Organizational governance: Resolving insufficient practice and quality expectation in Small Software Companies.

MICHEAL, TUAPE*
Department of software engineering, Lappeenranta-Lahti University of Technology, Finland
micheal.tuape@lut.fi

PETRUS, T, Iiyambo
Dept of Computing, Maths and Statistical Sciences University of Namibia, Windhoek, Namibia
piiyambo@unam.na

JUSSI, Kasurinen
Department of software engineering, Lappeenranta-Lahti University of Technology, Finland
Jussi.kasurinen@lut.fi

ABSTRACT

The quality of software products is among the most prevalent challenges threatening the software development primarily in small software companies (SSCs). These challenges are associated with insufficient practices affecting the production of software and the development processes. This paper explores the role of governance in streamlining software processes and practices to produce better quality software products. In a cross-sectional survey (n = 127), we reached out to software practitioners working in SSCs from four countries. We examined how SSCs engage in oversight and accountability and how SSCs perform management roles and activities, such as controlling, directing, and guiding in the process of developing software. Our findings indicate that although the SSCs minimally embrace governance practices, the smaller companies have a more challenging task embracing governance practices from the complexities arising out of these companies' structures. This study highlights the aspects of governance that need attention in the smaller category of SSCs. It proposes an organizational governance model to facilitate the SSCs in developing governance strategies to take advantage of the benefits of governance during software development.

CCS CONCEPTS

• **Software and its engineering**; • **Software creation and management**; • **Software development process management**;

KEYWORDS

Software Organizational governance, Software processes, Small Software Companies

ACM Reference Format:
MICHEAL, TUAPE, PETRUS, T, Iiyambo, and JUSSI, Kasurinen. 2022. Organizational governance: Resolving insufficient practice and quality expectation in Small Software Companies.. In *2022 The 3rd European Symposium on Software Engineering (ESSE 2022), October 27–29, 2022, Rome, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3571697.3571700

*Place the footnote text for the author (if applicable) here.

ESSE 2022, October 27–29, 2022, Rome, Italy

ACM ISBN 978-1-4503-9730-8/22/10.
https://doi.org/10.1145/3571697.3571700

1 INTRODUCTION

Software plays a central role in the growth and advancement of humanity in today's world, where technology has become part and parcel of every aspect of society [1]. The small software companies (SSCs) are at the center of the evolution of the software industry to the extent that the industry has experienced significant growth of the SSCs and a noticeable dominance of the entire industry [2], [3]. The SSCs represent almost 90% of the software companies in industry and are responsible for about 80% of the software products produced on the market [4]. The SSCs contribute significantly to the growth of the software industry and other businesses, leading to increased employment and substantial economic development in most economies of the world [5].

Although the influence of software continues to get more entrenched in our livelihood, the industry is grappling with quality challenges, including the high failure rate of software under construction, researchers [1], [3] suggests that over 70% of software under construction fails to meet the expected quality. Tamburri et al. [6] put software failure during construction as one of the key and least understood challenges of the software industry and practice. Software failure during development is considered higher in SSCs [7], [8], and the causes of this high failure are due to insufficient processes and practice in building software [8]. Researchers indicate that software developer productivity is among the critical challenges of software companies, caused by a lack of commitment, motivation [9], [10], and intentional neglect of quality practices by the developers [11]. The challenges associated with software productivity are intrinsically associated with software quality and cost [9]. A typical productivity challenge is seen through source code defects that translate into time and cost consuming aspects of software development [12]. The Cambridge Judge Business School estimates that over 600 million hours are spent on debugging code annually, translating to 61 billion United States dollars in North America alone. Nevertheless, these such errors are caused by a human error related to productivity, which can be mitigated by paying attention to the non-technical factors like organizational governance.

The bad governance practice in software organizations, pointed out by Perscheid et al. [12] and Juiz and Toomey [13], is what exposes companies to deterioration of performance, breeding instability in processes, and reduced developer productivity. These challenges are synonymous with SSCs and have compromised the quality of software produced by the SSCs, as noted in Tuape et al. [14]. The absence of governance exposes organizations to risks hindering growth, and destroying reputation and trust, among the

software team. Although most SSCs have few employees and primarily focus on creating software that works, the organizational structure in most cases disadvantages the SSCs. It is complex to implement effective governance under such circumstances, as cited by [1], [15]. Governance entails providing leadership and management. Furthermore, the attributes of governance like oversight ensures the effective application of procedures and processes to attain desired results [13]. In addition, it ensures that the applied methods and techniques provide the intended value for the customer [16].

We used the activities that enhance the leadership and management of SSCs, to answer the question of governance in practice concerning the respective size categories of SSCs. We sent out survey questions to software practitioners (n=127) in four countries (Finland, Ghana, Namibia, and Tanzania). This study answers two research questions: (1) How do the SSCs engage in oversight and accountability while developing software products in the respective size categories? (2) How SSCs perform management roles and activities, such as controlling, directing, and guiding in the respective size categories? The rational of these research questions is from the understanding that leadership and management are aspects of governance responsible for assembling, organizing, and integrating the necessary resources to execute tasks and direct group efforts to achieve the desired goals. The limited usage of these threatens the possibility of attaining the required culture and responsible practices for producing quality products within the respective size categories of SSCs.

This study is important for research and practice in SSCs because we have not come across any study specific to SSCs discussing governance. Moreover, most of the challenges in practice are related to governance. Our work is expected to make several theoretical and practical contributions. First, we advance the research on organizational governance in SSCs to indicate that the lack of a structured approach to leadership and management roles in the software processes seriously impacts the processes and quality of software produced by the SSCs. Secondly, we propose an organizational governance model that can be useful to structure governance in the SSCs. This would prepare the SSCs to take up the governance and management roles even when the staffing is at minimal. The practical contribution to software practice is that the model will enable SSCs to deal with the structural complexities in practice by providing a framework for developing a governance strategy to support the leadership and guide the development team in developing goals and expectations, policies, rules, best practices, and metrics based on feedback.

The rest of the paper is structured as follows: Section 2 presents the related literature, and Section 3 presents the methodology. We present the survey results in Section 4, discuss the findings, and propose an organizational governance model for the SSCs in Section 5.

2 RELATED WORKS

Although most of the challenges cited in the literature associated with software engineering can be related to governance, the attempt to specifically resolve governance-related challenges has received minimal attention in software practice. We summarize evidence from the literature that discusses challenges associated with software engineering and suggest the relationship between these challenges and software governance [2], [8], [17]. Secondly, we identify the few studies [16], [18], [19] that explicitly discuss governance issues in software engineering. Comparing our work with what has been done in literature, we emphasize two important aspects of the challenges of engineering processes and how governance can be used to streamline processes in SSCs.

Khokhar et al. [2] present an investigation in a Systematic Literature Review (SLR) protocol in which they studied software process improvement (SPI) factors for SSCs. Their study identified critical success factors (CSF) that positively impact the successful implementation of SPI. Their findings indicate 7 CSFs: (1) leadership involvement, (2) employee participation, (3) management commitment, (4) training, (5) business orientation, (6) organizational process focus, and (7) lack of quality conscious people (management skills). All the 7 CSF turn out to be non-technical factors and significantly related to governance in SSCs. They also identify critical barriers (CBs) as crucial for SPI initiation in SSCs and list three factors, (1) SPI understanding, (2) organizational structure and (3) project management. Their findings are in tandem with the findings of Dyba [17], who lists factors that highlight governance or mostly non-technical factors. Similarly, Tuape and Ayalew [8], in another SLR protocol, study SSCs and discuss the factors that affect the software processes in SSCs and mention organizational governance as a significant factor. Just as Khokhar et al. [2] highlight up to 75 percent of non-technical factors that relate to governance in the SSCs.

Juiz and Colomo-Palacios [18] proposed an enhanced software development governance model adopted from the ISO/IEC 38500, which is an international standard published jointly by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) for Corporate Information Technology (IT) Governance. This standard acknowledges the importance of governance. Although it takes care of preparing and implementing plans and policies, it also emphasizes the importance of monitoring conformance to policies and performance against the plans in an organization about IT in general. The Juiz and Colomo-Palacios model proposes new considerations beyond those proposed by Chulani et al. [16]. The researchers Nguyen et al. [19] present a generic framework to map IT governance principles to the GI-Tropos software processes, which adopts the COBIT 5, a famous IT governance and control framework that is formalized by the IT Governance Institute (ITGI). Generally, COBIT offers a reference model of 37 IT processes found in an organization. This framework provides a process reference model which defines governance and management processes in detail and operates with a full view of helping software organizations' business processes meet their strategic requirements.

The frameworks and models proposed by Chulani et al. [16], Juiz and Colomo-Palacios [18], and Nguyen et al. [19] herein address governance concerns in software engineering generally; however, the specific nature and uniqueness of the SSCs pose unique challenges as pointed out by Khokhar et al. [2], Tuape and Ayalew [8] and Dyba [17] in the respective studies may require unique solutions specific to SSCs.

3 METHODOLOGY

We used a quantitative cross-sectional study design to help answer the research questions regarding the implementation of governance activities during software practice in SSCs. The study uses closed-ended questions answered with a type 5-Likert scale. In this section, we present and discuss the population and sample of the study, we describe the participants, we discuss the design of the survey, and lastly, we discuss the method of analysis of the data used in the study.

3.1 Population and Sample

For this study, we implemented a set of criteria for selecting the appropriate sample for the study. We identified developer groups from four countries: Finland (F), Ghana (G), Namibia (N) and Tanzania (T) for the study sites. Although the sites were not exclusive to SSCs, the sites included a mix of software companies, academic institutions interested in software practice, and other organizations supportive of SMEs in software development. For the target population, we focused on SSCs (1-50 employees) developing software for various markets in the four study sites.

The purpose of the sample was to ensure that the participants gave the researchers a varied experience on governance in software practices within the different categories of SSCs. The researchers chose the participants through purposive sampling based on three criteria: (1) SSCs developing software-intensive products (1-50 employees), (2) the company must have existed for at least five years, and (3) respondents must have at least five years of experience. Based on these criteria, we identified 390 companies that fit within the study's sample of interest. From the loose groups, we could not establish the ultimate number of SSCs that met the set criteria; however, to achieve the purpose of the sample, we further profiled the selected companies into three categories: 1-25, 26-30, and 31-50 employees. The category choice was to look out for the diverse characters described in Tuape et al. [14]. By categorizing and subdividing the companies into these categories, we intended to evaluate governance in software practices within the different categories with different viewpoints.

Software engineering studies typically use a purposeful (heterogeneous) sampling approach, as advocated for in Baltes and Ralf [20] and used by researchers [21], [22]. Since the researchers wanted as much insight as possible into the phenomenon, including the different size profiles of SSCs in the study, would meet the researcher's expectation to uncover unique and diverse experiences of the participants from which to develop a governance model.

The researchers sent out survey questionnaires to a total of 390 companies, 108(F), 103(G), 84(N) and 95(T). After reviewing 163 datasets, the researchers determined that some datasets did not fit the set criteria, so the researchers excluded them from the ultimate analysis. We carefully considered the three categories' profiles as a basis for diversity in governance experiences, although the numbers of companies in the two categories, 26-30 and 31-50, were significantly small. Nonetheless, the researchers considered 127 participants representing 29, 35, 28 and 35 participants and 22.8, 27.6, 22.0 and 27.6 percent, respectively.

We combined several well-proven techniques for improving the response rate of mailed questionnaires, including contacting the participants on the phone and requesting participation in the study before mailing the questionnaire link on the Webropol survey system. A pilot study of the survey questionnaire revealed that respondents needed about 15 to 20 minutes to complete it. The results in this study are within the limits of adequate statistical power and generalizability. In addition, this represents an effective response rate of 41.8 percent, slightly above the minimum 40 percent suggested in [21].

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 (products produced by the company, her gender, her level of education, and the country from which the data was collected).

Most companies are involved in producing more than one software-intensive product, predominantly software solutions and web applications. The developers and software engineers are the dominant roles representing 66.9 and 47.2%, respectively. Considering the gender of the participants, 74% were male, 25.2% were female, and for the level of education, the participants with bachelor's degrees were dominant, followed by those with master's degrees representing 62.2 and 29.1%, respectively. The other aspect which is significant about the characteristics of the participants is that we categorized the companies by the number of professionals employed in the company, and of the 127 participants, representing 74.8, 13.4 and 11.8% from the companies with 1-25, 26-30 and 31-50 employees.

3.3 Survey Questions

The online survey questionnaire was designed to investigate software practice. The section of the questionnaire used in this part of the study was specifically to take care of the elements of organizational governance in the practice of the SSCs and the extent to which the respective categories of SSCs use governance in the development processes to produce quality software. 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 [22]. 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. There is evidence in the literature showing that [23] researchers and industrial practitioners often use different terminologies, and, when conducting industry research, there is a need for consensus on terminology. Early feedback from the practitioners helped ensure the proper context for the survey and the familiarity of the terminology used the questionnaire for industrial respondents. This was used to develop the second set of survey questions that were subjected to review by five other experts with over 5 years of experience in software engineering surveys. After improving the questions based on the feedback from the industry practitioners in the pilot study, the instrument ended up having 28 questions. A set of questions was dedicated to the profiles and demographics of the respondents. The other questions probed the software practice while focusing on organizational governance and quality practice.

Consent of every respondent to take part in the survey was sought, and each of the respondents was informed that the data from the survey would be used for research purposes.

3.4 Data Analysis Techniques

Cross-tabulation sometimes referred to as cross-tab or contingency tables, is used to analyze the relationship between categorical variables. This type of data involves categories of variables that are mutually exclusive from each other. Nominal and ordinal data were collected in numbers, but numbers have no quantifiable value unless they mean something also used in [24]. A Pearson Chi-Square test of independence was used to determine if there exists a significant relationship between the response variable (the number of employees in the respective company sizes) and the proposed explanatory variables of the study (the different attribute of governance). All analyses were conducted using SPSS version 26.

4 RESULTS

This section presents the investigation results of the two research questions in two subsections. Subsection 4.1 presents the extent to which SSCs embrace oversight and accountability activities in software practice. Subsection 4.2 presents the findings from the investigation of how management roles through activities of control, directing and guiding are implemented during software practice in the SSCs. Based on the company's number of employees, the results are discussed in three categories of company sizes.

4.1 Oversight and Accountability in Software Development Practice

4.1.1 Oversight. To investigate how SSCs have embraced oversight practices, we asked the respondents how frequently they used feedback in informing their oversight function to evaluate the processes for expected results and value for money of both the software product and process. The results show the cross-tabulation results between the number of employees in the company and the frequent use of feedback as a mechanism of oversight. Of 127 respondents, 43.3 % confirmed that they always get feedback as a mechanism of oversight, 29.1 % indicated that they often received feedback, and 18.9 % responded with a neutral answer (sometimes). In contrast, 8.7 % responded with rarely, and none embraced feedback for oversight purposes. The results also show that there are no significant differences in the proportion of SSCs that incorporate oversight and accountability activities in practice across the three categories of company sizes (1-25, 26-30, and 31-50) (Pearson Chi-Square value = 3.133, degrees of freedom = 6, p-value = 0.792).

4.1.2 Accountability. Accountability is about reviewing the results of the processes in retrospection; it is from this that the lists of lessons learnt during software development projects are drawn to establish and review metrics and best practices in an organization. The results show that overall, 54.3% of the respondents answered this question to the affirmative (often and always), 20.5% of the respondents answered neutrally (sometimes), and 25.3% of the respondents answered not to have or rarely used lessons learnt in their organizations. Comparing the use of lessons learned across the number of employees in a company, there seems to be insufficient evidence to conclude that there is a significant relationship between the number of software professionals and lessons learned in a project (Pearson Chi-Square value = 14.248, degrees of freedom = 8, p-value = 0.076).

4.2 Control, Direct and Guide Processes During Development

The respondents were asked three questions to investigate the implementation of controlling, directing, and guiding the software development processes. The subsequent subsections present the results.

4.2.1 Control. In investigating the extent to which management exerted control during software development, we examined the arrangements of formal meetings to discuss lessons learnt, 35.4% of the respondents responded to having either always or often had formal meetings with management. In comparison, 37% responded with a neutral answer, and 27.6% indicated that the management either never or rarely organized formal meetings to adopt lessons learnt as a control mechanism within the organization. However, there is insufficient evidence to suggest a significant association between the extent to which management organized formal meetings to adopt lessons learnt and the number of software professionals employed in the company (Pearson Chi-Square =13.704, degrees of freedom = 8, p-value =0.090).

4.2.2 Directing. Overall, 39.4 percent of the respondents reaffirmed that the use of meetings to make changes symbolizes a structured approach to directing activities in the organization, 37% indicated that is often the case, while 23% responded neutrally. Moreover, 35.8% of those employed by organizations with 1-25 software professionals agreed that management conducted meetings to make changes as a method of directing activities. Similarly, for SSCs employing 26-30 and 31-50 software professionals, 41.2% and 60% of the respondents reaffirmed that management conducted meetings to make changes to direct activities in the organizations, respectively. The neutral (sometimes) respondents represent 28.4%, 5.9%, and 13.3% of the SSCs employing 1-25, 26-30 and 31-50 software professionals, respectively.

4.2.3 Guiding Processes. Our findings show that overall, 57.5%, 26.8%, and 15.8% of the respondents affirmed having either often or always neutral and rarely or never used contingency planning to guide software development. The proportions of respondents who either always or often use contingency planning to guide software development in SSCs are 54.8%, 52.9% and 80% for SSCs employing 1-25, 26-30 and 31-50 software professionals, respectively.

4.3 DISCUSSION

In this section, we analyze, explore the meaning, and discuss the significance of our findings while answering the research questions and proposing remedies to some of the challenges highlighted in the data presented in the previous section. The overall aim of this study was to understand how organizational governance is conducted in the respective company categories of SSCs. To fulfil this aim, we address the research questions and attempt to relate our findings with earlier studies that have highlighted software

development challenges related to governance specific to SSCs. This section has two parts explaining our findings regarding research questions one and two, and the second part presents and discusses the significance of our proposed organizational governance model in solving some of the challenges associated with organizational governance highlighted in this study.

4.4 Discussion of the Research Questions

4.4.1 Oversight and Accountability. Overall, results show that SSCs relatively embrace accountability and oversight with 54.3% and 43.3%, respectively. This finding means that leadership functions are moderately prevalent in the SSCs; this contrasts with the findings of Nørbjerg et al. [25], who, in their case study, point out that lack of oversight, among other factors, has an effect of reducing the capability of SSCs to adapt project management processes and practices in response to changes at the company level. The difference in result could be because the authors conducted a single case study on a software company which is different from our study with multiple SSCs. The contrast, however, indicates that the SSCs in the category that is closer to the one in the case study portray significant usage of oversight. The likely explanation for this contrast is perhaps because Nørbjerg et al. studied only one company. It is generally known in governance in organizations that both accountability and oversight must be used concurrently to mutually support each other to attain the benefits they offer an organization. This means that oversight and accountability are expected to complement each other. The former resolves challenges like skill gaps, volatility in requirements, violations of best practices and overall compromise in quality that impair software quality and make software hard to maintain affecting software evolution. While the latter helps identify requirements mismanagement, poor requirements documentation, insufficient planning and all the associated challenges. The implication of volatility in requirements is observed through project delays, cost overruns and high defect density on software products, while violations in practice and processes increase the chances of failure and vulnerabilities in software products, generally compromising the overall quality of software.

The revelation that the categories of software companies employing 0-12 and 26 to 30 suggest less usage of accountability and oversight practices and it is perhaps related to instability witnessed in the practice of the SSCs in these categories, as highlighted by Tuape et al. [14] and Chulani et al. [16]. The authors highlight the confusion in understanding how software development practices related to governance and what could constitute effective organizational governance in SSCs, especially in the smaller categories. This is seen with the isolated result of SSCs with 31 to 50 employees that present relatively higher incidences in both accountability and oversight. Indeed, this is in tandem with other findings measuring quality and maturity in processes in the same categories that seem to have more structured mechanisms to ensure accountability and oversight in the software development practice.

Despite the lack of statistically significant effect on the relationship of oversight compared to the categories classified by several software development professionals employed in the companies, nonetheless, there is a pronounced trend illustrated by similar results indicated in literature where Muñoz et al. [26], cite poor quality software products by the Very Small Entities (VSE) due to lack skill and failure to follow best practices. This is likely to be because of the skewness of the data, given that up to 87 percent of the respondents fall in the two lower categories of companies employing 30 persons and below. The other likelihood is that the sample size was small, and a remedy to this would be further studies with larger sample size. On the other hand, accountability has a statistically significant effect on the categories. This is also seen in the literature indicating that the quality of products and processes in these categories remain a challenge, as cited in studies by Melegati et al. [27], and Tuape et al. [15]. Boehm and Turner [28] add that lack of accountability could lead to low morale among the individual developers or the software team. The authors further argue that unclear priorities across the team, minimal employee engagement, unachievable team and individual goals, low levels of trust and high turnover could pose a challenge as far as implementing measurement and control.

A software team cannot be accountable if they do not know why they are taking accountability. A remedy is that an explicit set of goals and expectations is a good starting point for software companies to ensure accountability. Goals must be set for the individuals and the development team; the goals must be clear and measurable so that all involved know what they are trying to achieve. It is therefore imperative to establish the usage of rules, metrics and best practices as a mechanism implemented through accountability and oversight using transformational leadership to foster relationships with the developers to guide and nurture their skills as alluded to by Eseryel and Eseryel [29], proposed in the model, in Figure 1.

4.4.2 Control, Directing and Guiding. The results indicate that the overall implementation of management roles and activities of control, directing and guiding during software development practice in the SSCs is moderate. Control, directing, and guiding is at 35.4%, 39.4%, and 57.5%, respectively, of the respondents across all categories of the number of software professionals employed in the organizations. Like the leadership roles, the management roles indicate high incidences of practice in all the aspects considered except for the control. This means that SSCs are not likely to hit goals because of a lack of effective control. Some evidence of this is suggested by Khokhar et al. [2]. They cite limitations in software practices in SSCs due to lack of control and insufficient leadership involvement, which affects employee commitment. Control allows SSCs to check errors, implement corrective actions, minimize deviations from methods, and keep software projects on track. This means that SSCs would not enjoy these benefits without control. Ultimately having less control means the companies cannot lead to increased productivity, as highlighted by Paiva et al. [10].

The interpretation of these results considers that there is no statistical significance between the control and the categories by the number of employees. However, the findings can be seen as relevant based on the evidence from previous work that highlights challenges in software development practice synonymous with SSCs like leadership involvement, employee participation and lack of quality conscious people to be key impediments to quality and critical success factors highlighted by Khokhar et al. [2] and also emphasized Tuape and Ayalew [8] in another study.

As a remedy to the challenges in the findings, SSCs need to be assisted in mitigating the challenges of governance that affect

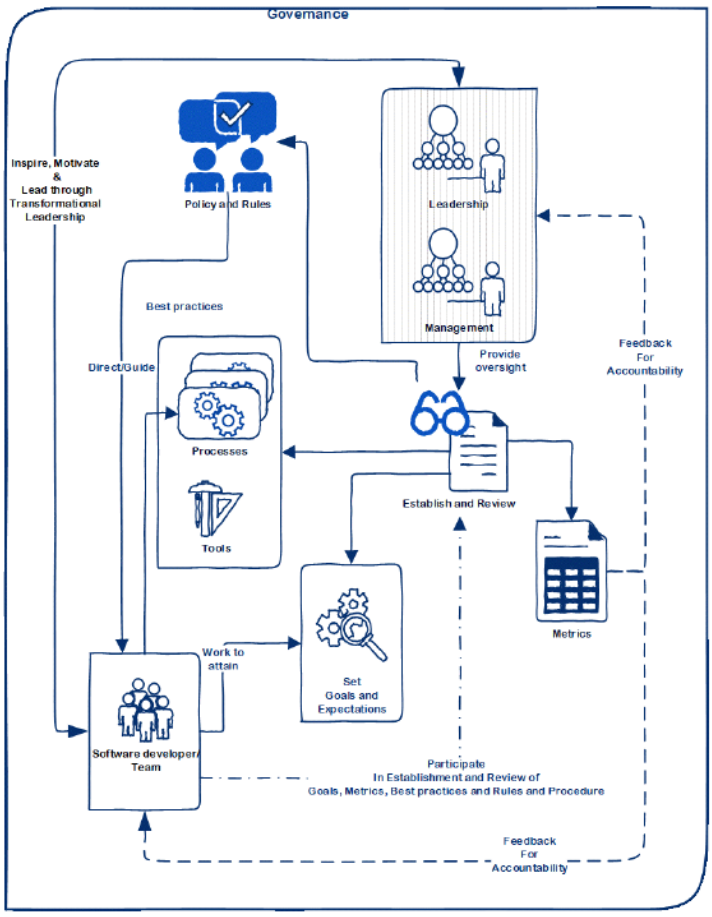

Figure 1: Proposed organizational governance model, it applies elements from the theory of transformational leadership and focuses on the leadership and management to inspire and drive

software practice and the quality of products by introducing mechanisms that are cognizant of the uniqueness of SSCs and can also facilitate them in adopting practices that foster control within the organizations, particularly during software practice.

4.5 Organizational Governance Model for Small Software Companies

Figure 1 represents the proposed governance model for SSCs; the model focuses on the leadership, management, and the software development team as critical aspects of organizational governance. It presents governance as a responsibility of executives to provide leadership, managers are responsible for organizing the routine running of the company, and the software development team is responsible for the execution of the tasks of developing quality software to the satisfaction of the customers. Unlike Juiz and Colomo-Palacios's [18] proposed software development governance model that focuses on leadership and management, our model proposes the application of transformational leadership while focusing on the leaders and managers to inspire positive change in the software team. The application of transformational leadership is to enable a push for the encouragement of a small team to realize overall success. By raising a team's morale and self-confidence, the team can align itself to an overall vision or common purpose.

The efforts are geared towards taking a struggling or stagnant team and completely transforming it into a productive and dynamic team that develops quality software to the satisfaction of the clients.

Hashmi et al. argue that transformational leadership is reciprocal, iterative and ideal for small teams [30]. It begins with working closely with the team to facilitate the identification of the team or individual's weaknesses to establish a clear pathway for improvement. The transformational leader works closely with the software development team to support, evaluate, and redirect the team's effort to achieve the overall goal. This offers additional positive effects, including nurturing company culture, increasing creativity, reducing developer turnover, and nurturing loyalty among the software development team. The model focuses on the 3 aspects, and each of the aspects takes up responsibility, as discussed below.

4.5.1 Leadership. First is idealized influence, where the leaders have the charisma to rally the software developers around a shared

vision. The availability of separate teams to take responsibility for the separate governance attributes. Separating leadership and management promotes accountability at all levels. The model envisages two scenarios to take care of the challenge highlighted in literature in which Khokhar et al. [2] discuss organizational structure in SSCs and a limitation to process improvement and affecting overall software development; responsible for establishing and reviewing processes and tools used during software development, policy, and procedure to guide the processes, the goals and expectations that the developers/software development team must achieve.

Scenario 1: ad hoc leadership

This focuses on the extreme cases where there are few members within an organization and, in most cases, no clear distinction between executives and management. The situation may be different for some SSCs, especially those with less staff; for instance, SSCs with less than 10 employees may have no clear separation between managerial and executive roles. The non-existence of executives and managers in most cases become involved in software production due to resource constraints. In scenarios the roles for internal control should be assigned to a quasi-executive to provide oversight and ensure accountability when it is required. This would mean team members assume the leadership roles, which reduces conflict of interest and clarifies accountability. To achieve this, managers can execute their leadership role by inspiring, motivating and leading the team involved in software development. Thus, given that scenario 1, managers may double as executives in most cases, managers should endeavor to separate themselves from their managerial role when acting as part of the governing body or executives.

Scenario 2: lean leadership

In this case, the team may have more employees more than 10, which makes it possible to separate the executive, management, and the software development team. The number of persons involved in the leadership and management team may not be important, provided the roles are well taken care of. Just like in Scenario 1, the responsibility of executives is to set organizational goals and priorities, chart direction, and set limitations and boundaries, which all constitute an accountability framework. The clear separation of roles also provides a mechanism for good governance that focuses on stakeholder value by balancing performance and conformance.

4.5.2 Management. Managers need to support the development team through planning, organizing, directing, guiding, and controlling as a remedy to the management commitment addressed by Khokhar et al. [2]. By using the functions, managers work to increase the efficiency and effectiveness of their development team, the processes and tools, the projects, and the organization in general. Inspirational motivation is attained through the encouragement of the software development team to commit to this vision by raising team spirit, fostering community and a sense of purpose. Management also improves the team's effectiveness by enabling good communication and effective measurement and control while keeping an eye on the metrics, the set goals and expectations, the best practices, and constantly looking out for feedback to improve the team and the whole organization. It is also the responsibility of management to intellectually stimulate and encourage the development team to think out of the box and be innovative.

4.5.3 Software Development Team. The software development team should individually consider focusing on how each developer connects with the overall goal of the organization, sorting out employee participation pointed out by Khokhar et al. [2]. The individual developers/development team need to be involved in establishing and reviewing the policy and procedure to guide the processes in setting the goals and expectations, metrics, and best practices. Their participation leads to a better understanding of what is being done versus what is supposed to be done. Notably, one can then take responsibility for their actions. Mistakes can be easily visible for the induvial or team to acknowledge the mistakes made; this also empowers one to recognize that they have the power to fix that mistake, which is essential for inspiration and motivation within the organization. The model proposes feedback to the software development team as a basis for accountability for the development team and enables effective resource planning. Additionally, metrics, best practices and a mechanism of feedback must be in place, as illustrated in Figure 1. The efficiency of all these is dependent on the inputs of the developers.

5 CONCLUSIONS

Governance is pivotal to the team's success in saving time and ensuring better productivity. We studied how SSCs embrace governance in practices of the different categories of SSCs, and our findings reveal that the smaller companies take up less of these practices compared to the stable companies in the upper category. This explains the difficulty in process utilisation and the persisted quality issues on processes and product prevalent in both categories with less than 30 employees. The SSCs should accept that if they must mature and grow, then governance should be considered as a powerful instrument for success, without which they could face loss of opportunity and potential failure.

For the SSCs to benefit from the governance practices, we propose an organizational governance model that can be useful for the SSCs in developing company-specific governance strategies to complement the software development methodologies and processes in ensuring high productivity and quality software. The absence of governance structures should not deter attaining the benefits of the governance to the SSCs; however, what should be taken importantly is to enforce the practices and activities involved in governance to attain competitive advantage, as also suggested by Chulani et al. [16].

Our model offers an avenue for the SSCs to establish responsibility, authority, and channels of communication for effective interaction and empowerment of individuals and teams involved in software development within an organization. The model also establishes a basis to guide the utilization of process for development and reviewing metrics, goals, and expectations to streamline measurement and control mechanisms that will enable software developers, project managers and teams to perform respective roles and responsibilities with accountability.

Our next steps will be to subject the proposed model to industry experiments to evaluate the model's efficiency in the different contexts in which the SSCs develop software. We also intend to integrate the model into the adaptable framework for quality software development for SSCs.

ACKNOWLEDGMENTS

We sincerely thank all the participants who offered their valuable time to take part in the survey during the data collection. We also extend our gratitude to the experts who supported the development of the questionnaire.

REFERENCES

[1] M. Tuape, V. Hasheela-Mufeti, A. Kayanda, J. Porras, and J. Kasurinen, "Software Engineering in Small Software Companies: Consolidating and Integrating Empirical Literature into a Process Tool Adoption Framework," IEEE Access, 2021.

[2] M. N. Khokhar, K. Zeshan, and J. Aamir, "Literature review on the software process improvement factors in the small organizations," in 4th International Conference on New Trends in Information Science and Service Science, 2010, pp. 592–598.

[3] 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 - 16th International Conference, {SPICE} 2016, Dublin, Ireland, June 9-10, 2016, Proceedings, 2016, vol. 609, pp. 150–165, doi: 10.1007/978-3-319-38980-6_12.

[4] M. Tuape, V. Hasheela-Mufeti, A. Kayanda, and J. Kasurinen, "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, 2021, pp. 29–38, doi: 10.1145/3501774.3501779.

[5] N. Tripathi, E. Annanperä, M. Oivo, and K. Liukkunen, "Exploring Processes in Small Software Companies: A Systematic Review BT - Software Process Improvement and Capability Determination," 2016, pp. 150–165.

[6] D. A. Tamburri, F. Palomba, and R. Kazman, "Success and failure in software engineering: a followup systematic literature review," IEEE Trans. Eng. Manag., vol. 68, no. 2, pp. 599–611, 2020.

[7] 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, no. January 2019, pp. 255–274, 2018, doi: 10.1016/j.jss.2018.06.043.

[8] M. Tuape and Y. Ayalew, "Factors Affecting Development Process in Small Software Companies," Proc. - 2019 IEEE/ACM Symp. Softw. Eng. Africa, SEiA 2019, pp. 16–23, 2019, doi: 10.1109/SEiA.2019.00011.

[9] E. D. Canedo and G. A. Santos, "Factors Affecting Software Development Productivity: An Empirical Study," in Proceedings of the XXXIII Brazilian Symposium on Software Engineering, 2019, pp. 307–316, doi: 10.1145/3350768.3352491.

[10] E. Paiva, D. Barbosa, R. Lima, and A. Albuquerque, "Factors that Influence the Productivity of Software Developers in a Developer View BT - Innovations in Computing Sciences and Software Engineering," 2010, pp. 99–104.

[11] H. Ghanbari, T. Vartiainen, and M. Siponen, "Omission of Quality Software Development Practices: A Systematic Literature Review," ACM Comput. Surv., vol. 51, no. 2, Feb. 2018, doi: 10.1145/3177746.

[12] M. Perscheid, B. Siegmund, M. Taeumel, and R. Hirschfeld, "Studying the advancement in debugging practice of professional software developers," Softw. Qual. J., vol. 25, no. 1, pp. 83–110, 2017.

[13] C. Juiz and M. Toomey, "To govern IT, or not to govern IT?," Commun. ACM, vol. 58, no. 2, pp. 58–64, 2015.

[14] 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 Proceedings of the Future Technologies Conference, 2020, pp. 341–356.

[15] M. Tuape, V. Hasheela-Mufeti, P. Iiyambo, A. Kayanda, and J. Kasurinen, "Software Development Practice: How Organisation Dynamics Inhibit the Utilization of Process Tools in Small Software Companies," in 2021 10th International Conference on Software and Information Engineering (ICSIE), 2021, pp. 35–40, doi: 10.1145/3512716.3512722.

[16] S. Chulani, C. Williams, and A. Yaeli, "Software development governance and its concerns," in Proceedings of the 1st international workshop on Software development governance, 2008, pp. 3–6.

[17] T. Dybå, "Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context," in Proceedings of the 11th {ACM} {SIGSOFT} Symposium on Foundations of Software Engineering 2003 held jointly with 9th European Software Engineering Conference, {ESEC/FSE} 2003, Helsinki, Finland, September 1-5, 2003, 2003, pp. 148–157, doi: 10.1145/940071.940092.

[18] C. Juiz and R. Colomo-Palacios, "Extending Software Development Governance to meet IT Governance," in Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, 2020, pp. 295–298.

[19] V. H. A. Nguyen, M. Kolp, Y. Wautelet, and S. Heng, "Mapping IT Governance to Software Development Process: From COBIT 5 to GI-Tropos.," in ICEIS (2), 2018, pp. 665–672.

[20] S. Baltes and P. Ralph, "Sampling in software engineering research: A critical review and guidelines," arXiv Prepr. arXiv2002.07764, 2020.

[21] S. Baltes and S. Diehl, "Towards a Theory of Software Development Expertise," in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, pp. 187–200, doi: 10.1145/3236024.3236061.

[22] V. Garousi and J. Zhi, "A survey of software testing practices in Canada," J. Syst. Softw., vol. 86, no. 5, pp. 1354–1376, May 2013, doi: 10.1016/j.jss.2012.12.051.

[23] V. Garousi, K. Petersen, and B. Ozkan, "Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review," Inf. Softw. Technol., vol. 79, pp. 106–127, 2016.

[24] T. Javed, M. e Maqsood, and Q. S. Durrani, "A study to investigate the impact of requirements instability on software defects," ACM SIGSOFT Softw. Eng. Notes, vol. 29, no. 3, pp. 1–7, 2004.

[25] J. Nørbjerg, P. A. Nielsen, and J. S. Persson, "Dynamic Capabilities and Project Management in Small Software Firms," in 50th Hawaii International Conference on System Sciences, {HICSS} 2017, Hilton Waikoloa Village, Hawaii, USA, January 4-7, 2017, 2017, pp. 1–10, [Online]. Available: http://hdl.handle.net/10125/41817.

[26] M. Muñoz, A. Peña, J. Mejia, G. P. G. 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, 2020, doi: 10.1002/smr.2300.

[27] 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, 2019.

[28] B. Boehm and R. Turner, "Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods," in Proceedings. 26th International Conference on Software Engineering, 2004, pp. 718–719.

[29] U. Y. Eseryel and D. Eseryel, "Action-embedded transformational leadership in self-managing global information systems development teams," J. Strateg. Inf. Syst., vol. 22, no. 2, pp. 103–120, 2013, doi: https://doi.org/10.1016/j.jsis.2013.02.001.

[30] A. Hashmi, S. Ishak, and H. B. Hassan, "Role of team size as a contextual variable for the relationship of transformational leadership and teamwork quality," Asian J. Multidiscip. Stud., vol. 6, no. 5, pp. 76–81, 2018.