Refactoring Community Smells: An Empirical Study on the Software Practitioners of Bangladesh

1st Noshin Tahsin
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
bsse0914@iit.du.ac.bd

2nd Kazi Sakib
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
sakib@iit.du.ac.bd

***Abstract*—Community smells are organizational and social anti-patterns in the development community that need to be refactored. In the literature, studies on community smell refactoring are found from the very conceptual level. However, little is known about the practitioners' perceptions, refactoring readiness and the refactoring strategies adopted in local software communities. This paper bridge this gap by exploring these issues in the software industry of Bangladesh. A depth interview-based study was conducted on local software practitioners chosen through a convenience sample recruitment strategy. Interviews were transcribed and analyzed using Straussian Grounded Theory. We collected data on the four prominent smells according to literature and introduced a new measure called 'Refactoring Readiness' to calculate the community smell refactoring preparedness of a software development community. Analyzing the data, it is seen that 85% local practitioners perceive community smells as harmful but less than half take step to mitigate those smells. We identified the refactoring strategies (e.g. creating a structured communication plan, mentoring) currently adopted by them and found that the Refactoring Readiness of the software industry of Bangladesh is 0.63 on a scale of 0-1. This provides evidence that more work needs to be done for refactoring community smells from the local sub-optimal development communities.**

***Index Terms*—community smell, refactoring, empirical study**

I. Introduction

Community smells are the set of organizational and social anti-patterns in a development community [1] [2]. If reiterated over time, they lead to the emergence of unforeseen project costs connected to a sub-optimal software development community. The existence of community smells eventually hampers the overall production, operation, maintenance, and evolution of software [3] [10] [14]. To minimize these issues, it is necessary to follow some mitigation strategies.

Catolino et al. first referred to the community smell mitigation strategies as "refactoring" [6]. The field of community smell refactoring has not received much attention in the local context. It is not yet known whether software practitioners from various types of local development communities are aware of community smells. Additionally, it is unknown if they are prepared to implement strategies to effectively reduce these smells. Furthermore, there may exist differences in refactoring practices among various local communities.

This study is supported by the Fellowship from ICT Division, Government of Bangladesh – No.: 56.00.0000.052.33.002.22-42, date: 12.06.2022

Community smell refactoring is a less explored field in the literature. In a study on a single software company, Tamburri et al. [1] reported six mitigation decisions made by the management board of that company. Catolino et al. [6] conducted a survey based study to identify developers' perception of community smells and the strategies adopted by them to mitigate those smells. Sarmento et al. [13] replicated the same study on Brazilian software teams to identify and analyze localized refactoring practices. Similar studies on developers' perception and their refactoring practices need to be conducted in other regions.

We have conducted such an empirical investigation on the software community of Bangladesh. To conduct the study, an interview questionnaire was designed based on [6] and modified according to our goals. Pilot interviews were conducted on five software practitioners to refine the draft questionnaire. Using the refined questionnaire, we took interview of 21 software practitioners having considerable experience in both software development and management. The practitioners were chosen through a convenience sample recruitment strategy [15]. Interviews were transcribed and analyzed applying Straussian Grounded Theory [17] to find out the community smell refactoring strategies adopted by the practitioners. We introduced a new measure called Refactoring Readiness score (RR score) to calculate the preparedness of a software development community to effectively refactor community smells and calculated the RR score of the software industry of Bangladesh. Similar to [6] [7] [8], we chose the four most prominent smells for our study - Organizational Silo, Black Cloud, Lone Wolf and Radio Silence.

We show that, community smells are relevant in the Bangladeshi software industry. 68.75% of respondents experienced the four most prominent smells on average. For 80.9% respondents, the smells occurred frequently or occasionally. Exploring the local practitioners' perception of community smells, it is seen that, they perceive these smells as harmful but do not take sufficient mitigation measures. Moreover, we calculate that the RR score of the local software industry is 0.63 on a scale of 0-1, which means more work need to be done to make the industry refactoring ready. In addition, we identify the refactoring strategies currently adopted in the software industry of Bangladesh.

II. Literature Review

Recently, community smells have received attention from the research community, which is growing day by day. Tamburri et al. [1] first identified these anti-patterns through an industrial case-study and named those anti-patterns as "community smells." Numerous studies have been conducted to investigate the relationship between those smells and a wide range of socio-technical factors e.g., number of developers who have changed the code in a project [4] as well as various software artifacts e.g., code smells, fix-inducing changes [8] [12]. Besides, studies have been conducted to determine the effect of community smell on such artifacts [11]. Investigation has also been done to find out ways of community smell detection [5] and prediction techniques [9].

However, there is lack of sufficient research on refactoring community smells. In a study on a software company, Tamburri et al. [1] reported six mitigation decisions made by seven interviewees from the company's management board. However, they mentioned that one of the decisions greatly hampered further consequences while another caused the emergence of subversive behaviour across the community. Besides, the mitigation decisions from the management board of a single company could not represent the whole software industry.

Catolino et al. [6] defined an empirically grounded taxonomy of refactoring strategies for the most prominent four smells - Organizational Silo, Black Cloud, Lone Wolf and Radio Silence. To build the taxonomy, a study was conducted on 76 software practitioners that confirmed the relevance of community smells in industrial practice. The practitioners were chosen through Convenience Sample Recruitment Strategy [15]. Refactoring strategies adopted by them were collected through an online survey. Data was analyzed using Straussian Grounded Theory [17]. Common strategies mentioned in their study include mentoring, creation of a communication plan, and restructuring the community.

Sarmento et al. [13] replicated the same study on Brazilian software teams to identify and analyze localized refactoring practices. They chose participants following a mixed approach; participants were selected based on availability, and referral-chain [20]. They translated the survey instrument used by Catolino et al. [6] in Brazilian Portuguese. A total of 184 responses were collected in the survey. The mitigation strategies found from their study were similar to [6]. To explore whether refactoring practices vary in regions, similar studies need to be conducted for software communities in other regions, such as the software community in Bangladesh.

III. Methodology

This study investigates the local practitioners' perception of community smells and refactoring strategies adopted by them. To conduct the study, a draft interview questionnaire was prepared based on the questionnaire used by Tamburri et al [6], which was modified according to our goal. Then, pilot interviews were conducted to refine the draft questionnaire and depth-interviews were conducted on practitioners from the Bangladeshi software industry. Finally, data collected through interviews were transcribed to achieve the goals of the study.

A. Research Questions.

The goal of this study is to assess the current community smell refactoring practices in the software communities of Bangladesh. The following research questions RQs address the goals of this study.

RQ1: How do the practitioners perceive community smells in the local software industry?

RQ2: What are the strategies adopted in the software industry of Bangladesh to refactor community smells?

B. Subjects and Objects of the Empirical Study.

The subjects of this study were carefully chosen through the convenience sample recruitment strategy [15]. We conducted interviews with our personal contacts working as software professionals in the industry and recruited practitioners having broader view of software project development and management. 70% of the participants rated themselves to have considerable experience in project management, whereas 85% of them rated themselves to have considerable experience in software development. That is, the appropriate target audience were reached through the recruitment strategy chosen. We conducted a total of 21 depth-interviews where 90% of the participants work as software engineers including development team members, operations team members, and software architects, whereas 10% of them work as project managers. The 21 respondents have experience of working in a total of 18 different companies throughout their career. From these basic descriptive statistics, we can claim that the opinions collected are likely to provide us with reliable and generalized insights regarding the current community smell refactoring practices in the software industry of Bangladesh. Similar to [6] [7] [8], we collected data about the most prominent community smells: Organizational Silo, Black Cloud, Lone Wolf and Radio Silence to answer these questions. The definition of the smells are given below:

- Organizational Silo: The existence of silos in the developer community that communicate only through one or two of its respective members.
- Black Cloud: Information overload due to the lack of structured communication, people capable of bridging knowledge and experience gap between teams, and knowledge sharing occasions (e.g., daily stand-ups).
- Lone Wolf: The presence of defiant contributors who carry out their work without communication, with little regard for their peers and their decisions.
- Radio-Silence: The situation when one developer from a sub-team interposes herself into all formal communications between two or more sub-teams, becomes a bottleneck and prevents the introduction of other parallel communication channels.

C. Questionnaire Construction and Interview Conduction.

To collect participants' opinions, we designed an interview questionnaire based on the study conducted by Tamburri et

al. [6], which is composed of four main sections. The first section of the questionnaire describe a vignette-based scenario [16] for each of the four community smells. All the scenarios can be found in the questionnaire [1]. After explaining the scenarios, we ask the participants the following questions:
Q1. Has this situation ever happened to you?
Q2. If yes, how did you address this situation?
Q3. Do you think the mentioned scenario is problematic?
Q4. Do you think a systematic approach is necessary to deal with the above scenarios?
Q5. If there is such an approach, will you follow it?
Q6. Have you ever heard of community smells?
Q7. How frequently did the aforementioned scenarios occur?

Q1, Q3-Q6 can be answered with yes/no. Q2 is an open-ended question. Q7 is a multiple-choice question with three options: frequent, moderately frequent and not so frequent.

We prepared a version of the questionnaire in Google Form and filled up the answers during the interview. The answers to the open-ended questions were recorded for further analysis. Pilot interviews were conducted on 5 local software practitioners to refine the draft questionnaire. The final questionnaire was then developed based on feedback from the pilot interviews. 21 depth interviews with local practitioners were conducted. We kept the interviews anonymous due to privacy concerns. Prior to the interviews, participants were informed of their anonymity so that they could answer honestly. We never mentioned the term Community Smell until the end of the interview to avoid influencing the responses. The interviews lasted approximately 25-35 minutes.

D. Data Analysis.

We transcribed the interview responses from the recordings for further analysis. To address RQ1 we counted a) how many times the practitioners answered 'yes' to Q1 so that we can understand the relevance of community smells in the software industry of Bangladesh b) how many times have the practitioners answered 'yes' to Q3-Q5, so that we can identify how the practitioners perceive community smells in the local software industry c) how many times have the practitioners answered 'yes' to Q6, so that we can understand the state of their awareness regarding community smells.

To answer RQ2, we analyzed the transcribed responses to Q2 using Straussian Grounded Theory [17]. It is appropriate in analyzing open-ended questions with exploratory nature since it does not assume the presence of any previous theory to be tested over the data. To perform open coding, we split the sentences from the transcribed interview responses using standard text separators (e.g., commas or semicolons). Then, initial labels were assigned to the split parts based on the content. We clustered labels which are semantically similar or identical according to the semantic similarity principle [18]. The labels were renamed to better reflect the various categories of refactoring strategies adopted. We iterated over the labels until we reached a theoretical saturation [19]. Finally, based on the labels assigned to the transcribed responses, a taxonomy of refactoring strategies was built for each community smell considered in the study.

[1]https://forms.gle/7H5yewDVvJqKdeXm9

E. Measuring Refactoring Readiness.

We consider the following indicators to determine whether a software industry is ready to refactor community smells: a) Refactoring History: The more cases where community smells are refactored (either mitigated after occurrence or prevented before occurrence) in an industry, the more refactoring ready the industry is. b) Perception: The more community smells are perceived as harmful, the more eager the industry will be to refactor those smells, and it will be more refactoring ready. c) Knowledge: The more practitioners know about community smells, the better they realize the consequences. As a result, they will want to refactor those smells and will be willing to implement mitigation measures.

Based on these indicators, we propose a new measure named Refactoring Readiness (RR) to provide a quantification of the practitioners' preparedness for adopting community smell refactoring strategies. Equation (1) presents the formula for measuring Refactoring Readiness of a software community; where *n* is the total number of smells taken into consideration, ts_i is the percentage of cases where either mitigation action was taken after occurrence of the smell *i* or practitioners did not experience the smell as a result of preventive measures, *e* is the percentage of practitioners who perceive community smells as harmful and want a mitigation strategy, and *k* is the percentage of practitioners who have heard of community smells before. We add up all the values for the indicators to get an overall idea of how much refactoring ready a software community is. Here, *n+2* is used as a normalization factor. The value for ts_i was found analyzing the answers of Q1 and Q2. The values for *e* was found analyzing the answers of Q3-Q5. The value for *k* was found analyzing the answer of Q6.

IV. Result

In this section, we report and discuss the results of our study. Our findings on the practitioners' perception of community smell in the software industry of Bangladesh and their Refactoring Readiness are presented first. Then, the refactoring strategies adopted by them to mitigate the Organizational Silo, Black Cloud, Lone Wolf and Radio Silence smells are presented.

*A. **RQ1. Practitioners' Perception of Community Smells***

To understand how the practitioners perceive community smells, we asked them Q1 and Q3-Q6 mentioned in Section III-C. Analyzing the responses to Q1, we see that, 70%, 65%, 80%, and 60% of participants faced the Organizational Silo, Black Cloud, Lone Wolf, and Radio Silence smells respectively. In 80.9% of the cases, the smells were frequent or moderately frequent. This was found from the responses to Q7. From the responses of Q3-Q5, it is seen that, 85% of them perceive community smells as detrimental to software

TABLE I
MEASURES AND THEIR CORRESPONDING VALUES FOR CALCULATING REFACTORING READINESS

Metric	Value	Metric	Value	Metric	Value
$ts_{organizationalsilo}$	0.7142	$ts_{lonewolf}$	0.7619	e	0.85
$ts_{blackcloud}$	0.5952	$ts_{radiosilence}$	0.5714	k	0.30

maintenance and perceive the need for some refactoring strategies. Moreover, analyzing the responses to Q2, it was found that in 38.1%, 47.61%, 19.04% and 42.86% of cases where the Organizational Silo, Black Cloud, Lone Wolf, and Radio Silence smells occur respectively, no refactoring strategies are adopted. Furthermore, from the responses to Q6, we see that, 70% of the respondents had never heard about community smells before. That is, the local practitioners faced these smells, perceive them as harmful but do not take sufficient measures to mitigate those smells.

Based on the data, we calculated the Refactoring Readiness of the local software industry using (1). Since only the four most prominent smells are being studied in this study, the value of n in our case is 4. The values of ts_i, e and k for our data is provided in Table 1. The Refactoring Readiness of the local software practitioners calculated from our data is 0.63 on a scale of 0-1. This score shows that more focus is necessary for making the local practitioners refactoring ready.

*B. **RQ2. Community Smell Refactoring Strategies Adopted by Local Practitioners***

In this subsection, we discuss the refactoring strategies adopted by the software practitioners of Bangladesh.

1) Organizational Silo: Practitioners reported the following strategies to deal with this smell.

Creating a structured communication plan. 38.1% of practitioners create a detailed and structured communication plan to ensure that communication channels are functioning properly and members are effectively conveying messages. Team leaders discuss the issue with the sub-team in charge and devise communication strategies to address it. Daily scrums, weekly follow-up meetings, and ice-breaking sessions are held. Number of meetings are increased to address the consequences of the smell. Strict guidelines are established to ensure everyone is communicating effectively.

Mentoring. In 4.76% of the cases, the project managers collaborate with practitioners to find solutions and ensure that everyone understands the communication rules. They ensure that everyone on the team is following the rules, communicating effectively, and keeping each other informed. Pressure from higher authorities is also applied in this case.

Restructuring communication approach. To address the problems caused by the smell, management replaces team members with new members in charge of communication. Furthermore, when a team member goes on leave, she designates someone else to cover for her in order to avoid the formation of silos. This strategy is used by 4.76% of practitioners.

$$RR = \frac{(\sum_{i=1}^{n} ts_i) + e + k}{n + 2} \quad (1)$$

Indirect approach. In 4.76% of the cases, the management reportedly adopt an indirect strategy to address this smell. For example, in daily meetings, they ask developers questions like "what has been done today?" rather "what have you done?" To respond to such a question, the team member must stay informed of what his fellow team members are doing. This results in the person responsible for the smell becoming aware of herself and improving her communication skills in future.

Nothing done. 38.1% of practitioners who experienced this smell reported that, despite the negative consequences and practitioners' interest, team management is not interested in resolving this problem. In 19.04% cases, the management does not perceive this as a problem at all.

2) Black Cloud: Practitioners reported the following strategies to deal with this smell.

Creating a structured communication plan. To refactor this smell, 67.67% of practitioners use a structured communication plan. They maintain regular formal and informal communication (e.g., daily scrum, day-end meetings,weekly follow-up meetings). The team leader thoroughly explains tasks to newcomers. Issue tracking tools (e.g. Jira, Trello) and communication tools (e.g. Skype, Microsoft Teams, Slack) are extremely beneficial in this regard. Proper documentation is maintained, recordings of meetings are stored for members absent in meetings so that they can catch up later. Members make up any missing items through subsequent formal or informal discussions. Furthermore, a clear HR policy for timely communication is beneficial to deal with this smell. For example, when employees go on leave, they notify the team lead as soon as possible.

Creating realistic task plan. 4.76% of the practitioners ensure realistic task assignment and time estimation considering the developers' viewpoint to deal with these smells.

Introduction of a social sanctioning mechanism. According to 4.76% of the practitioners, community members violating the specified communication guidelines and failing to attend review sessions are penalized to address this issue.

Restructuring community. 4.76% of the practitioners reported, when something is unclear (e.g. work context, particular feature), pairing of developers is done to clarify confusion.

Nothing done. 47.61% practitioners reported that team management is not interested in solving the issue; developers had to suffer the consequences on their own.

3) Lone Wolf: Practitioners reported the following strategies to deal with this smell.

Mentoring. 52.38% of the practitioners reported that mentoring is done to mitigate this smell. If necessary, the higher authority steps in to solve the problem.

Creating a structured communication plan. To identify and groom lone wolves, Weekly meetings are carried out. Work decisions are made after discussions in the presence of them. Team members communicate with them on a regular basis. Furthermore, the lone wolf is instructed to keep proper documentation of every task they perform in order to avoid information gaps. 19.05% practitioners follow this strategy.

Restructuring community. 14.28% of practitioners claim that lone wolves are paired with other developers so that the latter is kept informed of the lone wolf's tasks. Additionally, peer reviews are set up to address this problem. Moreover, to utilize the lone wolfs, team leaders assign the lone wolves to solo projects since solo projects require less communication.

Increased Cohesion. According to 4.76% of practitioners, arranging activities to increase cohesion among team members do not leave any room for the emergence of this smell.

Termination. 9.52% practitioners said that, if the lone wolf remains the same even after mentoring, they are terminated.

Nothing done. 19.04% practitioners claimed that team management takes no action to address the problem. 9.52% of them do not even consider it to be problematic.

4) Radio Silence: Practitioners reported the following strategies to deal with this smell.

Restructuring community. 38.1% of the practitioners reported that small teams are formed and parallel communication channels are established to address this smell. Additionally, to lessen the burden on the bottleneck, tasks and necessary knowledge are distributed among several people.

Creating a structured communication plan. A communication plan is developed to specify who the bottleneck should communicate with when they encounter problems. This enables them to get help and speed up the process. Besides, team members are encouraged to communicate spontaneously so that one particular individual do not have to maintain all the communication. 14.28% of practitioners followed this strategy.

Increased Cohesion. 4.76% of practitioners claimed to have work environment that improves team cohesion, which prevents the emergence of this smell.

Nothing done. 42.86% of those who faced this smell claimed that team management takes no action to address the problem. 4.76% of them do not consider it as problematic.

According to the findings, the Refactoring Readiness score of the software industry in Bangladesh is 0.63. This indicates that practitioners in the Bangladeshi software industry are still not ready to effectively refactor community smells, and that more work is required to prepare the industry for refactoring those smells. The refactoring strategies identified in this study can be useful in this regard. Practitioners who do not take any mitigation measure can follow this strategies to refactor community smells.

V. Conclusion

Refactoring community smells from software industry requires understanding the practitioners' current refactoring practices first. In this study, we have taken the first step towards addressing these issues for the software industry of Bangladesh. An interview questionnaire was designed and depth interviews were conducted on 21 software practitioners of Bangladesh. A measure for calculating the Community Smell Refactoring Readiness (RR) of a development community was introduced, the Refactoring Readiness for the software industry of Bangladesh was calculated as well as the refactoring strategies that the local practitioners adopt were presented. This study confirms that there is a visible lack of awareness among local practitioners regarding community smell refactoring. Our research agenda includes replicating the study with more practitioners and increasing the generalizability of the results achieved so far. We plan to include rest of the 21 community smells in our future work.

References

[1] Tamburri, D.A., Kruchten, P., Lago, P. and Vliet, H.V., 2015. Social debt in software engineering: insights from industry. Journal of Internet Services and Applications, 6(1), pp.1-17.

[2] Tamburri, D.A., Kazman, R. and Fahimi, H., 2016. The architect's role in community shepherding. IEEE Software, 33(6), pp.70-79.

[3] Tamburri, D.A., 2019. Software architecture social debt: Managing the incommunicability factor. IEEE Transactions on Computational Social Systems, 6(1), pp.20-37.

[4] Almarimi, N., Ouni, A. and Mkaouer, M.W., 2020. Learning to detect community smells in open source software projects. Knowledge-Based Systems, 204, p.106201.

[5] Almarimi, N., Ouni, A., Chouchen, M., Saidani, I. and Mkaouer, M.W., 2020, June. On the detection of community smells using genetic programming-based ensemble classifier chain. In Proceedings of the 15th International Conference on Global Software Engineering (pp. 43-54).

[6] Catolino, G., Palomba, F., Tamburri, D.A., Serebrenik, A. and Ferrucci, F., 2020, June. Refactoring community smells in the wild: the practitioner's field manual. In Proceedings of the acm/ieee 42nd international conference on software engineering: Software engineering in society (pp. 25-34).

[7] Catolino, G., Palomba, F., Tamburri, D.A., Serebrenik, A. and Ferrucci, F., 2020. Gender Diversity and Community Smells: Insights From the Trenches. IEEE Software, 37(01), pp.10-16.

[8] Palomba, F., Tamburri, D.A., Fontana, F.A., Oliveto, R., Zaidman, A. and Serebrenik, A., 2021. Beyond technical aspects: how do community smells influence the intensity of code smells?. IEEE Transactions on Software Engineering, 47(1), pp.108-129.

[9] Palomba, F. and Tamburri, D.A., 2021. Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach. Journal of Systems and Software, 171, p.110847.

[10] Tamburri, D.A.A., Palomba, F. and Kazman, R., 2021. Exploring community smells in open-source: an automated approach. IEEE Transactions on Software Engineering, 47(3), pp.630-652.

[11] Ahammed, T., Ahmed, S. and Khan, M.S.A., 2021. Do Missing Link Community Smell Affect Developers Productivity: An Empirical Study. Knowledge Engineering and Data Science, 4(1).

[12] Ahammed, T., Asad, M. and Sakib, K., 2021. Understanding the Relationship between Missing Link Community Smell and Fix-inducing Changes. In ENASE (pp. 469-475).

[13] Sarmento, C., Massoni, T., Serebrenik, A., Catolino, G., Tamburri, D. and Palomba, F., 2022, December. Gender Diversity and Community Smells: A Double-Replication Study on Brazilian Software Teams. In 29th IEEE International Conference on Software Analysis, Evolution and Reengineering.

[14] Palomba, F., Tamburri, D.A., Fontana, F.A., Oliveto, R., Zaidman, A. and Serebrenik, A., 2018. Beyond technical aspects: How do community smells influence the intensity of code smells?. IEEE transactions on software engineering, 47(1), pp.108-129.

[15] Robinson, O.C., 2014. Sampling in interview-based qualitative research: A theoretical and practical guide. Qualitative research in psychology, 11(1), pp.25-41.

[16] Finch, J., 1987. The vignette technique in survey research. Sociology, 21(1), pp.105-114.

[17] Corbin, J.M. and Strauss, A., 1990. Grounded theory research: Procedures, canons, and evaluative criteria. Qualitative sociology, 13(1), pp.3-21.

[18] Harispe, S., Ranwez, S., Janaqi, S. and Montmain, J., 2015. Semantic similarity from natural language and ontology analysis. Synthesis Lectures on Human Language Technologies, 8(1), pp.1-254.

[19] Walker, J.L., 2012. Research column. The Use of Saturation in Qualitative Research. Canadian journal of cardiovascular nursing, 22(2).

[20] Baltes, S. and Ralph, P., 2022. Sampling in software engineering research: A critical review and guidelines. Empirical Software Engineering, 27(4), pp.1-31.