An Empirical Study on the Relation between Programming Languages and the Emergence of Community Smells

Giusy Annunziata, Carmine Ferrara, Stefano Lambiase, Fabio Palomba,
Gemma Catolino, Filomena Ferrucci, Andrea De Lucia
Software Engineering (SeSa) Lab – University of Salerno, Salerno, Italy

***Abstract*—To provide a measurable representation of social issues in software teams, the research community defined a set of anti-patterns that may lead to the emergence of both social and technical debt, *i.e.*, *"community smells"*. Researchers have investigated community smells from different perspectives; in particular, they have analyzed how product-related aspects of software development, such as architecture and introducing a new language, could influence community smells. However, how technical project characteristics may be in relation to the emergence of community smells is still unknown. Different from those works, we aim to investigate how adopting specific programming languages might influence the socio-technical alignment and congruence of the development community, possibly inducing their overall ability to communicate and collaborate, leading to the emergence of social anti-patterns, *i.e.*, community smells. We studied the relationship between the most used programming languages and the community smells in 100 open-source projects on GITHUB. Key results of the study show a low statistical correlation for specific community smells like *Prima Donna Effects*, *Solution Defiance*, and *Organizational Skirmish*, highlighting the fact that for some programming languages, its adoption could not be an indicator of the presence or absence of community smells.**

***Index Terms*—Software Organizational Structures; Programming Languages; Community Smells; Empirical Studies.**

I. INTRODUCTION

Software development represents *de facto* a socially intensive activity where technical and social factors are intertwined; this encompasses the attention on software products and the relationship among the project stakeholders. Since decades ago, Brooks Jr [5] highlighted the importance of social and human factors as significant characteristics for ensuring software projects' success; other works [4, 20] were more specific promoting people's management and social interactions as vital aspects instead.

Researchers went beyond studying social aspects, deepening the relationship between them and the technical factors of a software product. In this context, the theory of *socio-technical congruence* stands out, as it focuses on understanding how well the organizational structure of a group aligns with the software architecture of the product they are creating [35, 40, 42]. Always related to socio-technical aspects, the research community focused on providing a measurable representation of them, defined *community smells*, *i.e.*, social anti-patterns that characterize collaboration and communication in software communities and can lead to social debts [36, 38]. Furthermore, Tamburri et al.[37] observed that development choices, *e.g.*, adopting a new language, are indirectly connected to social debt. Later, Wang and Hahn[47] verified that development style impacts the collaborative aspects of an open-source team.

Previous research has shown a strong correlation between software development's technical and social aspects. Ignoring the social aspect when studying the technical side could result in misleading conclusions. Although the research community effort in investigating phenomena such as socio-technical is increasing, the relation between the technical development aspects and the emergence of community smell is still unknown . Starting from this consideration and aiming to fill the gap, we used statistical analysis to verify the correlation between adopting a specific programming language and the emergence of community smells in open-source development teams.

We conjecture that the features implemented by programming languages—*e.g.*, inheritance and delegation mechanisms—might influence the socio-technical alignment and congruence of the development community, possibly inducing their overall ability to communicate and collaborate and leading to the presence/absence of community smells.

To verify our conjecture, we operationalized community smells on a set of 100 GITHUB open source projects, discriminated for their most used programming language. We extracted information on 32 socio-technical metrics and the presence of 10 different smells, ranging from Organizational Skirmish to Toxic Communication described in Table I), computing using CADOCS [45]. Then, we built a statistical model to analyze the relationship between the most used programming language and the presence of community smells.

The key results of the work highlight that a correlation exists between the use of some programming languages and the presence of community smells. For example, Python and PHP languages seem to be correlated with the community smell Prima Donnas Effects, *i.e.*, a situation in which a team member exposes constant disagreement and uncooperativeness [30]. Project managers could use those results to be aware of potential social anti-patterns and implement mitigation and contingency strategies to better perform risk management since the initial phases of the project.

II. Background and Related Work

In this work, we put our attention on the *community smells* [38], *i.e.*, anti-patterns in software development communities' communication and collaboration processes, often precursors of *social debt* [37]. An example of such smells is the *Radio Silence Effect* that represents the situation in which a stakeholder interposes themselves into every formal interaction across more sub-communities with little flexibility to introduce other channels [38].

Regarding studies investigating factors influencing and impacting community smells, Catolino et al. [8] studied how team composition correlates with the emergence of community smells, finding that gender diversity might reduce their presence. Later they also considered which factors correlate their variability [9], finding that socio-technical factors related to tenure and centrality of a developer can increase the presence of community smells. Finally, Lambiase et al. [20] conducted a quantitative investigation on the subject, identifying *cultural and geographical dispersion—i.e.*, how much a community is diverse in terms of its members' cultural attitudes and geographical collocation—as potential factors for the emergence and mitigation of community smells.

Starting from the previous literature on the matter, we conducted a novel study concerning how technical factors are correlated to the emergence of community smells. In particular, we started from the work of Valetto et al. [42] that showed that product-related software development aspects might largely influence a community's communicative and collaborative aspects. Moreover, Syeed and Hammouda [35] showed that the architectural design and interdependency among the modules of the software are directly related to the communication and collaboration patterns of the developer community of specific open source projects, *i.e.*, socio-technical congruence. Next, Tamburri et al. [37] observed that technical choices, like the introduction of a new programming language, can indirectly impact social aspects in a development context. Later, Wang and Hahn [47] went deepening demonstrating that specific product factors, *i.e.*, metrics inherent to programming style and language, are related to the collaborative aspects of a team in open-source development projects, but, to date, no one formally verified these kinds of correlation.

III. Research Study Design

The *goal* of this study is to analyze whether the adoption of a particular programming language—and what it implicitly entails, *e.g.*, different styles or skills—influences the presence of social anti-patterns during software development. The *propose* is to provide new insights to allow practitioners to increase awareness of possible issues within their software development community. The *perspective* is of project managers interested in monitoring product software factors, possibly discovering any helpful indicator of risky factors like community smells. The *objective* of the study is driven by some consideration. In particular, from the fact that the technical aspects and choices of software development—*i.e.*, architectural design, metrics inherent to programming style and languages, etc.—may largely influence the communicative and collaborative aspects [35, 37, 42, 47]. As such, we define the following research question:

> ***Research Question.*** *To what extent does the used programming language relate to social anti-patterns?*

Starting from those research question, we conjectured the following hypothesis:

H_0 *The adoption of a specific programming language does not significantly influence the extent of community smells.*

H_1 *The adoption of a specific programming language significantly influences the extent of community smells.*

We address this research question by investigating the correlation between the programming languages used in 100 GitHub open source projects and the emergence of social anti-pattern operationalized by using community smells [38]. The rationale behind this choice relied on previous literature Wang and Hahn [47] that analyzed the impact of programming style and programming language inconsistencies on open-source software collaboration, conjecturing that they could impact a software development community's communication and collaboration network. Besides their analysis, our intention was to deepen the relationship between software product factors and social metrics by quantitatively verifying if the *used programming language* is related to the presence of social anti-patterns in a development community, represented using community smells. Figure 1 shows an overview of the study. In particular, we perform the following steps.

> **$Step_1$ - Creating a Dataset**
>
> *Create a dataset containing open-source projects developed using different programming languages where it is possible to detect community smells and other socio-technical characteristics.*

We constructed a dataset consisting of 100 open-source repositories, taking into account information on (1) the programming language used, (2) factors previously found for indicating community smells presence [36], and (3) the presence or absence of specific community smells.

> **$Step_2$ - Building a Statistical Model**
>
> *Study statistical relationship between the programming languages used and the presence of community smells.*

For each community smell considered in our study, we built a statistical model to check whether the choice of the programming language used correlates significantly with the presence or not of a community smell.

We provide the online appendix [1] containing the data and scripts used and the results obtained.

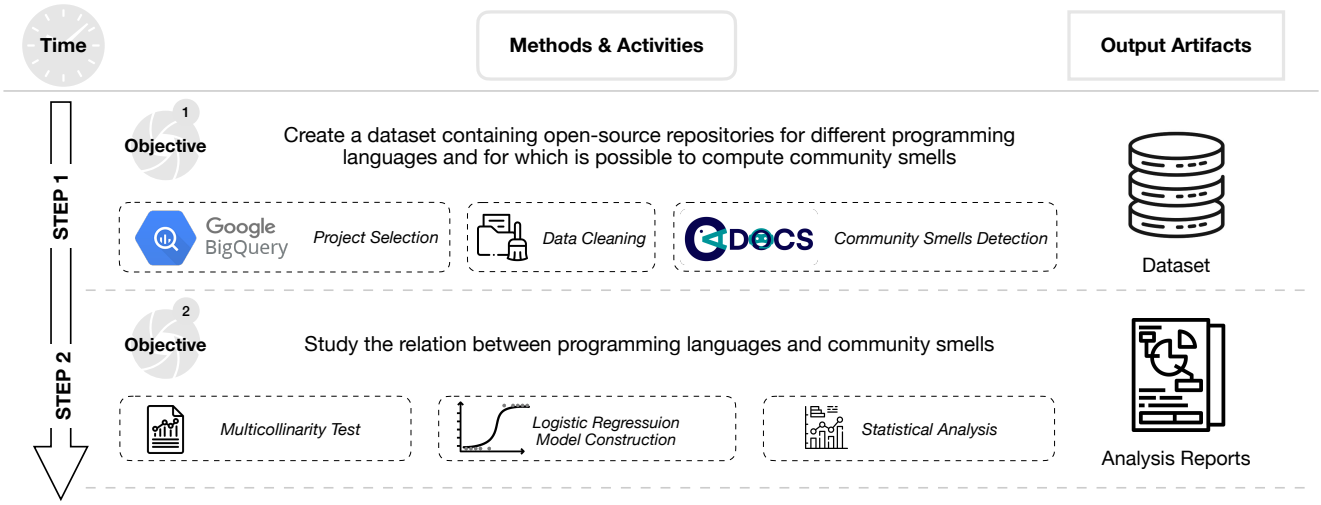

Fig. 1. Overview of our research study design.

TABLE I
COMMUNITY SMELLS.

Community Smells Definition
Organizational Skirmish (OS): Teams composed of members with different levels of competence lead to a drop in productivity, impacting time and costs.
Black Cloud Effect (BC): A lack of structured communications or cooperative governance can lead to information overload.
Radio Silence (RS): The use of formal communication between multiple subcommunities penalizes flexibility and causes loss of time.
Prima Donnas Effect (PD): Team members that expose condescending behavior, superiority, constant disagreement, and uncooperativeness.
Sharing Villainy (SV): Failure to exchange information can lead team members to share incorrect, obsolete, and unconfirmed information.
Organizational Silo Effect (OSE): Siloed community that do not communicate with each other except through one or two of their respective members.
Solution Defiance (SD): Having a community with different backgrounds of experience and culture can lead to subgroups with conflicting opinions.
Truck Factor Smell (TFS): Developer turnover may cause a significant loss of knowledge as it is concentrated in a minority of developers.
Unhealthy Interaction (UI): Slow, light, and short conversations and discussions caused by long delays in stakeholder communication.
Toxic Communication (TC): Developers may negatively interact with their colleagues, leading to frustration, stress, and project abandonment.

To conduct our analysis, we employed the guidelines by Wohlin *et al.* [50] and we followed the *ACM/SIGSOFT Empirical Standards*. Specifically, we employed the *"General Standard"* and *"Data Science"* definitions and guidelines.[1]

A. *Step$_1$* - *Building a dataset*

To conduct our study, we created a dataset by consulting the existing databank offered by GOOGLEBIGQUERY.[2] It is a Restful Web Service that allows interactive analysis of a large database that works with Google Storage. Moreover, it also contains the GITHUB ARCHIVE,[3] *i.e.*, the entire public repository of GITHUB that is automatically updated every hour. The execution of the queries on GOOGLEBIGQUERY and the creation of the dataset occurred between January and February 2022. As a first step, we focused on understanding the rationale behind extracting projects with different programming languages. Utilizing a query, we obtained a list of the most commonly used programming languages. Using a query, we labeled each project with the most present programming language among those used to develop it. After pulling the projects, we detected community smells using the already validated CADOCS [45] tool. CADOCS leverages a Slack interface to parse GitHub repositories whose links are passed as parameters and CADOCS extracts information about the presence of ten different community smells—presented in Table I—and additional metrics, *e.g.*, # commits, for constructing our statistical models. Further details are provided in the following section.

Selection Criteria: To create the dataset, we focused on the 10 most popular programming languages.[4] We selected the first 10 projects with the most different characteristics according to (1) Programming paradigm, (2) Compilation class, (3) Typing class, and (4) Memory class, according to Ray et al. [31]. For each of the 10 most used programming languages selected, we built a dataset containing the name, the link, and the number of contributions of several GITHUB projects. We selected 10 projects from each dataset, thus obtaining a complete dataset composed of 100 projects, 10 for each language, with the number of contributors between 30 and 50, ranges that guarantee a sub-optimal compromise between construct and external validity of the projects. We identified the number of contributors as the primary criterion for selecting projects. Our investigation revealed that projects with 30 to 50 contributors adequately represent the diver-

[1]ACM/SIGSOFT Empirical Standards: https://github.com/acmsigsoft/EmpiricalStandards

[2]https://cloud.google.com/bigquery?hl=en

[3]https://www.gharchive.org

[4]Spectrum-Report Top Programming Languages 2022: https://spectrum.ieee.org/top-programming-languages-2022

sity within each programming language. Consequently, we chose to focus solely on projects falling within this range of contributors. Furthermore, we performed a content analysis session [22] to label all 100 projects, aiming to categorize them based on their descriptions and features as provided by the authors. The first and second authors conducted the labeling process and agreed to assign each project a single value.

B. *Step₂ - Building a Statistical Model*

After data extraction, we constructed a statistical logistic model, for each community smell considered by CADOCS—Table I—to understand the relationship between a particular programming language and the presence of the specific smell. considering that the presence of community smells can assume Boolean value, we applied a *logistic regression* [41].

1) Independent Variable: We considered the development community's most used programming language as the independent variable. Moreover, since the relative variable is expressed in the dataset as categorical, we replaced it with a *dummy variable* before creating the regression models [3].

2) Response Variable: Since our goal was to understand the relation among the adoption of different programming languages and *presence/absence of a specific community smell*, we encoded this factor as the Response Variable. According to CADOCS [45] tool's output data representation, our response variable assumed value 1 when a specific community smell is present in a development project, 0 otherwise.

3) Control Variables: To well design a statistical model, a practice to consider is adding other variables that might affect the phenomenon analyzed beyond the independent variables. For this reason, we included a set of control variables that previous studies [8, 9, 17, 20, 28, 43] demonstrated to have a correlation with community smells:

1) **Number of commits**—the number of commits executed on a project repository. In most cases, a high number of commits is a symptom of the high productivity of the project team. But often, multiple committers work on the same portions of code, thus affecting the number of community smells [30];
2) **Project Contributors**—the number of different contributors that performed at least one commit on the project repository. Catolino et al. [8] demonstrated that this factor could influence the presence of community smells;
3) **Active days**—The number of active days of the project repository, i.e., the number of days the project has been active and at least one contributor works on the project [30];
4) **Bus Factor Number**—a specific number that estimates the minimum number of team members in a project that suddenly abandon the project before it fails due to lack of experienced personnel [49].

TABLE II
COMMUNITY SMELL DETECTED BY CADOCS [45]

Language	OS	BC	PD	SV	OSE	SD	RS	TFS	UI	TC
C	3	7	9	8	6	8	1	3	4	2
C#	3	3	6	8	2	7	3	5	5	5
C++	4	4	9	8	4	7	1	5	2	5
Go	3	10	9	6	7	9	4	2	4	0
Java	6	7	10	5	4	9	1	4	5	2
JavaScript	4	8	10	8	4	9	3	5	6	3
PHP	4	7	10	3	6	10	2	5	4	1
Python	2	6	10	10	5	7	3	3	6	3
Scala	2	6	10	10	5	7	3	3	6	3
TypeScript	3	7	8	6	4	10	4	7	5	4
Total	34	65	91	72	47	83	25	42	47	28

The occurrences of community smells for each language are made on 10 projects.
Total occurrences of community smells are made on 100 projects.

4) Statistical Model Construction: For the model construction, we relied on the functions `ols` available in the Python package `Stats Model` [32]. Based on guidelines and similar studies [16, 19], we faced the problems of data normality and multicollinearity [27] that can affect the reliability of the results. For this reason, we performed the well-known SHAPIRO-WILK test to verify the normality of data [33] and measured the PEARSON'S CORRELATION INDEX and the VARIANCE INFLATION FACTOR (VIF) to verify the absence of multicollinearity between the independent variables.[5] Moreover, to strengthen the reliability of our results, we evaluated the effect sizes of the model coefficients performing the ANOVA statistical test [14], provided in Python in package `Stats Model` [32]. According to ANOVA, we considered variables statistically significant if the p-value is less than 0.05.

Finally, we built a baseline statistical model containing only the control variables to analyze the models' reliability in predicting results. We evaluated the models with the corresponding baselines through the AIC (AKAIKE INFORMATION CRITERION) and BIC (BAYESIAN INFORMATION CRITERION) [2, 6] estimators, that are widely used as quality prediction criteria of statistical models [7]. Comparing AIC and BIC for each community smell prediction model and the relative baseline, we could identify the more statistically reliable one considering the lower values of these indicators.

IV. ANALYSIS OF THE RESULTS

In this subsection, we discuss the main findings of our study. We report details about: (1) dataset composition and (2) the main statistics about logistic regression analysis.

A. *Step₁—Dataset Construction*

This section briefly describes the open-source dataset we constructed to analyze the relationship between programming languages and community smells.

As a first step, we obtained an initial collection of 34678 open-source projects from GOOGLEBIGQUERY. Then, we

[5]We relied on their implementation provided by the Python packages `SciPy` [44] and `Stats Model` [32].

used CADOCS to compute socio-technical metrics and detect community smells on 10 projects for each programming language we previously selected (we considered only projects from 30 to 50 contributors to guarantee similarity, as explained more detailed in Section III-A). The dataset was created in *January 2022*, and CADOCS was executed in *February 2022*. The data extracted from the projects spans from their inception to February 2022. As last step, we analyzed the distribution of community smells for each programming language taken into consideration; Table II shows that there is a frequent presence of the community smells Prima Donnas Effect in most of the languages, but particularly in Python, Java, JavaScript, and PHP. In our online appendix [1] are available specific reports about the number of repositories detected by GOOGLEBIGQUERY for each selected programming language.

☞ Step$_1$: summary of the results.

The main results of the first step of the study consists in a dataset that reports information about the presence of Community Smells in 100 GITHUB open-source projects developed in 10 different programming languages. Particularly we identified a significant presence of the smell Prima Donnas Effect in projects developed in Python, Java, JavaScript, and PHP.

B. *Step$_2$—Statistical model*

This section shows the findings achieved when assessing the relationship between the chosen programming languages in a GITHUB open-source project and the presence or the absence of a specific community smell.

Table III reports the details regarding the statistical models realized. For each community smell considered in our dataset, the table reports the results achieved for the relative model with both independent and control variables. In particular, the language variable is represented as a dummy variable [3], so the table reports details for each value that the original qualitative one can assume.

Our results revealed that most of the community smells seem not to be correlated with the programming languages. Nevertheless, in inducing the community smell Prima Donnas Effects, the ANOVA test shows a more statistically significant correlation with the languages JavaScript, Java, PHP, and Python ($p-value < 0.01$). Other results are a low statistically significant correlation between the program languages C++ and Python and the community smell Solution Defiance. Another proof of low statistical significance of the independent variable, the values of AIC and BIC are similar both with and without the programming language variables as the community smells, inducing factor.

By analyzing our results, we may assume that the type of projects analyzed could be one of the origins of the correlation between our variables. State-of-the-art already demonstrated that the programming language often depends on the type of software that we want to develop [23]—*e.g.*, support tool or web application. This is an obvious fact, and as obvious as it is that different types of projects need different development approaches; for example, managers tend to prefer a prototyping lifecycle for support tools rather than a classical waterfall for desktop applications. Such a different approach could be the origin of collaboration and communication patterns resulting in social debt, and community smells.

For all these reasons, we conducted an iteration of content analysis session [22] to label all the 100 projects; in particular, the goal was to obtain clusters to categorize the projects according to the description and features provided by the authors. The first and second authors conducted such labeling and agreed on a single value to be assigned to the projects. These labeling values are given in the dataset in the online appendix [1]. Most of the project in the database were identifiable as *Development Support—i.e.*, a tool or a plug-in that supports developers in their activities. It may be possible that since such tools have not been developed by organized teams of practitioners but by research teams—potentially geographically distributed—the development process could suffer from a lack of organization, leading to the emergence of community smells. Indeed, further research on the relationship between the type of project and team and the emergence of community smells should be conducted to shed light on such a correlation.

☞ Step$_2$: summary of the results.

Our main findings show how *the programming languages* are not always correlated to the presence or absence of community smells in an open Source Development Process. An exception occurs for languages like Python, PHP, Java, and Javascript for Prima Donnas Effect and for Python and C++ for Solution Defiance.

Results

In conclusion, our findings suggest that specific programming languages are correlated with the occurrence of community smells, providing grounds to reject the null hypothesis for those languages. However, this correlation is not significant for all languages, indicating insufficient evidence to reject the null hypothesis in those cases.

V. DISCUSSION AND IMPLICATIONS

This Section discusses some implications for future research based on our findings reported in Section IV.

The most related community smell by source code. Analyzing the results of the study, it emerged that the most frequent community smells among the analyzed projects are Prima Donnas Effect and Solution Defiance. In particular—as shown in Table II—, they appear in all projects developed in Java, JavaScript, Python, PHP, and Scala and in 90% of projects developed in C++ and Go. Interestingly, the two

TABLE III
RESULTS ACHIEVED BY THE LOGISTIC REGRESSION MODEL WITH ALL VARIABLES.

Factor	OS		BC		RS		PD		SV		OSE		SD		TFS		UI		TC	
	Est.	Sig.	Est.	Sig.	Est.	Sig.	Est.	Sig.	Est.	Sig.	Est.	Sig.	Est.	Sig.	Est.	Sig.	Est.	Sig.	Est.	Sig.
(Intercept)	-0.437		-0.582		-0.043		1.577		0.594		-0.702		-1.103		1.250		0.426		-0.112	
Independent Variable: Programming Language (Shown as dummy variable)																				
Language: C	-0.068		0.035		-0.189		0.027		0.120		0.186		0.053		-0.013		0.011		-0.095	
Language: C#	-0.138		-0.256	—	-0.047		-0.138		0.101		-0.340	—	-0.207	—	0.187		-0.009		0.121	
Language: C++	0.014		-0.173		-0.288	—	0.149		0.115		-0.162		-0.217	*	0.185		-0.344	—	0.122	
Language: Go	-0.039		0.207		0.197		0.177	—	0.007		0.166		-0.136		-0.013		0.041		-0.192	
Language: Java	0.180		-0.001		-0.162		0.299	**	-0.149		-0.154		-0.114		0.167		0.084		-0.088	
Language: JavaScript	0.019		0.099		0.058		0.254	**	0.173		-0.110		-0.053		0.229		0.205		0.044	
Language: PHP	0.058		-0.024		-0.027		0.278	**	-0.319	*	0.066		-0.004		0.264		0.002		-0.121	
Language: Scala	-0.118		-0.079		0.132		0.085		-0.022		-0.173		-0.051		0.488	**	0.105		0.121	
Language: Python	-0.219		-0.132		0.049		0.293	**	0.362	*	-0.056		-0.330	**	0.078		0.200		0.025	
Language: TypeScript	-0.127		-0.258	—	0.235		0.160		0.126		-0.126		-0.045		0.077		0.131		-0.050	
Control Variables																				
# of commits	5.733e-06		1.0e-04	***	8.705e-05	**	1.127e-05		4.403e-05		1.384e-05		-5.596e-05	*	4.786e-05		9.53e-05	*	8.061e-05	*
Project Contributors	0.035	—	0.015		-0.009		0.019		0.006		0.013		0.031	*	-0.022		-0.0129		0.009	
Active Days	3.821e-05		7.627e-05	—	5.388e-05		3.059e-05		-2.17e-05		2.243e-05		9.388e-06		4.262e-06		-5.994e-06		4.629e-05	
Bus Factor Number	-0.640	—	0.671	*	0.143		-0.224		-0.120		0.693	*	0.982	***	-0.535		0.395		-0.172	

$***$:$p<0.001$; $**$:$p<0.01$; $*$:$p<0.05$; $-$:$p<0.1$

community smells share, by definition, a common problem: *the need for more communication between all team members*. In particular, insufficiently structured communication in the case of the Prima Donnas Effect and conflicting communication in the case of Solution Defiance. *This lack of communication is probably accentuated between the team and the project manager when the latter adopts specific programming languages such as those mentioned above.* Therefore, it might be interesting to investigate a possible correlation between the community smells mentioned above by repeating the study with a larger dataset mainly focused on these specific programming languages.

Programming language vs. community smells. Among the programming languages shown to have a correlation with community smells, we found a subset of sector-specific languages such as Python and JavaScript. A potential interesting context to explore could be the one of *Machine Learning Enabled* [21] systems—software products characterized by both a classical component and a machine learning one—and the related quality and community aspects. Those specific systems (i) strongly present the use of Python (or JavaScript) as a programming language and (ii) the development team usually is composed of different subgroups—*e.g.*, data scientists and software engineers—working together [26]. Such heterogeneity could lead to the emergence of conflicts between the various subgroups, potentially leading to social debt. Alternatively, the presence of socio-technical issues may be due to specific development skills required in these areas. For these reasons, in fields like Machine Learning, old development patterns, and habits might be disrupted by newcomers [34].

The impact of socio-technical factors. Our study reinforces what has already been shown regarding socio-cultural metrics [39]. In particular, our findings show that the number of committers has high statistical significance in influencing the dependent variable. Furthermore, another statistically significant control variable is the Bus Factor Number. Since these two variables are directly correlated with the size of the development team, it might be interesting to investigate how prediction models based on this type of variable statistically impact the dependent variable based on projects with teams of varying sizes. Considering this, our study could be enhanced by including other socio-technical indicators, *i.e.*, socio-technical congruence, as control or independent variables to statistically verify if programming languages, or other development choices, could impact the emergence of community smells.

VI. THREATS TO VALIDITY

This section illustrates the threats to the validity of the study and the way we mitigated them. For their identification, we relied on the well-known work by Wohlin et al. [51].

a) Threats to Construct Validity: Threats in this category refer to the relationship between hypothesis and observations and are mainly due to imprecision in performed measurements [51]. The first threat is related to the dataset chosen to conduct our study. To analyze well-structured resources, for each programming language we mined a well-formed GITHUB open-source projects repository provided by GOOGLEBIGQUERY, a well-structured database that was already used in similar studies [10, 11]. Moreover, to compute community smells and additional metrics, *e.g.*, # commits, we relied on CADOCS [45], which is a published and validated tool.

b) Threats to Conclusion Validity: Threats in this category concern the ability to draw correct conclusions about relations between treatments and outcomes [51]. To ensure the normality of data distributions, we applied the well-known SHAPIRO–WILK test [33]. We computed PEARSON'S CORRELATION INDEX and VIF to verify whether the independent variable and control variables did not suffer from multicollinearity [27]. Moreover, we used the ANOVA test [14] to check the significance of the results. In addition, we consider

the threat on the risk of omitting additional factors that may influence the emergence of community smells. To mitigate it, we reviewed the past literature [15, 18, 24, 25, 46] and identified a set of socio-technical control factors demonstrated impactful in indicating community smells, *e.g.*, *truck factor* [48].

c) Threats to External Validity: Threats in this category are concerned with the generalizability of the results [51]. We built a large dataset containing information about open-source projects on GITHUB, with many contributors. Moreover, we performed a preliminary investigation to identify the most used programming language on the platform, as indicated in Section III-A. Considering that we analyzed 100 Open Source projects, the generalizability in the field of our study might be regarded as satisfactory. However, we plan to conduct a further study and expand the results to a larger sample and more languages. Nevertheless, we plan to improve our study by (1) increasing our dataset with different sources from GITHUB, and (2) performing some qualitative studies—*e.g.*, focus groups, surveys, and interviews [29]. Another possible threat regards the number of contributors per project; the maximum of 50 contributors could not represent an optimal choice in terms of external validity, but we preferred to guarantee consistency in terms of the statistical model, as previously explained.

d) Threats to Internal Validity: Threats in this category are concerned with factors that might have influenced the obtained results [51]. To ensure that other casual and not considered factors influenced the outcome of our study, we implemented a logistic regression taking into account various control variables. Moreover, we evaluated the model using different baselines to study if the dependent variables effectively contribute to the statistical model. A potential threat arises from projects that utilize multiple widely used programming languages. To address this, we considered only the language with the highest usage percentage within each project. We made this decision based on the recognition of one specific language as the project's core language, as acknowledged by the development community. Another possible threat to avoid is related to the similarity between projects selected for the following model construction phase, in particular, by a preliminary analysis, we took into account the number of contributors as the principal projects selection criterion, and we observed that the range from 30 to 50 contributors guaranteed a sub-optimal projects representativeness for each one of the programming languages, so we decided to select only projects in this range of contributors.

VII. CONCLUSIONS

This paper investigates the relationship between the programming language and the emergence of social anti-patterns. Specifically, using statistical analysis, our work aimed to verify whether the choice and use of a particular programming language are correlated with the presence of community smells.

We built a dataset comprising 100 GITHUB projects implemented in 10 different languages. Then, we used the CADOCS tool [45] to extract control metrics and to detect community smells. Our results shed light on the fact that the choice of programming language may not consistently serve as a reliable indicator of the presence or absence of these undesirable patterns. However, one noteworthy exception is the *Prima Donnas effect,* which exhibits a strong statistical correlation with certain programming languages, e.g., PHP.

Building upon the outcomes and discussions presented in Section V, we find it worthwhile to pursue further investigations, including:

1) Replicating our research with a larger dataset and placing a more focused lens on the specific software categories under examination.
2) Keen to explore how the manifestation of community-related issues evolves within projects characterized by hybrid programming paradigms, as exemplified by "ML-Enabled Systems" [21].
3) Conducting a series of qualitative studies—*e.g.*, focus groups and surveys—to strengthen our results using a *mixed-method research approach* [12, 13].

Leaders and project managers can use our results to be aware of potential social anti-patterns and implement—since the initial phases of the project—mitigation and contingency strategies to better perform risk management activities.

ACKNOWLEDGMENTS

This work has been partially supported by the EMELIOT national research project, which has been funded by the MUR under the PRIN 2020 program (Contract 2020W3A5FY).

REFERENCES

[1] "An empirical study on the influence of programming languages on the emergence of community smells — online appendix," 2023. [Online]. Available: https://figshare.com/s/297fced044333134c1b7
[2] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected papers of hirotugu akaike*. Springer, 1998, pp. 199–213.
[3] P. Balestra, *Dummy Variables*. London: Palgrave Macmillan UK, 1990, pp. 70–74.
[4] G. Borchers, "The software engineering impacts of cultural factors on multi-cultural software development teams," in *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE, 2003, pp. 540–545.
[5] F. P. Brooks Jr, *The mythical man-month: essays on software engineering*. Pearson Education, 1995.
[6] K. P. Burnham and D. R. Anderson, "Multimodel inference: understanding aic and bic in model selection," *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.
[7] ——, "Multimodel inference: understanding aic and bic in model selection," *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.
[8] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, and F. Ferrucci, "Gender diversity and women in software teams: How do they affect community smells?" in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2019, pp. 11–20.
[9] G. Catolino, F. Palomba, D. A. Tamburri, and A. Serebrenik, "Understanding community smells variability: A statistical approach," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2021, pp. 77–86.
[10] F. Chatziasimidis and I. Stamelos, "Data collection and analysis of github repositories and users," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2015, pp. 1–6.

[11] ——, "Data collection and analysis of github repositories and users," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2015, pp. 1–6.

[12] J. Corbin and A. Strauss, *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.

[13] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[14] A. Cuevas, M. Febrero, and R. Fraiman, "An anova test for functional data," *Computational statistics & data analysis*, vol. 47, no. 1, pp. 111–122, 2004.

[15] S. R. de Lemos Meira, E. A. Barros, G. S. de Aquino, and M. J. C. Silva, "A review of productivity factors and strategies on software development," in *2010 fifth international conference on software engineering advances*. IEEE, 2010, pp. 196–204.

[16] N. E. Fenton and M. Neil, "Software metrics: roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 357–370.

[17] D. Graziotin, X. Wang, and P. Abrahamsson, "Do feelings matter? on the correlation of affects and the self-assessed productivity in software engineering," *Journal of Software: Evolution and Process*, vol. 27, no. 7, pp. 467–487, 2015.

[18] A. Hernández-López, R. Colomo-Palacios, and Á. García-Crespo, "Software engineering job productivity—a systematic review," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 03, pp. 387–406, 2013.

[19] S. Lambiase, G. Catolino, F. Pecorelli, D. A. Tamburri, F. Palomba, W.-J. Van Den Heuvel, and F. Ferrucci, ""there and back again?" on the influence of software community dispersion over productivity," in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2022, pp. 177–184.

[20] S. Lambiase, G. Catolino, D. A. Tamburri, A. Serebrenik, F. Palomba, and F. Ferrucci, "Good fences make good neighbours? on the impact of cultural and geographical dispersion on community smells," in *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society*, ser. ICSE-SEIS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 67–78.

[21] G. A. Lewis, S. Bellomo, and I. Ozkaya, "Characterizing and detecting mismatch in machine-learning-enabled systems," in *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, 2021, pp. 133–140.

[22] W. Lidwell, K. Holden, and J. Butler, *Universal principles of design, revised and updated: 125 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design*. Rockport Pub, 2010.

[23] P. Mayer and A. Bauer, "An empirical analysis of the utilization of multiple programming languages in open source projects," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '15. New York, NY, USA: Association for Computing Machinery, 2015.

[24] P. Mohagheghi and R. Conradi, "Quality, productivity and economic benefits of software reuse: a review of industrial studies," *Empirical Software Engineering*, vol. 12, no. 5, pp. 471–516, 2007.

[25] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde, "What predicts software developers' productivity?" *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 582–594, 2019.

[26] N. Nahar, S. Zhou, G. Lewis, and C. Kästner, "Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 413–425.

[27] R. M. O'brien, "A caution regarding rules of thumb for variance inflation factors," *Quality & quantity*, vol. 41, no. 5, pp. 673–690, 2007.

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

[29] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "Crowdsourcing user reviews to support the evolution of mobile apps," *Journal of Systems and Software*, vol. 137, pp. 143–162, 2018.

[30] F. Palomba, D. Andrew Tamburri, F. Arcelli Fontana, R. Oliveto, A. Zaidman, and A. Serebrenik, "Beyond technical aspects: How do community smells influence the intensity of code smells?" *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 108–129, 2021.

[31] B. Ray, D. Posnett, P. Devanbu, and V. Filkov, "A large-scale study of programming languages and code quality in github," *Commun. ACM*, vol. 60, no. 10, p. 91–100, sep 2017.

[32] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[33] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[34] N. Shrestha, C. Botta, T. Barik, and C. Parnin, "Here we go again: Why is it difficult for developers to learn another programming language?" *Commun. ACM*, vol. 65, no. 3, p. 91–99, feb 2022.

[35] M. M. M. Syeed and I. Hammouda, "Socio-technical congruence in oss projects: Exploring conway's law in freebsd," in *Open Source Software: Quality Verification*, E. Petrinja, G. Succi, N. El Ioini, and A. Sillitti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 109–126.

[36] D. A. Tamburri, P. Lago, and H. v. Vliet, "Organizational social structures for software engineering," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–35, 2013.

[37] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "Social debt in software engineering: Insights from industry," *Journal of Internet Services and Applications*, 2015.

[38] D. A. Tamburri, R. Kazman, and H. Fahimi, "The architect's role in community shepherding," *IEEE Software*, vol. 33, no. 6, pp. 70–79, 2016.

[39] D. A. Tamburri, F. Palomba, and R. Kazman, "Exploring community smells in open-source: An automated approach," *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 630–652, 2019.

[40] E. Trainer, S. Quirk, C. de Souza, and D. Redmiles, "Bridging the gap between technical and social dependencies with ariadne," in *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, 2005, pp. 26–30.

[41] G. Tripepi, K. Jager, F. Dekker, and C. Zoccali, "Linear and logistic regression analysis," *Kidney international*, vol. 73, no. 7, pp. 806–810, 2008.

[42] G. Valetto, M. Helander, K. Ehrlich, S. Chulani, M. Wegman, and C. Williams, "Using software repositories to investigate socio-technical congruence in development projects," in *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*. IEEE, 2007, pp. 25–25.

[43] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in github teams," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015, pp. 3789–3798.

[44] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[45] G. Voria, V. Pentangelo, A. D. Porta, S. Lambiase, G. Catolino, F. Palomba, and F. Ferrucci, "Community smell detection and refactoring in slack: The cadocs project," in *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2022, pp. 469–473.

[46] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," *arXiv preprint arXiv:1801.06475*, 2018.

[47] Z. Wang and J. Hahn, "The effects of programming style on open source collaboration," 2017.

[48] L. Williams and R. R. Kessler, *Pair programming illuminated*. Addison-Wesley Professional, 2003.

[49] ——, *Pair programming illuminated*. Addison-Wesley Professional, 2003.

[50] C. Wohlin, M. Höst, and K. Henningsson, "Empirical research methods in software engineering," in *Empirical methods and studies in software engineering*. Springer, 2003, pp. 7–23.

[51] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.