Software Development in Small Software Companies: Exploring the Usage of Procedures, Techniques, Methods and Models in Practice

Micheal, Tuape
Dept of Software Eng. Lappeenranta-Lahti University of Technology, Lappeenranta
micheal.tuape@lut.fi

Victoria, Hasheela-Mufeti
Dept of Computing, Mathematical and Statistical Sciences, University of Namibia
vhasheela@unam.na

Anna, Kayanda
Information Systems Dept, College of Business Education, Dar es Salaam
a.kayanda@cbe.ac.tz

Jussi, Kasurinen
Dept of Software Eng. Lappeenranta-Lahti University of Technology, Lappeenranta
Jussi.kasurinen@lut.fi

ABSTRACT

Small software companies have a challenge with utilizing process tools, which affects practice with significant quality-related challenges. This affects the software industry significantly because SSCs dominate the industry, and most of all, over 80 percent of software products are produced by SSCs. This cross-sectional survey was conducted in 3 countries attracting 115 respondents with the primary objective of investigating the software practice concerning the utilization of process tools in SSCs. The study focused on the tools used in requirements engineering and software testing as critical process areas for quality software products. Our findings indicate that the number of personnel intertwines with the complexities arising from lengthy procedures of the tools and processes, aggregating into difficulty in tool usage. Due to the constant evolution of practices, the volatility in processes also causes slow adoption of other tools, for instance, testing that must accompany the main engineering tools during a project. These findings are significant in informing theory and communicating to the practitioners what they should do regarding process tools.

CCS CONCEPTS

• **Software and its engineering**; • **Software creation and management**; • **Designing software**;

KEYWORDS

Small software companies, Utilization of process tools, Software development practice, Survey methodology

ACM Reference Format:
Micheal, Tuape, Victoria, Hasheela-Mufeti, Anna, Kayanda, and Jussi, Kasurinen. 2021. Software Development in Small Software Companies: Exploring the Usage of Procedures, Techniques, Methods and Models in Practice. In *2021 2nd European Symposium on Software Engineering (ESSE 2021), November 19–21, 2021, Larissa, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3501774.3501779

ESSE 2021, November 19–21, 2021, Larissa, Greece

ACM ISBN 978-1-4503-8506-0/21/11…$15.00
https://doi.org/10.1145/3501774.3501779

1 INTRODUCTION

Software engineering continues to face challenges amidst increased demand for software-intensive projects[23], including challenged projects, poor quality products with all attempts to transform processes with the different tools available for practice that continue to pose new challenges, especially for the Small Software Companies (SSCs) given the complex environment in which they operate[41].

Unlike other engineering fields, software engineering is a relatively new field that is very knowledge-intensive [15] [42]. This industry involves lots of design and is hugely dependent on people [14], which probably explains why the software industry faces different challenges and complexities [36]. Based on this background, the software failure rate is still high in the industry and has been this linked to poor practice where up to 40% of projects are considered failures [27]. Quality is another aspect that has been a challenge in the software industry [18] because of poor processes, as indicated by [17], causing delay to market, cost overruns, dissatisfaction of customers leading to the cancellation of projects. Today, the software industry is becoming global, putting pressure on the young industry, leading to high expectations and increasing demand for quality software products.

Creativity has been experienced in practice and research, with different solutions being suggested, primarily where the challenges in practice affect software engineering most. These challenges are aggregated by the underutilization of the tools that facilitate software development processes; the techniques, approaches, notations and methods prevalent for SSCs are very evident, with researchers [2] citing a low 7 percent usage. Authors have suggested that tools are often not easily adaptable for the SSCs, which leads to limited utilization. This is perhaps because most tools are made with the assumption that all software companies are the same. A typical example is the introduction of agile methods [8], which has effectively gained prominence among practitioners and researchers, with some researchers calling it a paradigm shift [20]. In support of this, Stankovic et al., in [36] are quoted as saying that "agile methods is not about the practices they use, but rather the recognition of people as principal drivers of project success, added to the intense

focus on effectiveness and maneuverability." This means the tools may remain the same, but the focus should be on ensuring that the tools are adaptable to have the processes utilized in practice. Similarly, software process improvement(SPI) has also taken credit as a key contributor in transforming practice; however, Tripathi et al. [37] also note that successfully implementing SPI is arguably the most challenging issue facing the SPI field today and most especially for the SSCs.

There is also a known concern that studies to understand software practice have been minimal, and most areas of software remain under-researched with glairing evidence of the gap in practice and researched information. Among areas that remain under-researched is software testing[20][21] and requirements engineering (RE)[39], whose adequate definition creates a significant impact on the general understanding of the product under construction and on the final product quality [5][13]. Significantly, 60 percent of development effort is used while generating system requirements for the system under construction, and most of the projects are challenged because of unsatisfactory requirements [13]. The tools available for RE are significant for practice; however, the usage of the tools is limited. Researchers in [13][16] have studied the effectiveness of the tools, citing, and acknowledging that RE's success greatly depends on the people involved in it. Software practitioners consider motivation, domain knowledge, attitude, communication skills and personality as critical human aspects [16] when involved in RE and development of software generally.

This study seeks to investigate software practice, creating a deeper understanding of why there is limited utilization of process tools and what tools are most utilized by the SSCs. Another objective is to investigate further what motivates the decision on the tools that the organizations often utilize. The other aspect of interest in this study is to ultimately develop a process adoption framework to provide developers of software process tools with a strategy to ensure that the tools contend with the context in which SSCs operate. Therefore, to do this, understanding software practice and the utilization of tools by the SSCs is paramount. This study is a survey in which 115 software development practitioners from software companies in 3 African countries, namely: Namibia, Tanzania and Ghana, have been reached through an online questionnaire to answer questions related to software practice in SSCs.

The rest of this paper is structured such that, section 2 presents the related work, section 3 the methodology is discussed, section 4 is the presentation of the results, section five is the discussion of the results and limitations of the study; and finally, section 6 covers the conclusion and future work.

2 RELATED WORK

Different researchers have attempted to explore many avenues in explaining the plight to understand the practice, especially process tools in software engineering. It is important to note that there are limited studies on software practice, especially those dedicated to studying the SSCs. Most of the available studies have paid significant attention to the larger companies and their practices in general. It is also evident that questions on what process tools are often used in practice and how they arrive at the choice of process tools remain a mystery. This has led to other researchers [21] questioning processes research while arguing that process-related research has lost momentum and has failed to aligned with practice concerns.

Generally, on software practice and the associated tools used in software development, researchers in [1] contend that the field of software development is not shy of introducing new methodologies; to this effect, in the last 25 years alone, different approaches to software development have been introduced, of which only a few have survived up today. Researchers with similar views also argue that development methodologies "are treated primarily as a necessary fiction to present an image of control or provide a symbolic status[27][23]." Similarly, other researchers [1, 31, 34] front the argument that traditional methods are unattainable ideals that only provide normative guidance to utopian development situations.

Some researchers give significant attention to SSCs, especially regarding processes and practice, among others are Wangenheim et al. in [42], who stated that many SSCs are unaware of existing software process assessment models and standards that are primarily responsible for the transformation of software development practice in SSCs. In addition, they have also argued that such procedures have been criticized as inappropriate for SSCs, which generally have informal processes and organizational structures that focus primarily on getting the product out to stay in business which is a concern of other researchers who have also explored the practice of SSCs observed in [41] [38][29].

Similarly, in [33], the researchers highlight the challenges of SSCs characteristics that clearly distinguish them from larger organizations and translate into restrictions for selecting engineering tools. They also add that the tools are constantly increasing in number and, unfortunately, are designed to adapt to different scopes, making the selection of the appropriate tools much more complex. On the differences in context and the associated difficulty in implementing tools in practice, Sánchez-Gordón and O'Connor in [35] also contend that all software companies are not the same and vary according to different factors including size, market sector, time in business, management style, product range and geographical location and that since all companies are not the same, then the critical questions for those who develop software process and tools should be questioning why the tools they have to be the same

Other researchers in [37] also add that SSCs are critical in driving the software industry today and are responsible for the rapid growth of most growing economies. The SSCs operate under challenging environments with difficulty utilizing tools and processes. The authors call for the need to assist the SSCs in seeking the relevant information to make their software development processes efficient because they lack systematic process knowledge for determining which type of processes and tools are more relevant to their context.

Some researchers have also undertaken studies from the scope of our study related to software practice in Nelulu and Mufeti [28] report software development practice in a study from Namibia. The authors share similar concerns of low utilization of process tools and point out agile methods during development. Similarly, in another study, Mushashu and Mtebe [25] also study the software practice in Tanzania and report the usage of traditional methods compared to the agile methods. The authors tag the reason for choosing the traditional methods to the type of projects handled by the companies. In a paper proposing a software maintenance

model, the authors hint at the poor practice in software development in Ghana[18]. Other studies related to software practice in other African countries include [2][26].

However, the literature reviewed in this work highlights the challenges of interest in this study, although the studies do not seek to understand what the practitioners are specifically doing on utilizing tools and what prompts the decision to choose a tool during software development. This gap remains unexploited, and we believe answering this will be a positive step in understanding practice in SSCs.

3 METHODOLOGY

This is a quantitative design study to help us answer the questions regarding software practice in SSCs. A cross-sectional survey with close-ended questions answered with a type 5 Likert scale. Descriptive studies like this are commonly used in software practice research [16, 30]; we would then understand the characteristics of practitioners and their usage of software tools in software practice in SSCs, which has limited studies. This will enable us to answer the what and why questions then pave the way for the how questions that will come up in the subsequent studies.

3.1 Population and Sample

Software-intensive companies were considered as the target population for this study. This population includes companies of different sizes developing software products for a wide variety of markets. This survey was sent out to a total of 282 companies in Namibia (84), Tanzania (95) and Ghana (103). We received 47, 38 and 30, totaling 115 data points, representing 26, 33, and 41 percent as represented in Figure 1. The respondents were reached through purposive sampling based on a criterion whose characteristics were defined for a purpose that is relevant to the study [4]. The main elements considered for the criterion while selecting the companies were; the number of persons in the company fitting the definition of SSCs (under 50 persons), the type of software-intensive products the company produces, and the respondent's role as being directly involved in the development of software. Purposive sampling is typical in software engineering studies as advocated for in [7] and used in [9] and [8]. The participants were contacted by telephone to request participation in the study before mailing the questionnaire link on the Webropol survey system. In attempt to keep the questionnaire as short as possible (a pilot study of the survey questionnaire showed that respondents needed about 15 to 20 minutes to complete it). We combined several well-proven techniques for improving the response rate of mailed questionnaires. Furthermore, this represents an effective response rate of 40.8 percent, which is about the minimum of 40 percent for representatives of organizations as suggested in [6].

3.2 Characteristics of respondents

Although there was a set criterion for identifying the participants, parts of the survey questionnaire asked characteristic questions to define the characteristics of the respondents. To understand the characteristics of the respondents we examined the type of software products produced by the participant's company, the gender, the level of education and the country from which the data was collected as shown in Figure 1.

Most of the companies produce more than one software-intensive product, predominantly software solutions and web applications. The developers and software engineers are the dominant roles taken by most of the participants in the study, with 82 and 52 participants, respectively.

The other aspect significant about the characteristics of the participants is the gender and number of personnel of their respective companies shown in Figure 2

The results show that, 85 percent of the participants came from companies that fall under the definition of SSCs with 1 to 50 personnel, and 70 percent of them are from the very small entities with between 0 and 25 personnel. However, the 0 to 5 category remain glaringly significant, with up to 41 percent of the participants belonging to this category.

3.3 Survey questions

The online survey questionnaire was designed to investigate software practice, specifically the tools used in practice by the SSCs. We developed a draft set of questions to comprehensively cover the software practice, considering the questionnaire's size and the number of questions, following the guidelines and experiences of other researchers in conducting online surveys [15].

The questionnaire was sent to fifteen practitioners in the industry to ensure that the language used in the study was familiar to the intended participants. Evidence in the literature shows that [13] researchers and industrial practitioners often use different terminologies and, when conducting industry research, consensus on terminology is paramount.

Early feedback from the practitioners helped ensure the proper context for the survey and the familiarity with its terminology for industrial respondents. This was used to develop the second set of survey questions subjected to review by five other experts with over five years of experience conducting software engineering surveys. After improving the questions based on the feedback from the industry practitioners in the pilot study, the instrument had 28 questions. A set of questions was dedicated to the profiles and demographics of the respondent. The questions probed data related to software companies in the context of characteristics, software development tools used in software practice with answers of type 5-point Likert scale giving participants options of 1. Never, 2. Rarely, 3. Sometimes, 4. Often, and 5. Always.

3.4 Assessment of Reliability and Validity

Cronbach's alpha α proposed in [11] is the most common measure of internal consistency ("reliability"). It is mostly used with multiple Likert questions in a survey that form a scale, and you wish to determine if the scale is reliable. Reliability refers to the consistency and stability of the score from a measurement scale, it is calculated using formular 1 below.

$$\alpha = \frac{k}{k-1}\left[1 - \frac{\sum s^2 y}{s^2 x}\right] \tag{1}$$

The reliability of the 64 sub-questions measured on the 5 Likert scale given a dataset of 115 was evaluated by internal consistency

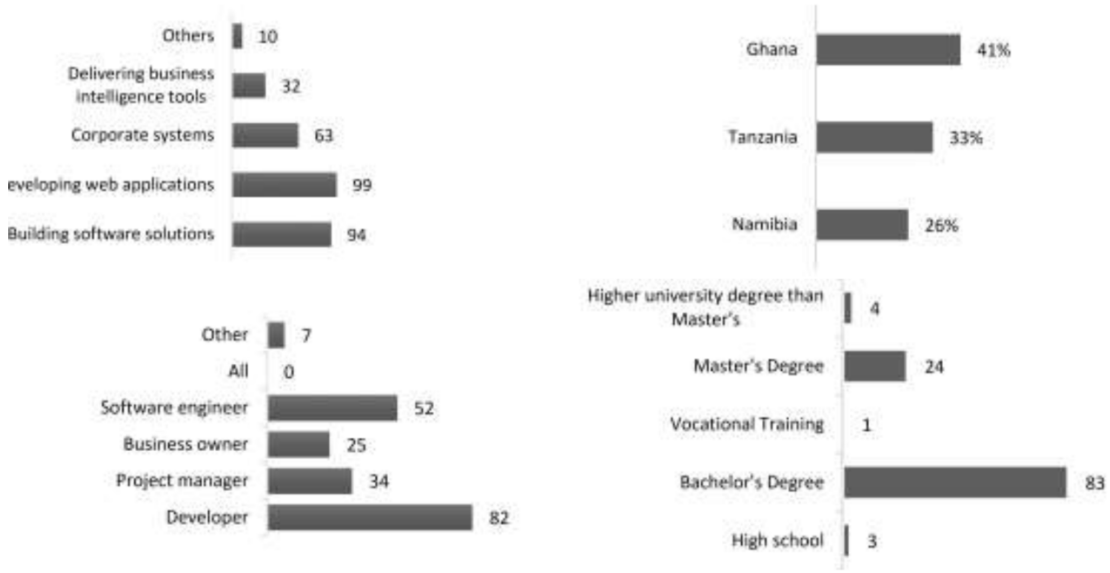

Figure 1: L-R top; respondents' responses on the software-intensive products produced by the companies, the countries from which the data was collected—L-R bottom roles of respondents in the company, level of education of the respondents (n=115)

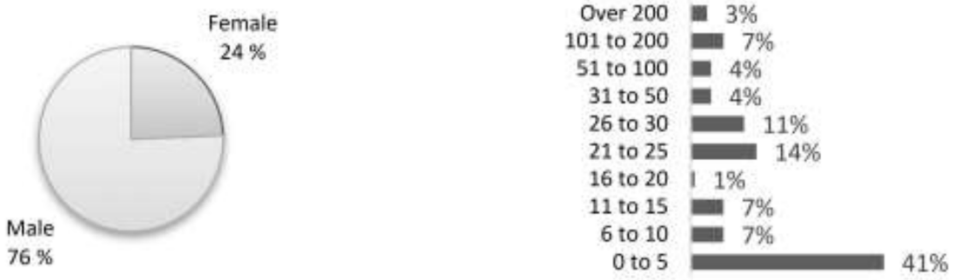

Figure 2: Left to right gender representation of the respondents and the number of personnel in the companies in the study (n=115)

Table 1: Assessment of reliability and validity using the Cronbach's alpha

Variables	Description	Value	Internal consistency
K	Number of Items	64	
$\sum s^2y$	Sum of item variance	88,17	Acceptable
s^2x	Variance of the total score	1590,74	
α	Cronbach's Alpha	0,96	

analysis, using coefficient alpha α [23]. The result of the Cronbach's alpha α interpretation was evaluated to .92, see Table 1 which means that our questions' internal consistency or reliability is acceptable. According to the rule, the ranges are from zero to one; however, .7 and above is acceptable. In cases with large datasets, this α is expected to be high closer to one.

3.5 Data Analysis Techniques

We used Cross-tabulation, sometimes referred to as cross-tab or contingency table, as a tool used to analyze categorical data. This type of data involves values that are mutually exclusive to each other. Data was collected in numbers, but numbers have no value unless they mean something as used in [17],

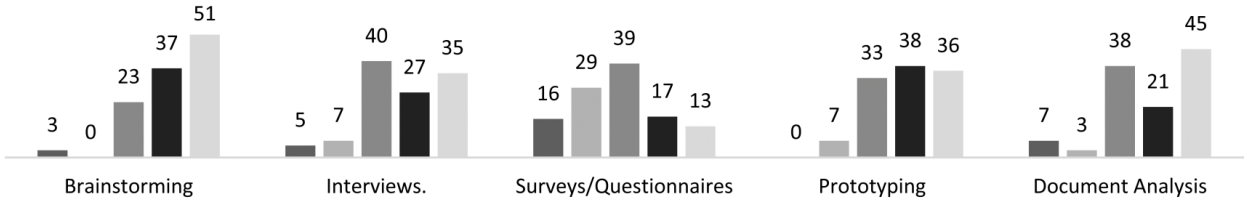

Figure 3: Responses of the question on the techniques for requirements elicitation by the SSCs (n=115).

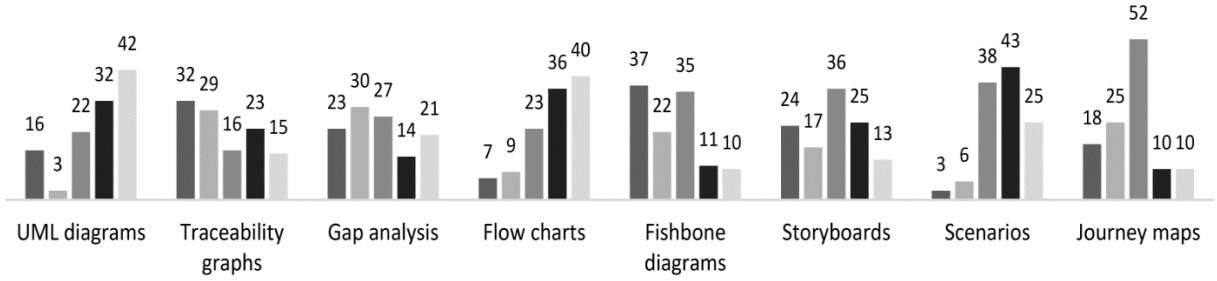

Figure 4: responses on the question on approaches used during software requirements analysis and modelling (n=115).

The results of testing these assumptions showed that kurtosis, skewness, and the one-sample Kolmogorov-Smirnov tests for all variables were within the acceptable range for the normal distribution assumption. Also, the assumptions of homoscedasticity, linearity, and independence of the error terms were supported, and no influential observations were identified. In other words, there were no extreme violations to the basic assumptions underlying the chosen data analysis techniques that could justify the use of less powerful nonparametric statistics. All quantitative analyses were conducted using SPSS.

4 RESULTS

Process tools are general implements used to enhance one's ability in building, modifying, and generally developing software. The tools include techniques, approaches, notations, and methods.

4.1 Requirements

4.1.1 Techniques used for requirements elicitation. Figure 3 represents the common techniques among the different techniques in eliciting requirements: brainstorming, interviews, surveys or questionnaires, prototyping, and document analysis. Eliciting requirements, the results indicate minimal usage of the techniques of surveys and questionnaires compared to the other. Brainstorming is, however, better utilized.

4.1.2 Approaches we use during the analysis and modelling of software requirements. Figure 4 illustrates the responses on the approaches that guide analysis and modelling practice, including UML diagrams, Flow charts and Scenarios, which are seen as more utilized than Traceability graphs, Gap analysis, and Storyboards minimally used. In contrast, Fishbone diagrams and Journey maps are the least used.

4.1.3 Notations used in expressing requirements specifications. Figure 5 presents the results of the notations used by the SSCs in the requirements specification. There is an overall low usage of the notations.

Structured natural language that uses standard Templates/form presents as a preferred choice, Design description language and Graphical notation which is also sometimes Text annotations are moderately used while, Mathematical Specifications with finite state machines/sets is minimally used.

4.1.4 Our team performs requirements inspections in your software projects. Inspection is a powerful method to uncover ambiguities, in which different readers interpret a requirement in different ways. If the reader has difficulty describing a requirement, perhaps it is too complex, poorly expressed, or incorrect. Figure 6 shows this study's key requirements inspection techniques: scenarios, ad hoc techniques, checklists, and the sample-driven technique.

4.2 Software development methods

The study asked whether Crystal Family, Nexus, DevOps, Dynamic Systems Development Method, Kanban, Lean Software Development, Scaled Agile Framework (SAFe), Scrum, ScrumBan, Spiral Model, Structured Systems Analysis and Design Method, Team Software Process, V-shaped Process (V-Model), Iterative Development, Personal Software Process and PRINCE2 were used. See Figure 7. The results show limited usage of the development methods by the companies that participated in the study. The Iterative development, scrum, DevOps, and SAFe recorded more than 40 respondents who have often used these methods. While the methods like Crystal Family, Nexus, Kanban, ScrumBan, V-Model, and Prince 2 are list utilized with over 70 respondents stating to have never or rarely used the methods.

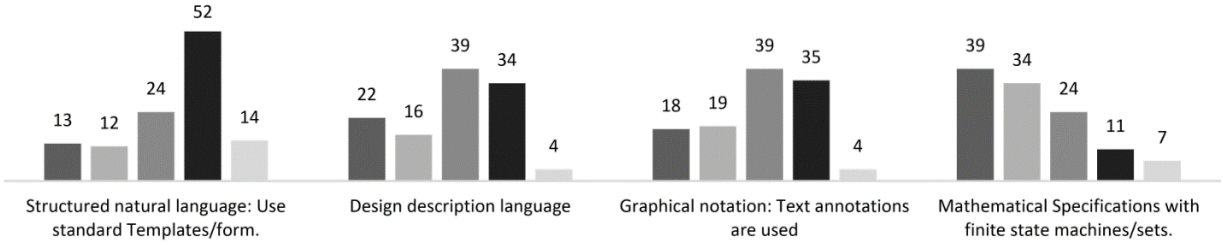

Figure 5: responses on the question on the notations used while expressing requirements specifications(n=115).

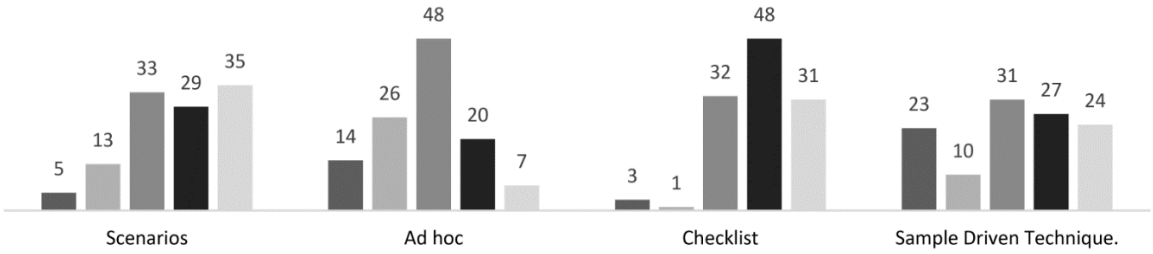

Figure 6: Requirements inspections techniques used in software projects (n= 93)

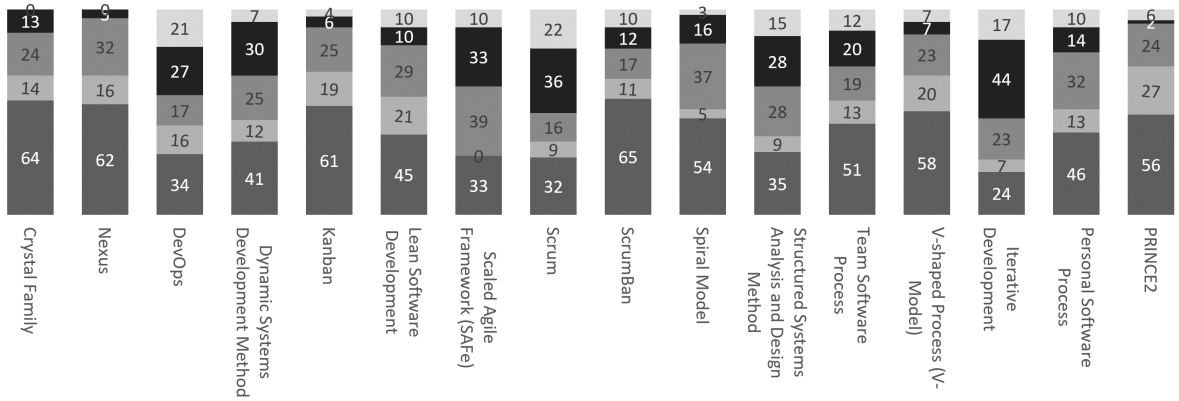

Figure 7: Software development methods n= 115

4.3 Software testing

4.3.1 Software testing strategy. Figure 8. shows the availability of a software testing strategy within the SSCs and how it is utilised.

We examined the availability of test strategies in the SSCs and the strategies' utilization on software testing. The results present data from the 93 companies that SSCs and we not that generally, 64 respondents answer in affirmative while 29 state that they do not have a test strategy in their companies representing 69 and 31 percent respectively.

4.3.2 Referral to company test strategy. The responses whether the participants referred to their test strategies were answered by 64 respondents, of which 38, 31, 36, 3 and 5 responded with Always, often, sometimes rarely and never, respectively, as illustrated above.

4.3.3 Models for software testing. Figure 9 shows that 3 of the software models have never been used by about 60 respondents, and 45 have also not used ISO/IEC29119. The difference is not significant, although there is a little higher usage of ISO/IEC29119 than the other three asked in the questionnaire.

5 DISCUSSION

This study reports quantitative research undertaken to investigate software practice, specifically the utilization of software process tools as techniques, models, methods, and processes. This section

Figure 8: Left representation of percentages of the respondents who confirmed having a software test strategy in the SSCs. n= 93; Right the usage of the strategy of those who confirmed to have a strategy n= 64

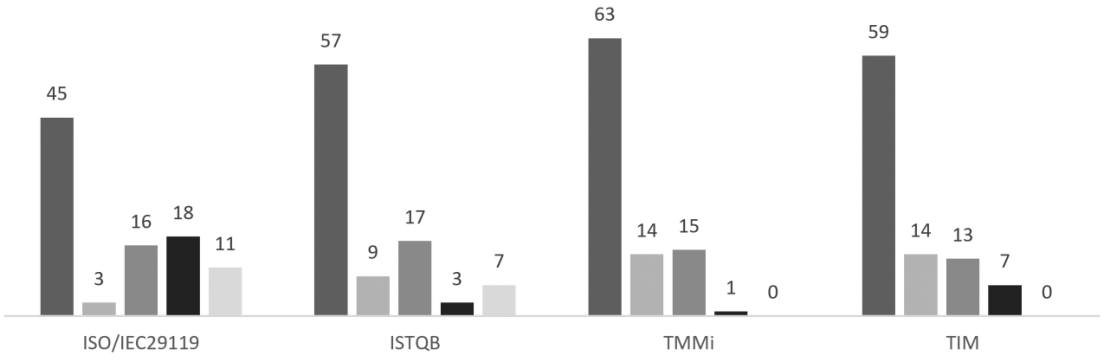

Figure 9: Responses of participants on the models used by the SSCs in software testing n= 93

discusses the insights gained for practitioners and researchers from the findings; we also further discuss the limitations of the study and some suggestions for further research

5.1 Insights from the researcher's perspective

5.1.1 Requirements Engineering tools in practice. The low usage of the requirements elicitation techniques is worrying, yet many stakeholders cannot accurately articulate the business needs and problems. This, therefore, means that, while performing the elicitation, it may be challenging to ensure that the requirements produced are understandable, practical, and relevant, a view shared in [39]. This confirms claims of rampant project failure being associated with requirements in [24]. This also shows a clear linkage to the findings that show low usage of notations in requirements specification and inspection, respectively. The unutilized tools are solutions to the challenges observed; this leaves us with the quest to ask why the tools are not being used yet there is a need.

Requirement elicitation techniques are skillful ways to carry out the process of collecting information from stakeholders. From this, we determine the customers' needs for the proposed system. Given the challenges from the industry as cited in [5] and [39], as it is expected. Eliciting requirements is considered by practitioners and researchers as one of the most challenging tasks in developing software majorly because establishing one's interest usually is very difficult when it comes to software. This procedure serves as a foundation in documenting the requirements for application development.

In linguistics and semiotics, a notation is a system of graphics or symbols, characters, and abbreviated expressions, expected to be helpful in software development practice to represent technical facts by convention as cited in[10]. Notations allow us to write and make calculations large numbers quickly and with minimal effort. Symbols play crucial roles in advanced mathematical thinking that they provide flexibility and reduce cognitive load, although often have a dual nature since they signify both processes and objects of mathematics. In software, engineering notations are expected to be significant in abstracting complexity, although in most cases, it is seen as a complex approach, which probably explains why its usage is very insignificant among SSCs.

The tools unutilised in the during RE require substantial time and are highly procedural with numerus formalities the trend seen here is that tools that enable what is perceived as "flexibility" seems to carry the day regardless of how useful a tool may be.

5.1.2 Software developing methodologies. Our findings have some similarities with the findings of [26], who study the utilization of development methodology. Their finding reported a low usage of methods and significantly low usage of the agile approaches in

Uganda compared to Sweden. Uganda has similar traits with the geographical scope of this study we see that software companies in Uganda are prone to utilizing traditional development methodologies. Because of proximity, the software practice can have a familiar pattern.

In another study [25], in Tanzania, the authors also found out that the methodologies varied significantly although the traditional methods were dominant and that the choice of methodology was largely dependent on nature and type of project among other reasons, a view also shared in [2] by Alamdy and Osman. Significant to this is that the methodologies of choice tend towards shorter processes aligning with the revelations on RE tools.

5.1.3 Software testing and practice. About 30 percent of the respondents from the SSCs not having a testing strategy is bad enough, and worst still, the strategies seem to be underutilized by the companies. In most cases, such strategies refer to utilizing internal documents. If internal documents and methods are not used, then the other models and methods cannot be taken up. Our findings on software testing closely relate to the study of [14], whereby the implementation of TMMI is not observed in Canada. This is also similar to the findings of [15]. This is perhaps because of the change in experience of the software practice in general and because changing development approaches would ordinarily require a change in test practice which unfortunately is given less priority; this view is highlighted in [20].

The interesting finding is how the ISO/IEC 29119 is most utilized compared to the others, although researchers in [19] state that it has laborious processes and no mechanism of adoption for companies. They also state that software testing is significantly ignored as we equally observe from the usage of the tools, be it internal or external. Additionally, the findings of Matalonga et al. [22] assert that ISO/IEC 29119 is intended to cover extensive testing approaches, and its application is convenient to traditional testing techniques; however, the SSCs could not have explored using this. This same view is shared by Eira et al. [12], who also shows how beneficial the standard can be to SSCs.

The low usage of inspections tools draws the attention of the researchers to the stages in the inspections process, which include planning, overview meeting, preparation, inspection meeting, rework, and follow-up, which perhaps becomes problematic given that the number of practitioners may not accommodate these tasks. A similar view is fronted in [33], associating this complexity to the characteristics of the companies. This probably makes inspection a mission impossible. It may mean developing more straightforward approaches for very small groups for adequate verification and validation. Keen interests need to be taken on studying the characteristics of the SSCs to inform the processes of choosing what tool could be ideal for what company a view in other studies [35][38][41].

5.2 Limitations and potential threats to validity

This sub-section discusses the threats to the validity this study was exposed to and the steps taken to minimize them. This study considered four types of threats to validity: internal validity, external, construct, and lastly, conclusion validity which is adopted from [43].

5.2.1 Internal validity. A major potential threat to the internal validity of our study lies in the selection of the sample population. The study considered only 115 respondents from 115 software companies; this may not represent the SSCs in the sample space. Due to limitations beyond our control, it was not feasible for us to reach out to only a sizable number from the three countries, and we continue to collect data from other countries that we shall use to triangulate at the end of the study to validate our findings.

Probabilistic sampling was not possible, and we used purposive sampling, where respondents are selected based on the set criterion discussed in section 3.1. We worked through established societies in these countries; however, the COVID-19 restrictions made it difficult as initially planned we then adjusted accordingly.

5.2.2 Construct validity. A threat to construct validity refers to whether what we measured is representative of what happens in the industry regarding software process tools. Counting the answers to each question designed to answer the questions as seen in [36], shows that results based on counting data and statistical inferences can reflect respondents' perceptions about the tools used in software development practice SSCs.

To further minimize this threat, we declared the study's intention to all participants as a survey to study the software engineering practices in companies. We also informed the participants that identifying information would not be used, and all participants will remain anonymous.

5.2.3 Conclusion validity. The threat to conclusion validity lies in the reliability of measures that depend on poor question wording and survey layout [43]. The authors reviewed the survey questions and subjected the questionnaire to a review of 5 other experts before subjecting the questionnaire to a pilot study. The software practitioners responded to the pilot study questions and provided feedback regarding the quality of our survey questions.

Another threat to conclusion validity was related to whether the conclusions we drew were based on rigorous and repeatable treatment [43] that we have done by an extensive description of the method. We also explicitly stated that our results would be based on a dataset of mostly SSCs, and others to contrast how much the companies may differ from those pivoting from SSCs. This was considered when interpreting the results and conclusions.

5.2.4 External validity. External validity deals with the limitations of generalizing the research results beyond the scope of the participants in our study. The extent to which the results are generalized to other people reflects its external validity; Although, this study has been triangulated with multiple data sites to reflect the more significant part of the community.

The use of 115 data points is a small sample, and we attempted to use purposive sampling to give room for qualitative generalisability. As pointed out by Amir and Ralph, purposive sampling is typical in software engineering to facilitate qualitative generalisability [3].

6 CONCLUSIONS AND FUTURE WORK

The software engineering community has, over the years, taken note of software process tools that are not adoptable by the SSCs and the big number of tools that remain unused in practice. This paper presents a survey in which we view software practice and

the usage of process tools in SSC as being minimal and the size of the companies is no absolute reason for how process tools are used, although we have observed that size has an effect on the tools used as also cited in[32][38], the lengthy processes of the tools seem to play a significant part in deciding on the choice of tool orchestrated with the urge for shortcuts and ad-hoc tendencies.

We also note that process areas like software testing tend to remain ignored, as noted in [15], because of the rapid changes in development processes, and this is also expected to transform the software testing approaches, yet volatility continues to overwhelm the practitioners in the SSCs. It is, however, essential to acknowledge the challenges of attempting to adopt a tool that is intently designed for larger companies and does not fit in the context of the SSCs, a view shared by other researchers in[1], [33] [40]. In this work, we present numerical summaries of the characteristics of the respondents and what tools are used by the practitioners during software development in SSCs.

In our ultimate goal of developing an adaptability framework to guide designers of software tools and help the software companies make correct decisions while choosing tools that fit their context, we also intend tools identified in this study to develop a deeper understanding of the characteristics of the SSCs to develop a classification taxonomy. This study gives us good insights to guide our next steps in which we then ask the why questions to give us a firm footing in understanding the software practice and the limited utilization of tools while developing software in SSCs. This study will also be validated with data that is being collected from two European countries whose data collection process is still ongoing.

ACKNOWLEDGMENTS

The authors would like to acknowledge the help of Victoria, Sharon, Pulafela, Joseph and Jonathan, who made the follow up calls to the respondents and Maria, Annika, Bilal, Shola, and Saltan for reviewing the survey protocol. We also appreciate the support of Denis, Kwaku, Joseph, Babatunde Nicole, Harriet and Daniel for their help during the survey piloting phase, and all the practitioners from the industry who voluntarily participated in the survey and helped us in the publicity of our survey.

REFERENCES

[1] Abrahamsson, P. *et al.* 2017. Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439.* (2017).
[2] Alamdy, S. and Osman, R. 2017. Software industry practice in Africa: case study Sudan. *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)* (2017), 743–748.
[3] Amir, B. and Ralph, P. 2018. There is no random sampling in software engineering research. *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings* (2018), 344–345.
[4] Andrade, C. 2021. The inconvenient truth about convenience and purposive samples. *Indian Journal of Psychological Medicine.* 43, 1 (2021), 86–88.
[5] Balikuddembe, J.K. and Tuape, M. 2017. An Ambiguity Minimization Technique during Requirements Elicitation Phase. *2017 International Conference on Computational Science and Computational Intelligence (CSCI)* (2017), 945–950.
[6] Baltes, S. and Diehl, S. 2018. Towards a Theory of Software Development Expertise. Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (New York, NY, USA, 2018), 187–200.
[7] Baltes, S. and Ralph, P. 2020. Sampling in software engineering research: A critical review and guidelines. *arXiv preprint arXiv:2002.07764.* (2020).
[8] Barr, M. and Parkinson, J. 2019. Developing a work-based software engineering degree in collaboration with industry. *Proceedings of the 1st UK & Ireland Computing Education Research Conference* (2019), 1–7.
[9] Cico, O. 2020. Towards transferring lean software startup practices in software engineering education. *{ESEC/FSE} '20: 28th {ACM} Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020* (2020), 1686–1689.
[10] Costagliola, G. *et al.* 2004. A framework for modeling and implementing visual notations with applications to software engineering. *ACM Transactions on Software Engineering and Methodology (TOSEM).* 13, 4 (2004), 431–487.
[11] Cronbach, L.J. 1951. Coefficient alpha and the internal structure of tests. *psychometrika.* 16, 3 (1951), 297–334.
[12] Eira, P. *et al.* 2018. Tailoring ISO/IEC/IEEE 29119-3 standard for small and medium-sized enterprises. *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (2018), 380–389.
[13] Garousi, V. *et al.* 2016. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. *Information and Software Technology.* 79, (2016), 106–127.
[14] Garousi, V. *et al.* 2017. What industry wants from academia in software testing? Hearing practitioners' opinions. *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering* (2017), 65–69.
[15] Garousi, V. and Zhi, J. 2013. A survey of software testing practices in Canada. *JOURNAL OF SYSTEMS AND SOFTWARE.* 86, 5 (May 2013), 1354–1376. DOI:https://doi.org/10.1016/j.jss.2012.12.051.
[16] Hidellaarachchi, D. *et al.* 2021. The Influence of Human Aspects on Requirements Engineering: Software Practitioners Perspective. *arXiv preprint arXiv:2109.07868.* (2021).
[17] Javed, T. *et al.* 2004. A study to investigate the impact of requirements instability on software defects. *ACM SIGSOFT Software Engineering Notes.* 29, 3 (2004), 1–7.
[18] Kajko-Mattsson, M. and Bosu, M. 2006. Eliciting an Enhancive Maintenance Model in Three Organisations in Ghana. *5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse (ICIS-COMSAR'06)* (2006), 244–251.
[19] Kasurinen, J. *et al.* 2011. A self-assessment framework for finding improvement objectives with ISO/IEC 29119 test standard. *European Conference on Software Process Improvement* (2011), 25–36.
[20] Kasurinen, J. *et al.* 2011. How Test Organizations Adopt New Testing Practices and Methods? *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops* (2011), 553–558.
[21] Klünder, J. *et al.* 2019. Catching up with method and process practice: An industry-informed baseline for researchers. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2019), 255–264.
[22] Matalonga, S. *et al.* 2015. Matching context aware software testing design techniques to ISO/IEC/IEEE 29119. *International Conference on Software Process Improvement and Capability Determination* (2015), 33–44.
[23] McLeod, L. and MacDonell, S.G. 2011. Factors that affect software systems development project outcomes: A survey of research. *ACM Computing Surveys (CSUR).* 43, 4 (2011), 1–56.
[24] Mohagheghi, P. and Jørgensen, M. 2017. What contributes to the success of IT projects? Success factors, challenges and lessons learned from an empirical study of software projects in the Norwegian public sector. *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (2017), 371–373.
[25] Mushashu, E.T. and Mtebe, J.S. 2019. Investigating Software Development Methodologies and Practices in Software Industry in Tanzania. *2019 IST-Africa Week Conference (IST-Africa)* (2019), 1–11.
[26] Nakatumba-Nabende, J. *et al.* 2017. Hybrid Software and Systems Development in Practice: Perspectives from Sweden and Uganda. *Product-Focused Software Process Improvement* (Cham, 2017), 413–419.
[27] Nandhakumar, J. and Avison, D.E. 1999. The fiction of methodological development: a field study of information systems development. *Information technology & people.* (1999).
[28] Nelulu, J. and Mufeti, T.K. 2021. An Exploratory Study of the Development Practices Used by Software Entrepreneurs in Namibia BT - Resilience, Entrepreneurship and ICT: Latest Research from Germany, South Africa, Mozambique and Namibia. J. Halberstadt *et al.*, eds. Springer International Publishing. 79–93.
[29] O'Connor, R. V. and Laporte, C.Y. 2017. The evolution of the ISO/IEC 29110 set of standards and guides. *International Journal of Information Technologies and Systems Approach.* 10, 1 (2017), 1–21. DOI:https://doi.org/10.4018/IJITSA.2017010101.
[30] Ralph, P. *et al.* 2020. ACM SIGSOFT empirical standards. (2020).
[31] Ralph, P. 2011. Introducing an empirical model of design. *The 6th Mediterranean Conference on Information Systems* (2011).
[32] Ribaud, V. *et al.* 2010. Software Engineering Support Activities for Very Small Entities. Systems, Software and Services Process Improvement - 17th European Conference, EuroSPI 2010, Grenoble, France, September 1-3, 2010. Proceedings (2010), 165–176.
[33] Rivas, L. *et al.* 2008. Towards a Selection Model for Software Engineering Tools in Small and Medium Enterprises (SMEs). *Proceedings of the Third International Conference on Software Engineering Advances, {ICSEA} 2008, October 26-31, 2008, Sliema, Malta* (2008), 264–269.

[34] Salo, R. 2014. A guideline for requirements management in GitHub with lean approach.
[35] Sánchez-Gordón, M.-L. and O'Connor, R. V 2016. Understanding the gap between software process practices and actual practice in very small companies. *Softw. Qual. J.* 24, 3 (2016), 549–570. DOI:https://doi.org/10.1007/s11219-015-9282-6.
[36] Stankovic, D. *et al.* 2013. A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software.* 86, 6 (2013), 1663–1678.
[37] Tripathi, N. *et al.* 2016. Exploring Processes in Small Software Companies: A Systematic Review BT - Software Process Improvement and Capability Determination. (Cham, 2016), 150–165.
[38] Tuape, M. *et al.* 2020. Does Context Matter? Assessing the Current State of Quality Practice During Software Development in Small Software Companies. *Proceedings of the Future Technologies Conference* (2020), 341–356.
[39] Tuape, M. *et al.* 2021. Software Engineering in Small Software Companies: Consolidating and Integrating Empirical Literature into a Process Tool Adoption Framework. *IEEE Access.* (2021).
[40] Tuape, M. and Ayalew, Y. 2019. A roadmap for a comparison framework for an adaptable software process improvement framework in small software companies. *Annals of Computer Science and Information Systems.* 20, (2019), 133–141.
[41] Tuape, M. and Ayalew, Y. 2019. Factors Affecting Development Process in Small Software Companies. *Proceedings - 2019 IEEE/ACM Symposium on Software Engineering in Africa, SEiA 2019.* (2019), 16–23. DOI:https://doi.org/10.1109/SEiA.2019.00011.
[42] von Wangenheim, C.G. *et al.* 2006. Helping Small Companies Assess Software Processes. *{IEEE} Softw.* 23, 1 (2006), 91–98. DOI:https://doi.org/10.1109/MS.2006.13.
[43] Wohlin, C. *et al.* 2012. *Experimentation in software engineering.* Springer Science & Business Media.