What Public Repositories Reveal: Inferring Developer Skills from GitHub

Gustavo Vitor da Silva
Federal University of Lavras (UFLA)
Institute of Sciences, Technology and Innovation (ICTIN)
São Sebastião do Paraíso-MG, Brazil
gustavo.silva39@estudante.ufla.br

Enzo Augusto Valentim
Federal University of Lavras (UFLA)
Institute of Sciences, Technology and Innovation (ICTIN)
São Sebastião do Paraíso-MG, Brazil
enzo.valentim@estudante.ufla.br

Johnatan Oliveira
Federal University of Lavras (UFLA)
Department of Computer Science (DCC)
Lavras-MG, Brazil
johnatan.oliveira@ufla.br

Abstract

Public software repositories record evidence about development activities, but they represent only part of a person's trajectory. This article presents SSM (*Source Signal Miner*), a method that organizes public GitHub data into five observable indicators: public participation, textual polarity, technological diversity, contribution continuity, and public activity intensity. The method also assigns a predominant technical area and generates a report associating the results with the amount and coverage of the evidence used. These indicators describe patterns of public activity and do not constitute direct measures of competence, productivity, or professional performance. The evaluation was conducted through two complementary studies. A perceptual study with nine Information Systems students analyzed reports produced from their own profiles. Separately, the technical classification component was evaluated using a labeled dataset of 1,240 real profiles, from which 372 were reserved for the final test. The mean perceived adherence of report components, on a normalized scale from 0 to 1, ranged from 0.63 to 0.73. The ability of the report to represent content available in the public profile received a mean score of 0.76, while the ability of the public profile itself to represent complete experience received a mean score of 0.42. In the independent test, the classifier achieved an accuracy of 88.17% and a macro F1-score of 0.88. Results indicate that public signals can support an initial characterization, provided that limitations in data coverage are made explicit and results are combined with other sources of information.

Keywords

Software Repository Mining, Developer Characterization, GitHub, Software Analysis, Software Maintenance, Evidence Provenance

1 Introduction

Software maintenance and evolution activities often depend on the identification of people with experience in specific components, technologies, or practices [19]. Information about previous activities of developers can support expert search, reviewer recommendation, and task assignment [1, 13, 19]. However, this experience may be distributed across different projects, organizations, and platforms, which makes its retrieval and analysis more difficult.

Public repositories provide a complementary source of evidence about these activities. Languages, dependencies, commits, issues, pull requests, reviews, and messages record technical, temporal, collaborative, and textual aspects. These records have been used in studies on familiarity with technologies, technical roles, and participation in software projects [11, 12, 14, 15]. However, these records represent only part of the experience of a developer. Activities may occur in private repositories or on platforms other than GitHub, while metrics such as commits and changed lines depend on project practices and the presence of automated content [10]. Therefore, the absence of public activity does not imply the absence of experience, and a larger volume of records does not, by itself, represent greater competence, productivity, or professional quality.

Previous studies use development histories to identify experts, recommend people, or classify technical roles [4, 14, 15]. In general, these approaches produce a recommendation, score, or class for a specific purpose. However, aggregated results may make it difficult to identify the evidence used and may hide situations in which the data source provides limited coverage.

This work investigates how heterogeneous signals from public repositories can be organized into an initial characterization that preserves information about their origin, amount, and coverage. For this purpose, this article presents SSM (*Source Signal Miner*), a method that collects public evidence from GitHub and produces five observable indicators: public participation, textual polarity, technological diversity, contribution continuity, and public activity intensity. The method also assigns a predominant technical area based on the technologies identified.

The indicators are presented separately in an individualized report, together with the technical area and information about the evidence used. SSM does not produce an overall score and does not aim to directly measure competence, productivity, or professional performance. The evaluation was conducted through two complementary studies. The first was an exploratory study with nine Information Systems students, who evaluated concepts related to developer characterization and later inspected reports constructed from their own profiles. The second study independently evaluated the technical area classification using a labeled dataset of 1,240 real GitHub profiles, from which 372 were reserved for the final test. The responses of the participants were not used as reference data to calculate classifier performance.

In the perceptual evaluation, normalized adherence scores ranged from 0.63 to 0.73. The report represented the public profile more strongly (mean 0.76) than the profile represented complete participant experience (mean 0.42). In the independent test, the classifier achieved 88.17% accuracy and a macro F1-score of 0.88. Open-ended responses highlighted incomplete public coverage, hybrid profiles, inaccurate technology attribution, and insufficient evidence.

The main contributions of this work are: (i) SSM, which integrates technical, temporal, collaborative, and textual signals into

a characterization based on public evidence; (ii) a presentation approach that keeps the indicators separated and associated with the origin, amount, and coverage of the data; and (iii) an empirical evaluation that combines developer perceptions, qualitative analysis of profile limitations, and an independent evaluation of the technical classification. The results provide exploratory evidence about the feasibility of organizing public traces to support an initial characterization of developers.

2 Related Work

Development histories have been used to identify people with knowledge about specific system parts. The Expertise Browser, e.g., associates developers with the artifacts with which they have greater familiarity [13]. Other approaches use authorship, code changes, and project participation to recommend defect assignees, collaborators familiar with components, or code reviewers [1, 5, 9, 19]. In general, these solutions analyze project or organizational history and produce recommendations for specific tasks.

The availability of public repositories expanded these analyses to activities distributed across different projects. Montandon et al. [14] used GitHub data to identify developers associated with libraries and *frameworks* and later classified them into technical roles, such as *backend*, *frontend*, and *mobile* [15]. Del Vescovo et al. [4] also constructed profiles based on languages, repositories, and contribution histories. These studies provide the foundation for the technical classification component adopted by SSM.

Other studies investigate participation forms in *open source* communities. Liang et al. [11, 12] analyzed issues, pull requests, and other collaboration mechanisms, while recurrence, persistence, and temporal distribution data have been used to study participation continuity [2, 21]. Commit and code change metrics have also been linked to activity and productivity perceptions [17].

However, these records depend on platform coverage and usage practices. Commit volume may be influenced by project organization, automated operations, or version control practices, while the absence of public contributions may result from activities in private repositories or on other platforms [10]. For this reason, SSM uses these data only to describe publicly observable patterns.

Textual interactions constitute another source of evidence. Previous studies applied sentiment analysis to commit messages, issues, and discussions [6, 18]. However, general-purpose tools may produce inconsistent results in the context of Software Engineering [3, 8, 16]. Linguistic and technical characteristics of the messages may affect the calculated polarity. In SSM, this information represents only the polarity of the available textual content, and not communication, collaboration, or professional behavior.

Previous studies mainly focus on people recommendation, expert identification, technical role classification, or analysis of specific forms of participation. SSM differs by integrating technical, temporal, collaborative, and textual signals into an initial characterization of the public profile, while keeping the indicators separated and associated with the origin, amount, and coverage of the evidence.

3 SSM: Developer Characterization

SSM collects public artifacts from GitHub and organizes them into five observable indicators and one predominant technical area. The results are consolidated into a report that presents each component separately, together with the amount of evidence used. When the available data are insufficient or a collection step cannot be completed, this condition is explicitly reported instead of being interpreted as a negative result.

3.1 Collection and Indicator Construction

SSM uses public information retrieved through the GitHub GraphQL API, including repository metadata, languages, dependencies, commits, issues, pull requests, reviews, and messages. These records are organized into technical, temporal, collaborative, and textual signals. The same event may contribute to more than one category, depending on the aspect being considered. Each signal remains associated with the profile, repository, artifact type, and, when available, the time of the event. Collection failures and missing artifacts are recorded separately, preventing unavailable data from being interpreted as negative evidence.

From these signals, SSM produces five indicators in the $[0, 1]$ range: public participation, textual polarity, technological diversity, contribution continuity, and public activity intensity. Table 1 presents, for each indicator, the criteria used in its construction and the limits of its interpretation. The second column describes how public records are aggregated, while the third column defines what can be concluded from the result. This distinction is necessary because numerically higher values may represent a greater amount of observed activity without indicating, for example, greater competence, productivity, or collaboration ability.

3.2 Technical Area Classification

In addition to the indicators, SSM assigns one predominant technical area among six classes: *backend*, *frontend*, *fullstack*, *mobile*, *cloud/devops*, and *data science*. These classes are based on roles from previous GitHub studies [15]. The categories do not cover all professional activities and may overlap in hybrid profiles.

Each profile contains the five most frequent languages and the five most frequent libraries or frameworks. Each language's volume is discretized into *zero*, *low*, *medium*, *high*, and *very high*, producing a fixed-size categorical representation.

The classifier produces a class of most likelihood, accompanied by the main technologies identified. This decision simplifies the presentation, but it may provide an incomplete representation of developers who work in multiple areas. The labeled dataset and the evaluation protocol are presented in Section 4.

4 Evaluation Steps

The evaluation of SSM was organized into two complementary studies. The first consisted of a perception study with Information Systems students, investigating their views about the evidence available on GitHub, the perceived adherence of the report components, and the limitations of the characterization. The second study evaluated the predictive performance of the technical area classification using a labeled dataset of real profiles collected from GitHub. The following subsections describe the execution process and the results of each evaluation stage.

Table 1: Operationalization of the observable indicators of SSM.

Indicator	Construction	Interpretation limited to
Public participation	Mean of the proportion of closed issues, the proportion of merged pull requests, and commit activity, normalized with saturation at 200. Issues and pull requests created and resolved by the same user may also contribute to the result	Participation in the public artifacts analyzed, not necessarily engagement in an *open source* community
Textual polarity	Application of VADER [7] to up to 50 recent discussions and 20 comments per repository, with translation into English and aggregation of scores by repository and profile	Polarity of the processed content, not communication, collaboration, emotional state, or professional behavior
Technological diversity	Mean of the subscores for distinct languages, with saturation at eight; introduction of new languages, with saturation at five transitions; and topic entropy, limited to three bits	Publicly observed technological variety, not technical expertise or adaptability
Contribution continuity	Proportion of three satisfied criteria: a minimum interval of six months between the first and last contribution, activity in at least three months, and contributions in at least 15% of the observed weeks	Persistence of the observed public activity, not professional continuity
Public activity intensity	Mean of the subscores for changed lines, commits, and the longest sequence of active weeks, with saturation at 50,000 lines, 200 commits, and ten weeks. The analysis considers up to 15 repositories per profile	Volume and frequency of public records, not productivity or quality; profiles with activity distributed across a larger number of projects may be underrepresented

4.1 Research Questions

The evaluation was guided by the following questions:
RQ1. How do the participants perceive the presence of evidence on GitHub and the adherence of the components presented by SSM?
RQ2. What is the predictive performance of the technical area classification on an independent set of real profiles?
RQ3. Which factors limit the representativeness of profiles constructed from public GitHub evidence?

RQ1 covers the perceptions collected before the presentation of the reports and the perceived adherence after their inspection. RQ2 addresses only the technical classification, while RQ3 uses the open-ended responses to identify limitations related to platform coverage and to the design decisions adopted in SSM.

4.2 Perception Study

The accessible population consisted of 50 fifth-semester Information Systems students enrolled in a Software Engineering course. The entire class was invited, without selection based on academic performance, professional experience, or GitHub usage. Recruitment was non-probabilistic, and nine students participated voluntarily.

Before data collection, the participants received information about the study objectives, the data used, and the data processing procedures. They could withdraw from the study or request the removal of their data without academic consequences. GitHub identifiers were temporarily used to associate responses with reports and were later replaced with codes. The results were presented in aggregated form, and the comments were reviewed to avoid identification of the participants.

The study was conducted in two stages between May 24 and June 10, 2026. First, the participants evaluated five concepts regarding their importance for developer characterization, their expected presence in public activities, and the suitability of GitHub as a source of evidence. Next, they inspected reports constructed from their own profiles and evaluated the adherence of the components, the representativeness of the profile and the report, and the compatibility of the assigned technical area. Open-ended questions allowed the participants to report experiences that were not represented, incorrect assignments, coverage limitations, and difficulties in interpreting the results.

4.3 Construction of the Labeled Dataset

The profiles used in the predictive evaluation were collected from GitHub in January 2026. The collection process sought to obtain heterogeneous users, without previous restrictions related to language, technology, location, or technical area. The initial selection used pseudo-random positions in the user space accessible through the API, with different starting points to reduce dependence on queries ordered by popularity. Since GitHub does not provide a complete sampling structure for all active developers, this procedure does not constitute a representative probabilistic sample of the platform.

Eligible profiles represented individual users, contained at least one retrievable public repository, provided sufficient information to construct the classifier attributes, included a public indication of a professional area that could be used as a reference label, and were not explicitly identified as bots or automation accounts. Supervised evaluation requires both technical input attributes and a reference area. Therefore, duplicate profiles, organizations, automated accounts, users without sufficient public data, and cases for which the technical area could not be determined were excluded.

After applying these criteria, the final dataset contained 1,240 real profiles. The six areas defined in Subsection 3.2 were considered: *backend*, *frontend*, *fullstack*, *mobile*, *cloud/devops*, and *data science*. The initial label was obtained from the area declared by the user in the GitHub description and, when available, from publicly linked personal pages and professional profiles. The use of declarative sources aimed to avoid circularity, since repository technologies constitute the predictive attributes of the model.

Profiles associated with more than one area were labeled according to the activity presented as predominant in the self-description. Cases without an identifiable predominant area were considered ambiguous and were removed. The labeling process was conducted by two evaluators. Disagreements were discussed until consensus was reached based on the public sources associated with each profile.

Table 2 presents the distribution of the 1,240 profiles among the six technical areas. The dataset is not uniformly distributed. *Backend* is the most frequent class, with 303 profiles (24.4%), while *Fullstack* is the least frequent, with 100 profiles (8.1%). Together, *Backend*, *Mobile*, and *Cloud/DevOps* represent approximately 68.8% of the profiles. This difference in class frequencies reinforces the

need to analyze metrics by class and to use the macro F1-score as the main aggregated measure.

Table 2: Distribution of profiles by technical area.

Area	Profiles
Backend	303
Cloud/DevOps	270
Data Science	137
Frontend	150
Fullstack	100
Mobile	280
Total	1,240

4.4 Classifier Training and Evaluation

The 1,240 profiles were divided through stratified sampling, with 70% assigned to training and 30% assigned to testing, resulting in 868 and 372 profiles, respectively. The test partition was defined before synthetic data generation and was not used during training, augmentation ratio selection, or model configuration selection. The training set was augmented with categorical instances produced by GANBLR [22]. GANBLR was selected because it generates categorical data and represents dependencies among attributes through a Bayesian structure.

Augmentation ratios equivalent to 25%, 50%, 100%, 200%, and 400% of the training set were evaluated. The configurations were compared through stratified five-fold cross-validation applied only to the training partition. The macro F1-score was used as the main selection criterion because it assigns the same weight to all six classes. In each fold, GANBLR was fitted only with the corresponding training subset, while the validation subset remained composed exclusively of real profiles.

The selected configuration added two synthetic instances for each real instance, corresponding to an augmentation of 200%. Synthetic data generation used pseudo-random seed 32 to support reproducibility. After configuration selection, the classifier was refitted using the complete training partition and evaluated once on the 372 real profiles in the test set. Accuracy, precision, recall, and F1-score were calculated for each class, together with macro and weighted averages. The macro average was used as the main measure, while the weighted average was presented as a complementary measure that considers class frequency.

The augmented classifier was compared with the same estimator trained only with the 868 real profiles. This comparison describes the behavior of the configurations on the partition used in the experiment, but it does not isolate the causal effect of GANBLR or demonstrate its superiority on other datasets, partitions, or random seeds.

5 Results and Discussion

5.1 RQ1: Perceptions of the Evidence and the Report

This subsection addresses RQ1: How do the participants perceive the presence of evidence on GitHub and the adherence of the components presented by SSM?

RQ1 considered two distinct moments: the general perceptions of the participants before the presentation of the reports and the adherence of the components after report inspection. Since these evaluations represent different objects, no differences were calculated between the two moments.

Figure 1 presents these two perspectives. Panel (a) presents the evaluations conducted before the report, comparing the importance attributed to the concepts, the expected manifestation of these concepts in public activities, and the suitability of GitHub as a source of evidence. The main contrast appears between the high importance attributed to the concepts and the more cautious evaluation of the ability of the platform to represent them, particularly for aspects related to emotional management, interaction, and adaptability. Panel (b) presents the perceived adherence after report inspection. The means of the five components remain close, without a strong predominance of a single component. However, the distribution of individual responses shows that adherence varied among the participants, indicating that aggregated results may hide divergent individual evaluations.

Perceptions before the report. The Friedman tests confirmed the heterogeneity observed in Panel (a), indicating global differences among the concepts regarding perceived importance (adjusted p = 0.003), expected public manifestation (adjusted p = 0.007), and suitability of GitHub as a source of evidence (adjusted p = 0.044), with moderate to large effect sizes.

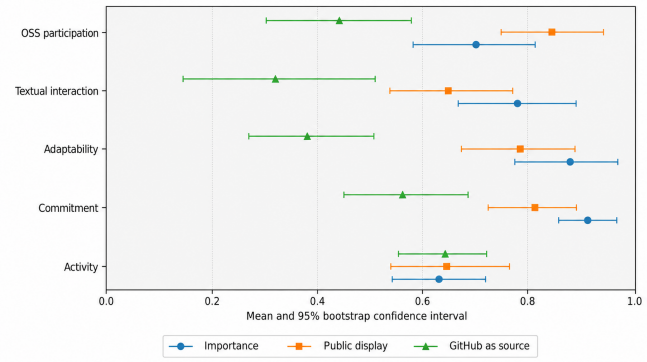
Commitment and adaptability received the highest mean importance scores, while participation in *open source* projects presented the highest expected public manifestation. Regarding platform suitability, activity and commitment received the highest evaluations, while emotional management, interaction, and adaptability were considered less observable through GitHub.

No pairwise comparison remained significant after the Holm correction. Therefore, although the results indicate global differences, it is not possible to state conclusively which pairs of concepts differ. Even so, the observed pattern shows that the importance attributed to a concept does not imply that GitHub is considered an equally suitable source for representing it.

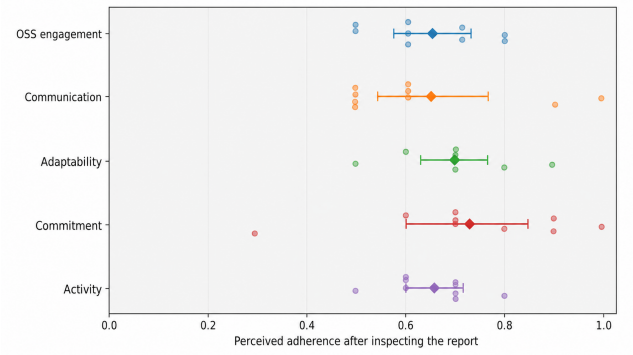
Adherence after report inspection. As presented in Panel (b), the mean adherence scores ranged from 0.63 to 0.73. Commitment presented the highest perceived adherence, while communication in public repositories presented the lowest. The proximity among the means indicates that no component was clearly predominant.

Adherence also presented a global difference among the components (adjusted p = 0.044), with a moderate effect size, but no pairwise comparison remained significant after correction. The dispersion of the individual responses recommends caution when interpreting the aggregated values.

These labels represent broader concepts than the indicators implemented by SSM. Textual polarity is not equivalent to communication ability, technological diversity is not equivalent to adaptability, and contribution continuity does not constitute a direct measure of commitment. Therefore, the results represent perceived adherence and not construct validation.



(a) Perceptions before the report



(b) Adherence after report inspection

Figure 1: Results of the perception study with nine participants.

5.2 RQ2: Technical Classification Performance

This subsection addresses RQ2: What is the predictive performance of the technical area classification on an independent set of real profiles?

In the independent test set, the classifier correctly identified 328 of the 372 profiles, corresponding to an accuracy of 88.17%, with a 95% Wilson confidence interval ranging from 84.5% to 91.1%. Table 3 details the performance for each technical area. In the table, Prec. represents precision, Rec. represents recall, and n represents the number of profiles in each class.

Three patterns deserve attention. First, *Mobile* presented the highest F1-score and a strong balance between precision and recall. Second, *Frontend* achieved maximum recall but the lowest precision, indicating that all profiles from this class were identified, although profiles from other areas were also classified as *Frontend*. Third, *Backend* presented the opposite behavior: high precision and lower recall, suggesting more conservative classifications for this area.

Table 3: Performance of the technical classification on the test set.

Area	Prec.	Rec.	F1	n
Backend	0.97	0.78	0.87	91
Cloud/DevOps	0.96	0.85	0.90	81
Data Science	0.78	0.95	0.86	41
Frontend	0.66	1.00	0.80	45
Fullstack	0.93	0.90	0.92	30
Mobile	0.96	0.92	0.94	84
Macro	0.88	0.90	0.88	–
Weighted	0.91	0.88	0.88	–

The F1-score ranged from 0.80 for *Frontend* to 0.94 for *Mobile*. *Fullstack* and *Cloud/DevOps* also presented balanced results, while *Data Science*, similarly to *Frontend*, presented higher recall than precision. This pattern indicates a greater ability to retrieve profiles from these areas, accompanied by a higher occurrence of false positives. The macro and weighted averages produced the same F1-score of 0.88. Since the macro average assigns the same weight to all classes and the weighted average considers class frequency, this proximity indicates that the aggregated performance was not determined only by the areas with the largest number of profiles.

The classifier trained only with the 868 real profiles obtained a macro F1-score of approximately 0.56 on the same test set, while the configuration with synthetic augmentation achieved 0.88. This difference is compatible with a benefit from synthetic data generation in the evaluated configuration, but it does not allow the improvement to be causally attributed to GANBLR, since multiple partitions, random seeds, and alternative balancing strategies were not examined.

5.3 RQ3: Coverage and Representativeness Limitations

This subsection addresses RQ3: Which factors limit the representativeness of profiles constructed from public GitHub evidence?

The ability of the public profile to represent the complete experience of the participant received a mean score of 0.42, the lowest value among the general evaluations. Seven of the nine participants provided substantive comments, which were grouped into the themes presented in Table 4. In the table, Freq. represents the number of participants who mentioned each theme. Since one response could address more than one limitation, the frequencies are not mutually exclusive.

Two patterns deserve attention. First, incomplete coverage was mentioned by six of the seven respondents, showing that the main limitation is not necessarily in the processing performed by SSM, but in the available public data. Second, four participants reported distortions in the technologies or in the assigned area, indicating that the presence of code in a profile does not, by itself, determine authorship, intensity of use, or technical expertise.

Incomplete coverage included activities performed in private repositories, GitLab, unpublished academic projects, and outdated profiles. This result reinforces that a small number of public records may indicate limited coverage of the source rather than limited developer experience.

Distortions in technologies or in the assigned area involved, among other cases, automated exercises, code produced by collaborators, and technologies used only in private environments. The other two themes highlight interpretation limitations. Activities in personal repositories, including issues created and resolved by the same user, may be confused with participation in *open source*

Table 4: Themes identified in the open-ended responses.

Theme	Freq.	Examples
Incomplete coverage of experience	6	Private repositories, GitLab, missing academic projects, and an outdated profile
Distortion of technologies or technical area	4	Overestimated language, omitted language, and incompatible technical class
Need for greater clarity	3	Explanation of criteria, metrics, coverage, and limitations
Public activity interpreted as OSS participation	2	Issues from personal projects and personal repositories treated as community engagement

communities. In addition, three participants requested clearer explanations about criteria, metrics, and the amount of evidence. Therefore, the results should be accompanied by information that allows the coverage of the data and the conditions used to calculate the indicators to be examined.

6 Threats to Validity

The threats to validity were organized into construct, internal, external, and conclusion validity, following Wohlin et al. [20].

Construct validity. The SSM indicators are operationalizations of public traces and not direct measures of competence, productivity, communication, adaptability, or commitment. To reduce broader interpretations, the indicators were presented separately, no overall developer score was produced, and the interpretation limits of each indicator were made explicit. The questionnaire concepts were also treated as broader than the implemented indicators. Therefore, participant responses were interpreted as perceived adherence and not as construct validation. The results still depend on the selected signals, aggregation rules, and saturation points.

Internal validity. Voluntary participation may have favored students with greater interest in the topic or more frequent GitHub use. To reduce researcher selection, the entire class was invited without filtering by academic performance, professional experience, or platform activity. The academic relationship with the researchers may still have influenced the responses.

Technical area labeling also involved subjective decisions. To reduce this threat, two evaluators examined public self-descriptions and complementary sources, discussed disagreements until consensus, and removed cases without an identifiable predominant area. In the predictive evaluation, the test set remained outside training, configuration selection, and synthetic data generation. GANBLR was also fitted only on the training subset of each cross-validation fold to reduce information leakage.

External validity. The perception study included nine students from one Information Systems class. Therefore, its findings were interpreted as exploratory and cannot be directly generalized to experienced developers, industrial teams, or *open source* communities.

The classifier dataset was collected without restrictions on technology, location, or technical area, and pseudo-random starting points were used to reduce dependence on popular profiles. However, GitHub does not provide a complete sampling frame, and the 1,240 profiles are not a probabilistic representation of the platform. Moreover, SSM observes only public GitHub activity and does not capture private repositories, other platforms, or local development environments.

Conclusion validity. The small perception sample makes the results sensitive to individual responses. To reduce inappropriate statistical assumptions, the analysis used the Friedman test, Holm correction, effect sizes, and individual response distributions. Since no pairwise comparison remained significant after correction, no conclusion was drawn about specific pairs of concepts.

The predictive results were reported using per-class metrics, macro and weighted averages, and a Wilson confidence interval for accuracy. However, the evaluation used one final test partition and one synthetic generation seed. Therefore, the observed benefit of GANBLR applies only to the evaluated configuration. The qualitative themes were also used descriptively, since they were derived from seven respondents without independent coding or agreement analysis.

7 Conclusion

This article presented SSM (*Source Signal Miner*), a method that organizes technical, temporal, collaborative, and textual signals extracted from public GitHub repositories. The method produces five observable indicators, assigns a predominant technical area, and consolidates the results into a report that makes explicit the amount and coverage of the evidence used. The indicators describe patterns of public activity and do not constitute direct measures of competence, productivity, or professional performance.

In the perception study with nine participants, the mean adherence of the report components ranged from 0.63 to 0.73. The report received a mean score of 0.76 regarding its representation of the content available in the public profile, while the ability of the profile itself to represent the complete experience received a mean score of 0.42. In the predictive evaluation with 372 real profiles, the classifier achieved an accuracy of 88.17% and a macro F1-score of 0.88. However, the results also revealed limitations related to the incomplete coverage of public data, the attribution of technologies, and the representation of hybrid technical profiles. Therefore, SSM should be used as a complementary source for organizing and initially inspecting evidence, and not as an autonomous mechanism for ranking or evaluating developers.

Future work includes evaluations with professionals and in organizational environments, sensitivity analysis of the indicators, comparison of balancing strategies, and the development of multilabel models or abstention mechanisms for hybrid profiles. Future studies will also investigate approaches to indicate insufficient evidence and to objectively evaluate the traceability of the results presented in the report.

Artifact Availability

The instruments, scripts, SSM implementation, and other replication materials are available on Zenodo [1].

Acknowledgments

This research is supported by FAPEMIG (APQ-00763-24).

[1] Zenodo repository

References

[1] John Anvik, Lyndon Hiew, and Gail C. Murphy. 2006. Who Should Fix This Bug?. In *Proceedings of the 28th International Conference on Software Engineering*. ACM, 361–370. doi:10.1145/1134285.1134336

[2] Lingfeng Bao, Xin Xia, David Lo, and Gail C. Murphy. 2019. A Large Scale Study of Long-Time Contributor Prediction for GitHub Projects. *IEEE Trans. Software Eng.* 47 (2019), 1277–1298. doi:10.1109/tse.2019.2918536

[3] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment Polarity Detection for Software Development. *Empirical Software Engineering* 23, 3 (2018), 1352–1382. doi:10.1007/s10664-017-9542-0

[4] Chiara Del Vescovo, Davide Taibi, and Andrea Janes. 2021. Profiling Developers Through the Analysis of GitHub Public Repositories. In *Proc. IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. doi:10.1109/msr52588.2021.00059

[5] Thomas Fritz, Jingwen Ou, Gail C. Murphy, and Thomas Zimmermann. 2010. A Degree-of-Knowledge Model to Capture Source Code Familiarity. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*. ACM, 385–394.

[6] Emitza Guzmán, David Azócar, and Yang Li. 2014. Sentiment Analysis of Commit Comments in GitHub: An Empirical Study. In *Proc. 11th Working Conf. on Mining Software Repositories*. 352–355. doi:10.1145/2597073.2597118

[7] Clayton J. Hutto and Eric Gilbert. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *Proc. 8th International AAAI Conf. on Weblogs and Social Media*.

[8] Robbert Jongeling, Subhajit Datta, and Alexander Serebrenik. 2015. Choosing Your Weapons: On Sentiment Analysis Tools for Software Engineering Research. In *2015 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 531–535. doi:10.1109/ICSM.2015.7332510

[9] Huzefa Kagdi, Maen Hammad, and Jonathan I. Maletic. 2008. Who Can Help Me with This Source Code Change?. In *2008 IEEE International Conference on Software Maintenance*. IEEE, 157–166. doi:10.1109/ICSM.2008.4658064

[10] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The Promises and Perils of Mining GitHub. In *Proc. MSR*. 92–101. doi:10.1145/2597073.2597074

[11] Jenny T. Liang, Thomas Zimmermann, and Denae Ford. 2022. Towards Mining OSS Skills from GitHub Activity. In *Proceedings of the 44th International Conference on Software Engineering: New Ideas and Emerging Results*. 106–110. doi:10.1145/3510455.3512772

[12] Jenny T. Liang, Thomas Zimmermann, and Denae Ford. 2022. Understanding Skills for OSS Communities on GitHub. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. doi:10.1145/3540250.3549082

[13] Audris Mockus and James D. Herbsleb. 2002. Expertise Browser: A Quantitative Approach to Identifying Expertise. In *Proceedings of the 24th International Conference on Software Engineering*. ACM, 503–512. doi:10.1145/581339.581401

[14] João Eduardo Montandon, Luciana Lourdes Silva, and Marco Tulio Valente. 2019. Identifying Experts in Software Libraries and Frameworks among GitHub Users. In *Proceedings of the 16th International Conference on Mining Software Repositories*. 276–287. doi:10.1109/MSR.2019.00054

[15] João Eduardo Montandon, Luciana Lourdes Silva, and Marco Tulio Valente. 2021. Mining the Technical Roles of GitHub Users. *Information and Software Technology* 131 (2021), 106485.

[16] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. 2018. A Benchmark Study on Sentiment Analysis for Software Engineering Research. In *Proceedings of the 15th International Conference on Mining Software Repositories*. ACM, 364–375. doi:10.1145/3196398.3196403

[17] Edson Oliveira, Eduardo Fernandes, Igor Steinmacher, Marco Cristo, Tayana Conte, and Alessandro Garcia. 2020. Code and Commit Metrics of Developer Productivity: A Study on Team Leaders Perceptions. *Empirical Software Engineering* 25 (2020), 2519–2549. doi:10.1007/s10664-020-09820-z

[18] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and Burnout in Open Source: Toward Finding, Understanding, and Mitigating Unhealthy Interactions. In *2020 IEEE/ACM 42nd ICSE-NIER*. 57–60. doi:10.1145/3377816.3381732

[19] Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Raula Gaikovina Kula, Norihiro Yoshida, Hajimu Iida, and Ken ichi Matsumoto. 2015. Who Should Review My Code? A File Location-Based Code-Reviewer Recommendation Approach for Modern Code Review. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering*. IEEE, 141–150. doi:10.1109/SANER.2015.7081824

[20] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer. doi:10.1007/978-3-642-29044-2

[21] Wenxin Xiao, Hao He, Wenxin Xu, Yuxia Zhang, and Minghui Zhou. 2023. How Early Participation Determines Long-Term Sustained Activity in GitHub Projects?. In *Proc. 31st ACM ESEC/FSE*. doi:10.1145/3611643.3616349

[22] Yishuo Zhang, Nayyar Zaidi, Jiangning Zhou, and Gang Li. 2021. GANBLR: A Tabular Data Generation Model. In *Proc. IEEE International Conference on Data Mining (ICDM)*. 181–190. doi:10.1109/icdm51629.2021.00103