–12.

[104] M. Muñoz and J. Mejia, "Letting organizations to find the correct way to start in the implementation of software process improvements," in *New Contributions in Information Systems and Technologies*, vol. 353. Cham, Switzerland: Springer, 2015, pp. 503–512, doi: 10.1007/978-3-319-16486-1_49.

[105] A. Singh and S. S. Gill, "Measuring the maturity of Indian small and medium enterprises for unofficial readiness for capability maturity model integration-based software process improvement," *J. Softw., Evol. Process*, vol. 32, no. 9, p. e2261, Sep. 2020, doi: 10.1002/smr.2261.

[106] A. Nguyen-Duc, S. M. A. Shah, and P. Ambrahamsson, "Towards an early stage software startups evolution model," in *Proc. 42th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2016, pp. 120–127.

[107] X. Larrucea and I. Santamaria, "Comparing SPI survival studies in small settings," in *Proc. Int. Conf. Softw. Process Improvement Capability Determination*, 2017, pp. 45–54.

[108] T. Varkoi, "Process assessment in very small entities," in *Proc. 7th Int. Conf. Qual. Inf. Commun. Technol. Process*, 2010, pp. 436–440.

[109] Q. Boucher, G. Perrouin, J.-C. Deprez, and P. Heymans, "Towards configurable ISO/IEC 29110-compliant software development processes for very small entities," in *Systems, Software and Services Process Improvement*, vol. 301. Berlin, Germany: Springer, Jun. 2012, pp. 169–180, doi: 10.1007/978-3-642-31199-4_15.

[110] M. Sivashankar, A. M. Kalpana, and A. E. Jeyakumar, "A framework approach using CMMI for SPI to Indian SME'S," in *Proc. Int. Conf. Innov. Comput. Technol. (ICICT)*, Feb. 2010, pp. 1–5.

[111] R. Eito-Brun and M.-A. Sicilia, "Innovation-driven software development: Leveraging small companies' product-development capabilities," *IEEE Softw.*, vol. 33, no. 5, pp. 38–46, Sep. 2016, doi: 10.1109/MS.2016.63.

[112] S. Galván-Cruz, M. Mora, and R. O'Connor, "A means-ends design of SCRUM+: An agile-disciplined balanced SCRUM enhanced with the ISO/IEC 29110 standard," in *Proc. Int. Conf. Softw. Process Improvement*, 2017, pp. 13–23.

[113] M. Negrete, U. Infante, and M. Muñoz, "A case study of improving a very small entity with an agile software development based on the basic profile of the ISO/IEC 29110," in *Proc. Int. Conf. Softw. Process Improvement*, 2020, pp. 3–19.

[114] C. Y. Laporte and R. V. O'Connor, "A multi-case study analysis of software process improvement in very small companies using ISO/IEC 29110," in *Proc. Eur. Conf. Softw. Process Improvement*, 2016, pp. 30–44.

[115] P. Knauber, D. Muthig, K. Schmid, and T. Widen, "Applying product line concepts in small and medium-sized companies," *IEEE Softw.*, vol. 17, no. 5, pp. 88–95, Sep. 2000, doi: 10.1109/52.877873.

[116] T. Besker, A. Martini, R. E. Lokuge, K. Blincoe, and J. Bosch, "Embracing technical debt, from a startup company perspective," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2018, pp. 415–425, doi: 10.1109/ICSME.2018.00051.

[117] M. Chicote, "Startups and technical debt: Managing technical debt with visual thinking," in *Proc. IEEE/ACM 1st Int. Workshop Softw. Eng. Startups (SoftStart)*, May 2017, pp. 10–11, doi: 10.1109/SoftStart.2017.6.

[118] F. Silva, R. Souza, and I. Machado, "Taming and unveiling software reuse opportunities through white label software in startups," in *Proc. 46th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2020, pp. 302–305.

[119] L. Riungu-Kalliosaari, O. Taipale, and K. Smolander, "Testing in the cloud: Exploring the practice," *IEEE Softw.*, vol. 29, no. 2, pp. 46–51, Mar. 2012.

[120] M. Felderer and R. Ramler, "Risk orientation in software testing processes of small and medium enterprises: An exploratory and comparative study," *Softw. Qual. J.*, vol. 24, no. 3, pp. 519–548, Sep. 2016, doi: 10.1007/s11219-015-9289-z.

[121] R. Kaushik, C. J. M. Tauro, V. D. Souza, and K. Bhowmick, "A novel approach for collaborative last-mile performance testing implementation using an object-oriented approach," *ACM SIGSOFT Softw. Eng. Notes*, vol. 39, no. 2, pp. 1–4, 2014.

[122] L. Mathiassen and A. M. Vainio, "Dynamic capabilities in small software firms: A sense-and-respond approach," *IEEE Trans. Eng. Manage.*, vol. 54, no. 3, pp. 522–538, Jul. 2007, doi: 10.1109/TEM.2007.900782.

[123] U. Rafiq *et al.*, "Requirements elicitation techniques applied in software startups," in *Proc. 43rd Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, 2017, pp. 141–144, doi: 10.1109/SEAA.2017.73.

[124] C. Y. Laporte and R. V. O'Connor, "Implementing process improvement in very small enterprises with ISO/IEC 29110: A multiple case study analysis," in *Proc. 10th Int. Conf. Qual. Inf. Commun. Technol. (QUATIC)*, Sep. 2016, pp. 125–130.

[125] M.-L. Sánchez-Gordón and R. V. O'Connor, "Understanding the gap between software process practices and actual practice in very small companies," *Softw. Qual. J.*, vol. 24, no. 3, pp. 549–570, Sep. 2016, doi: 10.1007/s11219-015-9282-6.

[126] V. Vukovic, J. Djurkovic, M. Sakal, and L. Rakovic, "An empirical investigation of software testing methods and techniques in the province of vojvodina," *Tehnički Vjesnik*, vol. 27, no. 3, pp. 687–696, 2020.

[127] A. L'Erario, H. C. S. Thomazinho, and J. A. Fabri, "An approach to software maintenance: A case study in small and medium-sized businesses IT organizations," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 30, no. 5, pp. 603–630, May 2020.

[128] J. K. Balikuddembe and M. Tuape, "An ambiguity minimization technique during requirements elicitation phase," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2017, pp. 945–950, doi: 10.1109/CSCI.2017.164.

[129] V. Kettunen, J. Kasurinen, O. Taipale, and K. Smolander, "A study on agility and testing processes in software organizations," in *Proc. 19th Int. Symp. Softw. Test. Anal. (ISSTA)*, 2010, pp. 231–240, doi: 10.1145/1831708.1831737.

[130] C. Wohlin, D. Šmite, and N. B. Moe, "A general theory of software engineering: Balancing human, social and organizational capitals," *J. Syst. Softw.*, vol. 109, pp. 229–242, Nov. 2015.

[131] S. M. Sutton, "The role of process in software start-up," *IEEE Softw.*, vol. 17, no. 4, pp. 33–39, Jul. 2000.

[132] R. Xue, C. Baron, and P. Esteban, "Optimising product development in industry by alignment of the ISO/IEC 15288 systems engineering standard and the PMBoK guide," *Int. J. Prod. Dev.*, vol. 22, no. 1, pp. 65–80, 2017.

[133] C. Y. Laporte and R. V. O'Connor, "Systems and software engineering standards for very small entities: Accomplishments and overview," *Computer*, vol. 49, no. 8, pp. 84–87, Aug. 2016.

[134] P. Clarke, R. V. O'Connor, and B. Leavy, "A complexity theory viewpoint on the software development process and situational context," in *Proc. Int. Conf. Softw. Syst. Process*, May 2016, pp. 86–90.

[135] P. Ralph, "The sensemaking-coevolution-implementation theory of software design," *Sci. Comput. Program.*, vol. 101, pp. 21–41, Apr. 2015.

[136] R. V. O'Connor and C. Y. Laporte, "The evolution of the ISO/IEC 29110 set of standards and guides," *Int. J. Inf. Technol. Syst. Approach*, vol. 10, no. 1, pp. 1–21, Jan. 2017, doi: 10.4018/IJITSA.2017010101.

[137] C. Giardino, S. S. Bajwa, X. Wang, and P. Abrahamsson, "Key challenges in early-stage software startups," in *Proc. Int. Conf. Agile Softw. Develop.*, 2015, pp. 52–63.

[138] C. Y. Laporte, R. V. O'Connor, and L. H. García Paucar, "Software engineering standards and guides for very small entities–implementation in two start-ups," in *Proc. 10th Int. Conf. Eval. Novel Approaches to Softw. Eng.*, Barcelona, Spain, 2015, pp. 5–15, doi: 10.5220/0005368500050015.

[139] J. Melegati, "What influences software startups to use lean startup?" in *Proc. 19th Int. Conf. Agile Softw. Develop., Companion*, May 2018, pp. 1–3.

[140] K.-K. Kemell, V. Ravaska, A. Nguyen-Duc, and P. Abrahamsson, "Software startup practices–software development in startups through the lens of the essence theory of software engineering," in *Product-Focused Software Process Improvement*, vol. 12562. Cham, Switzerland: Springer, Nov. 2020, pp. 402–418, doi: 10.1007/978-3-030-64148-1_25.

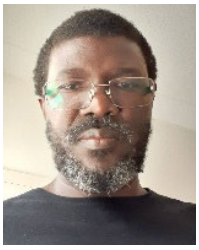
[141] Y. S. Lincoln and N. K. Denzin, *The Sage Handbook of Qualitative Research*. Newbury Park, CA, USA: Sage, 2011.

[142] S. J. Taylor, R. Bogdan, and M. DeVault, *Introduction to Qualitative Research Methods: A Guidebook and Resource*. Hoboken, NJ, USA: Wiley, 2015.

[143] R. Whittemore, S. K. Chase, and C. L. Mandle, "Validity in qualitative research," *Qual. Health Res.*, vol. 11, no. 4, pp. 522–537, 2001.

[144] B. Kitchenham, H. Al-Khilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, and L. Zhu, "Evaluating guidelines for reporting empirical software engineering studies," *Empirical Softw. Eng.*, vol. 13, no. 1, pp. 97–121, Feb. 2008.

[145] T. Dybå, R. Prikladnicki, K. Rönkkö, C. Seaman, and J. Sillito, "Qualitative research in software engineering," *Empirical Softw. Eng.*, vol. 16, no. 4, pp. 425–429, Aug. 2011, doi: 10.1007/s10664-011-9163-y.
[146] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009.
[147] B. C. O'Brien, I. B. Harris, T. J. Beckman, D. A. Reed, and D. A. Cook, "Standards for reporting qualitative research: A synthesis of recommendations," *Academic Med.*, vol. 89, no. 9, pp. 1245–1251, Sep. 2014, doi: 10.1097/ACM.0000000000000388.
[148] J. S. Molléri, K. Petersen, and E. Mendes, "CERSE–catalog for empirical research in software engineering: A systematic mapping study," *Inf. Softw. Technol.*, vol. 105, pp. 117–149, Jan. 2019.
[149] M. Tuape and Y. Ayalew, "A roadmap for a comparison framework for an adaptable software process improvement framework in small software companies," *Ann. Comput. Sci. Inf. Syst.*, vol. 20, pp. 133–141, 2019.



MICHEAL TUAPE (Member, IEEE) was born in Pakwach, Uganda, in March 1979. He received the Bachelor of Science degree (Hons.) in information technology from Uganda Christian University Mukono, Uganda, and the Master of Science degree in software engineering from Makerere University, Kampala, Uganda. He is currently pursuing the Doctor of Science degree in technology and software engineering with Lappeenranta-Lahti University of Technology, Lappeenranta, Finland.

From 2015 to 2018, he was a Research Associate with the Software Systems Center, Makerere University, Kampala. From 2018 to 2020, he was a Research Associate in the UIG Project with the University of Botswana, Gaborone, Botswana. He has been a Junior Researcher at Lappeenranta-Lahti University of Technology, since 2020. His research interests include software engineering, requirements engineering, software development process, small software companies, and open science.

Mr. Tuape is a member of ACM and PMI.

VICTORIA T. HASHEELA-MUFETI received the Bachelor of Science degree in computer science and economics from the University of Namibia, in 2005, the Bachelor of Science degree (Hons.) in computer science from Stellenbosch University, South Africa, in 2007, the Master of Science degree in informatics from the University of Mannheim, Germany, in 2010, and the Doctor of Science degree in technology from Lappeenranta University of Technology, Finland, in 2018.

She was a Fulbright Scholar at The University of New Mexico, USA, in 2018. She is currently a Senior Lecturer with the University of Namibia. Her research interests include digital preservation of African indigenous knowledge and languages, software development for SMEs, and data analytics.

ANNA KAYANDA received the Bachelor of Science degree in computer science from the University of Dar es Salaam, Tanzania, in 2009, and the Master of Science degree in computer science from the University of Mysore, India, in 2013. She is currently pursuing the Ph.D. degree in science, technology and computing with the University of Eastern Finland, Finland.

She is working as an Assistant Lecturer with the College of Business Education. Her research interest includes information systems development for decision support.

JARI PORRAS (Member, IEEE) received the D.Sc. (Tech.) degree from Lappeenranta University of Technology, Finland, in 1998, with a focus on modeling and simulation of communication networks in distributed computing environment.

He is currently a Professor of software engineering (especially distributed systems) with Lappeenranta-Lahti University of Technology (LUT). He has supervised approximately 500 master's thesis works and 22 dissertations as well as acted as an External Evaluator for 21 doctoral thesis works since the start of his professorship. He has conducted research on parallel and distributed computing, wireless and mobile systems and services, and sustainable ICT. In last years, he has focused his research on human and sustainability aspects of software engineering. He is actively working in international networks and organizations.

JUSSI KASURINEN was born in Savonlinna, Finland. He received the master's degree in information technology from Lappeenranta University of Technology, in 2007, and a Doctor of Science (Tech) from Lappeenranta University of Technology (LUT), Lappeenranta, Finland, in 2011.

He is currently a Doctor of science (technology), specializing in software engineering and software testing. He is an Adjunct Professor of entertainment software engineering. He works as an Associate Professor and the Head of degree programs in software engineering with LUT School of Engineering Sciences. During his career, he has authored over 50 scientific publications in various topics of software engineering and four non-fiction books discussing programming languages and software testing. He is the current LUT University Representative of the Finnish Software Measurement Association (FiSMA), Computer Science Association of Finland, and Academic Engineers and Architects in Finland (TEK). He received his adjunct professorship from LUT University, in 2017. His current research interests include but are not limited to smart systems for software engineering, games from the viewpoint of software, software testing practices, and software process quality.