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

Giusy Annunziata[1], Carmine Ferrara[1], Stefano Lambiase[1], Gemma Catolino[2], Fabio Palomba[1], Filomena Ferrucci[1], and Andrea De Lucia[1]

[1] University of Salerno, Salerno, Italy
[2] Jheronimus Academy of Data Science, Hertogenbosch, The Netherlands

Abstract. Software development is a complex activity which is not only influenced by technical issues but also by socio-technical patterns among the involved stakeholders—particularly in the open-source context. 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 been investigated community smells from different perspectives, however, how technical project characteristics may impact their emergence is still unknown. Moreover, previous works identified *socio-technical congruence*—i.e., the alignment between a community's technical and social behaviour as a crucial factor in the success rate of software projects. Consequentially, 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 impacting their overall ability to communicate and collaborate, leading to the emergence of social anti-patterns, i.e., community smells. Starting from this hypothesis, we studied the relationship between the most used programming languages and the emergence of community smells in 100 open-source projects on GitHub. Key results of the study show that a correlation exists for some community smells like *Prima Donna Effects*, *Solution Defiance*, and *Organizational Skirmish*, providing managers with a new instrument to make more informed decisions.

Keywords: Software Organizational Structures · Programming Languages · Community Smells · Empirical Studies.

1 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 [6]. Researchers have studied social aspects in software development, also deepening

their relationship with technical factors of a software product. In that sense, of particular interest is the theory of *socio-technical congruence*, which focuses on investigating the alignment between the organizational structure of a community and the software architecture of the developed product [40, 46, 47]. For example, Syeed and Hammouda [40] showed that socio-technical congruence could be directly affected by development choices, *e.g.*, the software product architecture. Moreover, to provide a quantitative and measurable representation of social aspects, the research community defined *community smells*, *i.e.*, social anti-patterns that characterize collaboration and communication in software communities and can lead to social debts [42, 44]. Previous works highlighted that technical development aspects could influence the social aspects: Tamburri et al.[43] observed that development choices, *e.g.*, the adoption of a new development language, are indirectly connected to social debt. Later, Wang and Hahn[52] verified that development style impacts on collaborative aspects of an open-source team. However, it is still unknown how technical development aspects directly impact the emergence of community smell.

Starting from such a consideration and aiming to bridge the above gap, in this paper, we focus on one of the most important technical development choices, *i.e.*, the adopted programming language, and verify if exists a correlation with the emergence of community smells in an open-source development team. Specifically, 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 identified community smells on a set of 100 GitHub open-source projects chosen for the prevailing programming language. We use CADOCS [50] to extract information on 32 socio-technical metrics and the presence of 10 different smells, ranging from Organizational Skirmish to Toxic Communication. Then, we built a logistic regression to analyze the relationship between the most used programming language and the presence of community smells. The main results of our work highlight that a correlation exists between some programming languages and the presence of community smells, *e.g.*, Prima Donnas Smell, *i.e.*, a team member exposes constant disagreement and uncooperativeness [35].

To sum up, our research work resulted in the following three contributions:

1. we provide a dataset containing information on 32 socio-technical metrics and the presence and/or absence of ten community smells of 100 open source projects on the well-known GitHub platform;
2. a logistic regression model that measures the relationship between the most used programming language and the presence of community smells;
3. a publicly available online appendix [3] containing raw data and scripts designed to promote replications and further research on the matter.

2 Background and Related Work

Software development is *de facto* a human-intensive activity involving—most of the time—stakeholders with different backgrounds and interests [6, 14, 39]. Nevertheless, despite this being a well-known fact, research on social and human aspects of software development started growing only recently [41, 43, 44]. Indeed, Tamburri et al. [43] put a fundamental step for such a research field, deepening the concept of *social debt in software engineering*—*i.e.*, the project's additional and unexpected costs deriving from unmanaged or bad-managed social problems between involved stakeholders [43]. Looking at the causes of social debts, *community smells*, *i.e.*, anti-patterns in software development communication and collaboration processes, are among the main causes of their emergence [44]. An example of such smells is the Radio Silence Effect smell that represents the situation in which a stakeholder interposes herself into every formal interaction across more sub-communities with little flexibility to introduce other channels [44].

After defining community smells, several researchers started focusing on providing approaches to detect them within software development teams. Tamburri et al. [45] defined an approach based on network analysis and capable of automatically detecting four community smells, *i.e.*, Organizational Silo, Lone Wolf, Radio Silence, and Black Cloud. On the other side, Almarimi et al. [2] defined an approach based on machine learning that detects more than 10 different community smells and various socio-technical metrics, *e.g.*, centrality. Finally, a most recent and promising approach is CADOCS [50], a recommendation system based on the approach by Almarimi et al. [2], that also provides possible refactoring strategies for ten types of community smells [10], using a conversational agent that communicates with developers on the SLACK platform.

Regarding the factors behind community smells, Catolino et al. [9] studied how team composition correlates with the emergence of community smells, showing that gender diversity might reduce their presence. Later they also considered the factors correlate to the variability of the community smells [11], highlighting that socio-technical factors related to tenure and centrality of a developer can increase the presence of community smells. Finally, Lambiase et al. [24] 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.

Especially important for our work is the study by Valetto et al. [47] showing that product-related software development aspects might largely influence a community's communicative and collaborative aspects. Moreover, Syeed and Hammouda [40] showed how 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 open source projects, *i.e.*, socio-technical congruence. Next, Tamburri et al. [43] observed how technical choices, like introducing a new programming language, can indirectly impact social aspects in a development context. Later, Wang and Hahn [52] showed that specific product factors, i.e., metrics inherent to programming style and lan-

guage, 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. However, these studies analyze communication and collaboration patterns using manual techniques (*e.g.*, manual inspection of source code) or considering a limited set of open-source projects without considering an actual quantitative statistical correlation between them and the most commonly used programming language. Considering all preview results, we conjecture that the features implemented by programming languages, *e.g.*, inheritance mechanisms, might induce the socio-technical alignment and congruence of the development community, possibly impacting their overall needs to communicate and collaborate.

We have been inspired by previous literature focusing on the impact of social factors on community smells [9, 11, 24], considering new product metrics. In particular, we investigated whether a correlation exists between the most used programming language and the presence of community smells. Although previous studies on community smells [9, 24] relied on the same dataset, we performed our study on a newly created dataset—consisting of 100 open-source projects on GitHub—as a complementary contribution of this work.

3 Research Study Design

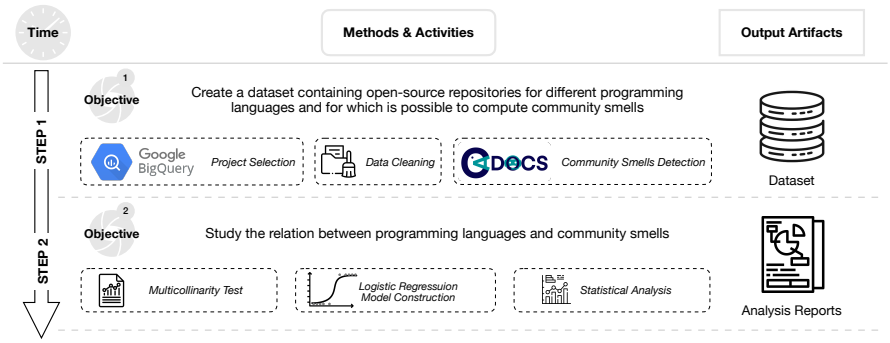

Fig. 1: Overview of our research study design.

This study aims to analyze whether the usage 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 *purpose* 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 study asks the following research question:

***RQ**: To what extent does the used programming language relate to social anti-patterns?*

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 [44]. The rationale behind this choice relied on previous literature Wang and Hahn [52] that analyzed the impact of programming style and programming language inconsistencies on open-source software collaboration, demonstrating 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 by community smells. Figure 1 shows an overview of the steps described below.

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 languages used, (2) factors previously found impactful for indicating community smells [42], 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. To conduct our analysis, we employed the guidelines by Wohlin et al. *et al.* [55] and we followed the *ACM/SIGSOFT Empirical Standards*. Specifically, we employed the *"General Standard"* and *"Data Science"* definitions and guidelines.[3]

3.1 Step$_1$ - Build a Dataset

To conduct our study, we created a dataset by consulting the existing databank offered by GoogleBigQuery,[4]. It is a Restful Web Service that allows

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

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

interactive analysis of a large database that works with Google Storage. Moreover, it also contains the GitHub Archive,[5] *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. We selected the first 10 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. [36]—A summary table with the 10 languages and their characteristics is available in the online appendix [3]. For each of the 10 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 (more details are discussed in the threat to validity section). After pulling the projects, we detected community smells using the CADOCS [50] tool, which extracts information about the presence of ten different community smells— described in the online appendix [3]— and additional metrics, *e.g.*, # commits, for constructing our statistical models. Further details are provided in the following section.

3.2 Step$_2$ - Building a Statistical Model

After data extraction, we constructed a logistic regression model for each community smell considered by CADOCS to understand the relationship between a particular programming language. Each model contains all the projects considered, and the presence or absence of community smells.

Independent Variable In the context of our study, we considered the programming language of the project as the independent variable—considering that software can be made with multiple languages, we assume the most used one. Therefore, we selected the 10 languages most used (3.1). Moreover, since the relative variable is expressed in the dataset as categorical, we replaced it with a *dummy variable* before the logistic regression models were created [5].

Response Variable Since we aimed to understand the impact of adopting a specific programming language on the *presence/absence of a specific community smell*, we encoded this factor as the Response Variable. According to CADOCS [50] tool's output data representation, for each community smells, our response variable assumed value 1 when it impacted a development project, 0 otherwise.

[5] `https://www.gharchive.org`

Control Variables To well design a statistical model, a good 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 [9, 11, 20, 24, 33, 48] 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;
2. **Number of Contributors** - the number of contributors that performed at least one commit on the project repository. In most cases, the greater the number of contributors, the greater the likelihood of community smells; Catolino et al. [9] demonstrated that this factor could influence the presence of community smells;
3. **Active Days** - the number of active days of the project repository, namely the number of days at least one contributor works on the project. In most cases, the longer the activity time, the more likely it is that smells are present;
4. **Bus Factor Number** - specific socio-technical number that estimates the minimum number of team members that suddenly abandon the project before it fails due to the lack of experienced personnel [4, 19, 54].

Like for detecting community smells, we also used CADOCS [50] to extract the above control factors.

Statistical Model Construction For the model construction, we relied on the functions `logit` available in the Python package `Stats Model` [37]. Based on similar studies [18, 23, 27], we analyzed multicollinearity [31] that can affect the reliability of the results. For this reason, we measured the Pearson's Correlation Index and the Variance Inflation Factor (VIF) to verify the absence of multicollinearity between the independent variables. We set a maximum threshold of 5 for identifying independent variables that are free from multicollinearity. Should any variable exceed this limit, it will be excluded from the logistic regression models [32]. [6]

Finally, we constructed 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) [1, 7] estimators, that are widely used as quality prediction criteria of the statistical models [8]. 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.

[6] We relied on their implementation provided by the Python packages `SciPy` [49] and `Stats Model` [37].

4 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 analysis.

Table 1: Community Smell detected by CADOCS [50]

Language	OSE	BCE	PDE	SV	OS	SD	RS	TFS	UI	TC
C	6	7	9	8	3	8	1	3	4	2
C#	2	3	6	8	3	7	3	5	5	5
C++	4	4	9	8	4	7	1	5	2	5
Go	7	10	9	6	3	9	4	2	4	0
Java	4	7	10	5	6	9	1	4	5	2
JavaScript	4	8	10	8	4	9	3	5	6	3
PHP	6	7	10	3	4	10	2	5	4	1
Python	5	6	10	10	2	7	3	3	6	3
Scala	4	7	8	6	3	10	4	7	5	4
TypeScript	3	3	9	8	3	8	4	4	5	2
Total	45	62	90	70	35	84	26	43	46	27

all languages : $x/10$ − *total* : x:/100

4.1 Step$_1$ - Dataset Information and Composition

Firstly, we obtained an initial collection of 34678 open-source projects from GoogleBigQuery. Then, we 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). It is worth observing that the statistics obtained by CADOCS refer to the time window between *January and February 2022*. Finally, we analyzed the distribution of community smells for each programming language taken into consideration; Table 1 shows that there is a significant 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 [3] we provided available specific reports about the number of repositories detected by GoogleBigQuery for each selected programming language.

Table 2: Results for the Logistic Regression Models

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.
Independent Variable: Programming Language (Shown as dummy variable)																				
C	-7.2367	—	-3.7332		-3.9603		26.1533	*	-0.7313		-2.7477		-8.7687	*	3.6091		0.3463		-5.0171	
C#	-7.5802	—	-4.6323		-2.3561		26.4524	*	-1.8108		-4.6227		-9.4057	—	4.5316		0.5379		-4.1264	
C++	-7.5802	—	-4.2784		-4.4385		28.7233	*	-1.5321		-3.6058		-9.1823	—	4.425		-1.3717		-4.0251	
Go	-7.7179	—	22.6691		-0.6004		28.5026	*	-2.1065		-2.2358		-8.8448	—	3.5287		0.8926		-42.4639	
Java	-6.7351		-3.2383		-3.161		54.9637		-2.985		-3.6202		-8.6561	—	4.544		0.8949		-5.3535	
JavaScript	-7.4319	—	-2.7778		-1.6205		51.2172		-1.1944		-3.5658		-8.5307	—	4.6381		1.785		-4.3391	
PHP	-7.3879	—	-3.4418		-2.0122		50.4452		-3.5614		-2.6492		12.6902		4.7287		0.8523		-5.588	
Python	-8.7814	—	-3.9184		-1.5113		53.9787		30.0797		-3.2309		-10.0661	*	4.0935		1.4146		-4.672	
Scala	-8.1204	—	-3.681		-1.1747		28.6245	*	-2.3272		-3.7419		23.8846		5.8586		1.1414		-4.095	
TypeScript	-7.8568	—	-4.772		-0.6522		28.3731	*	-1.5101		-3.7954		-8.8964	—	4.2139		1.119		-4.8052	
Control Variables																				
# of commits	0.0001		-0.0008	***	0.0005	*	0.0005		0.0004		0		-0.0006	*	0.0003	—	0.0004	*	0.0004	*
# Contributors	0.1808		0.0866		-0.0346		-0.7059	*	0.0746		0.0655		0.2648	*	-0.1252		-0.0488		0.065	
Active Days	0		0.0008	**	0.0005	—	0		-0.0002		0.0003		0.0006	—	-0.0002		0.0002		0.0003	
Bus Factor #	0.5242		0.6953		0.5331		33.2888		-0.2402		0.1107		24.1738		0.751		-0.9971		-0.4009	

***:p<0.001; **:p<0.01; *:p<0.05; −:p<0.1

👉 Step$_1$: summary of the results.

The main artifact 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.

4.2 Step$_2$ - Statistical model

Table 2 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 build used both independent and control variables. In particular, the language variable is represented as a dummy variable [5], 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 specific community smells, *i.e.*, Organizational Skirmish, Prima Donnas Effect, and Solution Defiance, it is possible to observe a low statistically significant correlation with the majority of the specific programming languages we included in the original dataset. Overall, the variability of the adopted programming language has a partial impact on correlating with the presence of community smells. As proof of the low statistical significance of the independent variable, the values assumed by AIC and BIC are similar both with and without the programming language variables as the community smells inducing factor. As further proof of the goodness of our statistical models, it can be observed that in no case is the value of VIF ever greater than 1.5 for each of the independent variables.

By analyzing our results, we may assume that the type of projects analyzed could be one of the origins of the correlation between programming languages and the variables we consider. Specifically, Mayer and Bauer already demonstrated that the programming language often depends on the type of software that we want to develop [26]—*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. To better analyze this, we conducted a manual inspection of all 100 projects in our dataset, finding that most of them were identifiable as *Development Support*—*i.e.*, research projects in which the final output turns out to be, for example, 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 community smells. Indeed, further research considering the correlation between the type of project and team and the community smells should be conducted to shed light on such a correlation.

☞ Step$_2$: summary of the results.

Our main findings show that *the programming languages* are not always correlated to the presence of community smells in an open Source Development Process. Some exception occurs for Community Smells *Organizational Skirmish, Prima Donnas Effect*, and *Solution Defiance*, for which there is a slight statistical correlation with different programming languages.

5 Discussion and Implications

In this section, we start from the findings reported in Section 4 and discuss some implications for future research on the matter.

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 1—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 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 emphasized 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 with a constant presence of community smells in the projects and with a statistically significant correlation, we found a subset of sector-specific languages such as Python and JavaScript. An interesting context to explore could be the one of *Machine Learning Enabled* [22, 25] software products characterized by both a classical component and a machine learning one—and the related quality and community aspects. In fact, 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, data engineers, and software engineers—working together [30]. Such heterogeneity could lead to conflicts between the various subgroups, potentially resulting in 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 [38].

The impact of socio-technical factors. Our study reinforces what has already been shown regarding socio-cultural metrics [45]. The number of committers was also found in our study to have high statistical significance in correlation

to the dependent variable. 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 those factors, our study could be enhanced by including other socio-technical indicators and metrics, *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.

6 Threats to Validity

This section illustrates the threats to the validity of the study and the way we mitigated them—according to Wohlin et al. [56] guidelines.

6.1 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 [56]. 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 [12, 13]. Another possible threat to avoid is related to the similarity between projects selected for the model construction phase; in particular, in the preliminary analysis, we considered the number of contributors as the principal projects selection criterion. We observed that the range from 30 to 50 contributors guaranteed sub-optimal project representativeness for each programming language, so we selected only projects in this range of contributors. Moreover, to compute community smells and additional metrics, *e.g.*, # commits, we relied on CADOCS [50], which is a published and validated tool. To conduct the analysis, we focused only on the 10 most used programming languages, as they provide a greater probability of capturing a representative sample of projects.

6.2 Threats to Conclusion Validity

Threats in this category concern the ability to draw correct conclusions about relations between treatments and outcomes [56]. To address any threat, we computed Pearson's Correlation Index and VIF to verify whether the independent variable and control variables did not suffer from multicollinearity [31]. This threat regarded the risk of omitting additional factors that could influence the emergence of community smells. To mitigate this, we reviewed the past literature [17, 21, 28, 29, 51] and identified a set of socio-technical control factors demonstrated impactful in indicating community smells, *e.g.*, *truck factor* [53].

6.3 Threats to External Validity

Threats in this category are concerned with the generalizability of the results [56]. We built a large dataset containing information about open-source projects on GitHub, with a large number of contributors. Moreover, we performed a preliminary investigation to identify the most used programming language on the platform. Considering that we analyzed 100 Open Souce projects, the generalizability of our study might be considered satisfactory. 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 [34]. Another possible threat regards the number of contributors per project, in particular, the maximum of 50 contributors could not represent an optimal choice in terms of external validity, but we preferred to guarantee consistency in term of the statistical model as previously explained.

7 Conclusions

Our study on the relationship between the programming language and the emergence of social anti-patterns highlights that the programming language does not always represent a good indicator for the presence or absence of such smells. The most notable exception is Prima Donnas effect, whose prediction model shows a statistical correlation for a set of programming languages. In addition, although less significant, also the Community Smells Solution Defiance and Organizational Skirmish are found to correlate. 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.

According to the results achieved by the study and the considerations reported in Section 5, we designed our future research agenda. First, we plan to replicate the study using a larger dataset and making more focus on the type of software analyzed to going deepen the relationship that emerged in this study. Second, we want to investigate how the emergence of community smells changes in projects characterized by hybrid programming paradigms like *ML-Enabled Systems* [22, 25]—As suggested by discussion point *Programming language vs. community smells* in section 5. Last but not least, we are going to conduct a series of qualitative studies—*e.g.*, focus groups and surveys—to strengthen our results using a *mixed-method research approach* [15, 16].

Acknowledgments

Fabio is partially supported by the Swiss National Science Foundation - SNF Project No. PZ00P2_186090. 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).

Bibliography

[1] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected papers of hirotugu akaike*. Springer, 1998, pp. 199–213.

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

[3] G. Annunziata, C. Ferrara, S. Lambiase, G. Catolino, F. Palomba, F. Ferrucci, and A. De Lucia, "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

[4] G. Avelino, L. Passos, A. Hora, and M. T. Valente, "A novel approach for estimating truck factors," in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. IEEE, 2016, pp. 1–10.

[5] P. Balestra, *Dummy Variables*. London: Palgrave Macmillan UK, 1990, pp. 70–74.

[6] F. P. Brooks Jr, *The mythical man-month: essays on software engineering*. Pearson Education, 1995.

[7] 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.

[8] ——, "Multimodel inference: understanding aic and bic in model selection," *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.

[9] 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.

[10] ——, "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*, 2020, pp. 25–34.

[11] 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.

[12] 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.

[13] ——, "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.

[14] S. Cherry and P. N. Robillard, "Communication problems in global software development: Spotlight on a new field of investigation," in *International Workshop on Global Software Development, International Conference on Software Engineering, Edinburgh, Scotland*. IET, 2004, pp. 48–52.

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

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

[17] 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.
[18] N. E. Fenton and M. Neil, "Software metrics: roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 357–370.
[19] M. Ferreira, M. T. Valente, and K. Ferreira, "A comparison of three algorithms for computing truck factors," in *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. IEEE, 2017, pp. 207–217.
[20] 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.
[21] 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.
[22] J. Horkoff, "Non-functional requirements for machine learning: Challenges and new directions," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 386–391.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] H. Midi, S. K. Sarkar, and S. Rana, "Collinearity diagnostics of binary logistic regression model," *Journal of interdisciplinary mathematics*, vol. 13, no. 3, pp. 253–267, 2010.
[28] 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.
[29] 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.
[30] 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.
[31] R. M. O'brien, "A caution regarding rules of thumb for variance inflation factors," *Quality & quantity*, vol. 41, no. 5, pp. 673–690, 2007.

[32] ——, "A caution regarding rules of thumb for variance inflation factors," *Quality & quantity*, vol. 41, pp. 673–690, 2007.
[33] 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.
[34] 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.
[35] 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.
[36] 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.
[37] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.
[38] 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.
[39] I. Sommerville, *Software Engineering*, 10th ed. Pearson College Div, 8 2015.
[40] 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.
[41] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "What is social debt in software engineering?" in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2013, pp. 93–96.
[42] 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.
[43] 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.
[44] 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.
[45] 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.
[46] 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.
[47] 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.
[48] 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.

[49] 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.

[50] 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.

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

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

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

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

[55] 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.

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