DisTrac: **Dis**engagement **Trac**king Tool in Open Source Projects

Samuel Utzinger*, Pedro Oliveira*, Fabio Calefato†, Marco Gerosa*, Igor Steinmacher*
*Northern Arizona University, Flagstaff, AZ, USA; †University of Bari, Bari, Italy
Email: {swu3, Pedro.Oliveira, Marco.Gerosa, Igor.Steinmacher}@nau.edu, fabio.calefato@uniba.it

***Abstract*—Open Source Software (OSS) projects often depend on a small number of developers, making contributor disengagement a risk for project sustainability. Prior work has characterized developer inactivity, but maintainers still lack practical tools to anticipate potential departures and understand their impact. This paper presents DISTRAC, an interactive tool for forecasting core developer inactivity and visualizing its potential consequences. DISTRAC builds per-developer activity profiles from public GitHub traces and combines project health, socio-technical network, and knowledge distribution features into a horizon-aware LSTM model that predicts the likelihood that a core developer will become inactive. The tool connects these predictions to an interactive dashboard that displays files at risk, project health indicators, and changes to the collaboration network in the event of a departure. We report preliminary feedback from 2 OSS maintainers, showing that prediction accuracy alone does not guarantee usefulness. Maintainers of small projects often already have contextual knowledge, while larger distributed communities and future successors may benefit more from the tool. A demonstration video is available at https://youtu.be/AnKz4ZnoVs0 and an online tool access at https://samutz1-distrack.hf.space/**

***Index Terms*—OSS, developer inactivity, sustainability**

I. INTRODUCTION

Open Source Software (OSS) enables anyone to inspect, modify, and distribute code, sustaining communities built on distributed collaboration, and collective innovation [1]. These communities operate in a decentralized way, with limited co-worker awareness, and a decentralized peer-production model [2]. In practice, this socio-technical structure tends to concentrate expertise and decision-making authority disproportionately in a small group of contributors, a phenomenon known as the *Maintainer Paradox* [3]. The concentration of knowledge creates fragile dependencies in OSS projects, where the developers' departure can harm a project.

Avelino et al. [4] formalized this fragility through the truck factor (TF), which identifies the minimum number of developers whose loss would impair a project. Their study showed that 65% of 133 popular GitHub projects have a TF of at most two $\leq$ 2 developers. As stated by Coelho and Valente [5], the disengagement of central contributors is a recurring pattern in OSS projects, with 16% of the abandoned projects analyzed in their study exhibiting inactive behavior. Compounding this, Calefato et al. [6] showed that core developers frequently take extended breaks from OSS projects; 45% of breaks exceed a year, and only 21–26% of developers return after such breaks [6], [7].

For critical infrastructure ecosystems, these disruptions can propagate across the software supply chain [8], [9], [10]. Despite the extensive research on TF, inactivity patterns, developer knowledge concentration, and socio-technical networks [7], [4], [11], [12], [6], [13], [14], we still lack operational tools to anticipate breaks from OSS communities. Previous studies do not translate these findings into warning mechanisms and typically defer such tooling to future work. Project maintenance remains reactive, analyzing the impacts after consequences materialize.

To fill this gap, we present DISTRAC, a tool for forecasting developer inactivity and visualizing its potential impact on OSS projects. We describe a pipeline that integrates developer contribution indicators into a unified maintenance support system and a preliminary evaluation with OSS practitioners, discussing the perceived usefulness of the tool.

II. TOOL OVERVIEW

DISTRAC is an interactive web-based tool to support OSS community maintainers in forecasting developer disengagement and understanding project risks. The tool transforms public GitHub activity into information that helps maintainers anticipate the project landscape if a core developer (identified in this study as the Truck Factor developers) leaves. We restrict modeling to those developers identified via the truck factor algorithm [4]: the smallest set of developers whose departure would leave most of the project's files without an owner. This set captures the contributors in whom the project's knowledge is most concentrated and whose disengagement poses the greatest sustainability risk. The primary users of DISTRAC are OSS maintainers and community managers who need early warnings about potential departures and visibility into the consequences of such departures. DISTRAC implements a three-stage pipeline (Figure 1):

1. **Data Collection & Modeling**. Given a target repository, DISTRAC collects GitHub activity data for the project and repositories under the same organization, then curates it into per-developer daily activity profiles that capture both technical and social contributions.
2. **Prediction Dashboard**. DISTRAC extracts features along project health, socio-technical network, and knowledge distribution dimensions and feeds them into a temporal model that forecasts the probability of a core developer transitioning into an inactive state within the next K days.

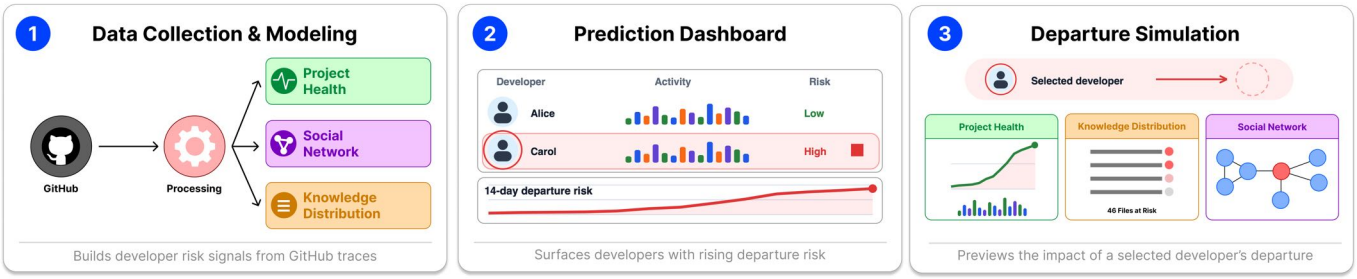

Fig. 1. DISTRAC Overview

3. **Departure Simulation**. For any selected developer, DISTRAC previews the consequences of their departure across the three dimensions used by the model, showing at-risk files, affected collaborators, and shifts in project activity.

III. FORECASTING DEVELOPER INACTIVITY

A. Data Collection

OSS projects are collaborative, and the actual labor is largely invisible. Reading code, debugging, designing changes, and drafting patches happen on individual developer machines and leave no online traces. What reaches GitHub is only the published, finalized work, such as commits pushed to a branch, PRs opened, reviews, or comments. There is no clock-in signal or live presence indicator. We cannot watch a developer work; we can only see what they have finished and shared.

DISTRAC treats data collection as a sweep of every public trace a developer leaves on a project. For each target repository and every repository under the same GitHub organization, we collect commits, PRs and issues (and their related events: merges, comments, reviews, labels, and assignments), developer metadata, and commit history. Each record is preserved with its timestamp and author. This data stream represents temporal events that capture the technical side (what code was touched) and the social side (who reviewed whom, who answered whose comments). The dense raw outputs are finally processed into per-developer daily activity profiles.

B. Features

For each core developer and day, we collapse the surrounding events into a single fixed-width record organized along three dimensions of OSS sustainability validated in prior work [13], [11], [14]: **Project Health (PH)**: the volume, consistency, and continuity of the developer's contributions; **Socio-Technical Network (STN)**: the structure of their collaborations, their level of connectedness, and the extent to which they interact with regular contributors versus newcomers; **Knowledge Distribution (KD)**: files owned, code modified, and code collaborations with other developers' work. The full feature set is available in the replication package [15].

C. Prediction Target

We adopt the four-state activity model from Calefato and colleagues [6] (*Active*, *Non-Coding*, *Inactive*, and *Gone*) to label each day in a developer's timeline based on their observed contributions. We re-implement the model causally: day d is labeled using only events up to day d, so the response matches what is observable at inference time. These labels are a rule-based approximation of a latent construct that has no objective ground truth; Calefato et al. [6] report 97% agreement between such labels and developers' self-reported breaks.

DISTRAC forecasts the transition into the *Inactive* or *Gone* states. We frame this as horizon-aware binary classification: given a developer's past 90 days of daily activity, we predict whether they will enter one of these two states within the next K days.

D. Model and Evaluation

We trained with a three-layer LSTM with 128 hidden units (that consumes 90-day windows of developers' daily feature vector $X_{t-89:t} \in \mathbb{R}^{90\times d}$) with binary cross-entropy, AdamW, positive-class loss weighting to address class imbalance, and early stopping on a held-out validation fold. Our project collection list spans 45 GitHub organizations with 100+ repositories, yielding 226 core (Truck Factor) developers and 870,000 developer-days. We label a developer-day as a positive example if the developer transitions into the *Inactive* or *Gone* state within the next K days finding only a few useful transition periods.

We evaluate under Leave-One-Developer-Out (LODO) to test within-project generalization, and Leave-One-Project-Out (LOPO) to test transfer to projects unseen at training time. We compare against a 7-day silence heuristic and a logistic regression on the same features. Because inactivity states are rare, we report *PR-AUC* and window-level recall at a fixed basic global threshold of one alert each week for a developer. The LSTM reaches a PR-AUC of 0.7754 (LOPO) above the LOPO logistic baseline (0.4506) and the LOPO naive heuristic (0.4937). Window-level recall reaches 87.6% of true K-day onset windows at the target alert rate under LODO.

IV. DASHBOARD AND TOOL VISUALIZATIONS

DISTRAC displays the model's output through two linked views: a prediction timeline for spotting at-risk developers, and a departure simulation that answers the question maintainers actually ask next: *what happens if they leave*? The simulation is organized around the same three dimensions that feed the model (PH, STN, KD), so the features driving a prediction are also the lenses through which its consequences are explored.

A. Prediction Timeline

The maintainer begins by selecting a repository and a time window. DISTRAC renders each core developer's contribution history as a temporal strip (Figure 2), with commits, pull requests, reviews, and issue activity laid out day by day. Beneath the strip, two parallel tracks show the labeled state y and the model's predicted probability $\hat{y}$ of an upcoming transition into *Inactive* or *Gone*.

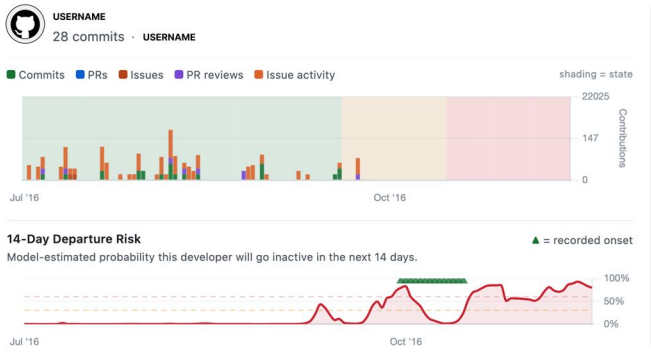

Fig. 2. User Activity Trend Visualization

The timeline answers *who should I be paying attention to?* by showing developers whose predicted probability $\hat{y}$ exceeds their baseline. They can click on the developers "at risk" (or any developer) to simulate their departure.

B. Departure Simulation

Based on the previous step, a maintainer can select a developer who may depart and simulate the effect on the repository. Three tabs structure this exploration:

1) ***Knowledge at Risk****:* This view highlights files that are primarily developed by the selected developer. It is built based on the *Degree of Expertise* (DOE) [16] (developer's level of knowledge and ownership over a file) and the *Truck Factor* (TF) [4] (minimum number of developers whose departure would impair the project). Rather than displaying the results as a standalone tool, we feed the per-file expertise distribution directly into DISTRAC's inactivity-aware dashboard.

We classify each file into one of 3 risk tiers or discard it based on how many experts the file has (see Table I and Figure 3). When the inactivity model flags a developer, the dashboard prominently surfaces files where that developer holds the dominant DOE and the file falls into a non-green risk tier (e.g., Sole Expert, Sole Maintainer, and Narrow Expertise).

2) ***Collaboration Network****:* DISTRAC renders the project socio-technical network as a graph where developers are nodes and edges are weighted by shared activity (co-participation in issues, pull requests, reviews, comments, and file contributions). When a developer is selected, DISTRAC simulates their removal from the network and compares the project structure before and after the departure (Figure 4). This comparison allows maintainers to inspect how many collaboration links would be lost, which contributors would be most affected, and whether the departure could isolate parts of the community.

TABLE I
PER-FILE KNOWLEDGE-DISTRIBUTION RISK TIERS, DERIVED FROM THE DISTRIBUTION OF DOE ACROSS CONTRIBUTORS.

Tier	Condition
SOLE EXPERT	Exactly one developer holds non-zero DOE on the file.
SOLE MAINTAINER	A single developer holds >80% of the file's total DOE.
NARROW EXPERTISE	Exactly two developers hold meaningful DOE between them.
BROADER COVERAGE	Three or more devs hold meaningful DOE.

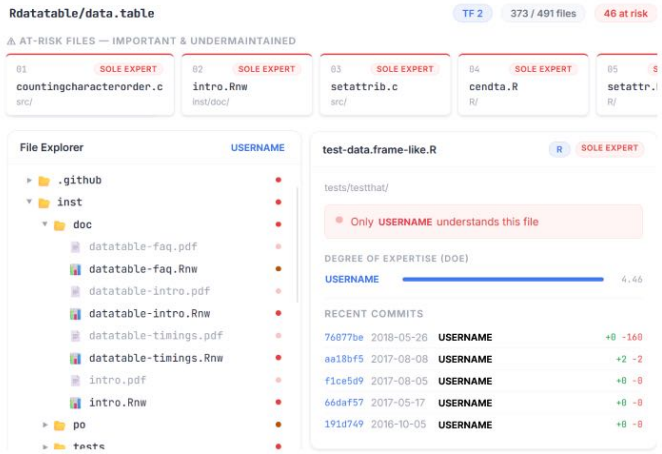

Fig. 3. Files at Risk Visualization

3) ***Project Health****:* DISTRAC provides a summary view of key project health metrics, with the developer's share of total activity, weekly contribution trends, and the proportion of the repository's recent commits and reviews they account for (Figure 5). This view is intentionally lightweight; the goal is to show a quick signal, so a maintainer can tell whether the loss in question is "10% of the work" or "60% of the work."

C. Usage Scenario

We illustrate a DISTRAC session on `Rdatatable/data.table`, a widely used a free reference manager maintained by a small core team. A maintainer opens the tool, points it at the repository, and

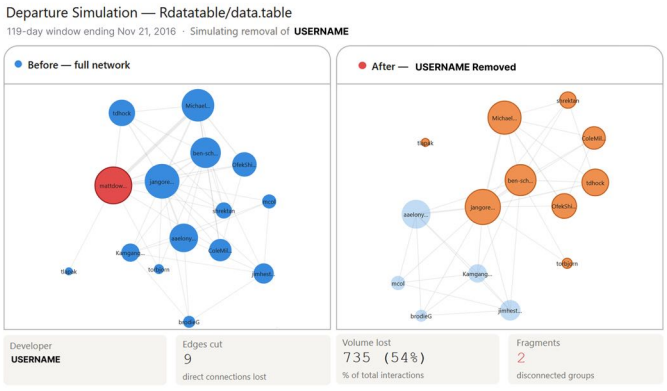

Fig. 4. Social Network and Departure Analysis

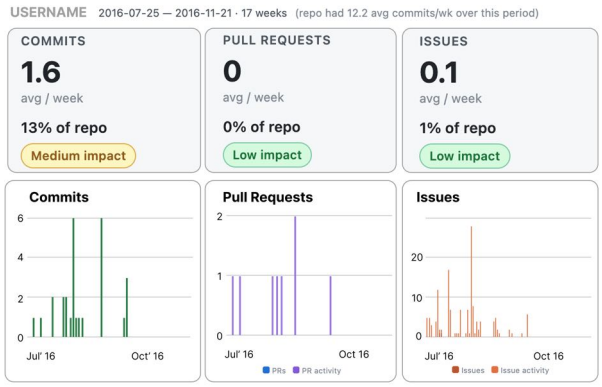

Fig. 5. Project Metrics Impact on Developer Disengagement

selects a 17-week window ending August 4th 2023. The prediction timeline (Figure 2) flags one core developer whose $\hat{y}$ rises in the weeks preceding observed gaps in activity. The manager clicks through to the simulation.

Project Health DISTRAC provides a lightweight summary of the selected developer's footprint in the project. For commits, pull requests, and issues, it reports their average weekly volume, the share of the repository's total that each represents, and an impact label, alongside per-week trend charts (Figure 5).

The **Knowledge at Risk** tab reveals that 46 of the developer's 373 touched files are solely maintained by them, all marked SOLE EXPERT. One – `test-data.frame-like.R` – carries the message *"Only USERNAME understands this file"*, with a commit history confirming no other contributor has touched it. These files are concrete targets for documentation, pair programming, or successor onboarding.

The **Collaboration Network** tab (Figure 4) shows that the simulated removal eliminates 9 direct collaboration edges (54% of the developer's interaction volume in the window) and fragments the graph into 2 disconnected components. The manager can identify which contributors lose their strongest link and which sub-community is most exposed.

V. COMMUNITY FEEDBACK

To get an initial idea of DISTRAC's practical usefulness, we conducted informal interviews with OSS maintainers from JabRef and data.table. Each session began with a discussion of the warning signs preceding departures and how the participant currently monitors activity, followed by a walkthrough of DISTRAC on their own repository. Participants were asked whether the information matched their understanding of the project, how they interpreted each view, and how a K-day forecast would have changed how they handled past departures.

A. Feedback Analysis

The simulation views externalize knowledge that is otherwise tacit. Both maintainers reported that the dashboard's ownership and collaboration views matched what they already held in their heads as long-term contributors. Neither identified the views as personally informative, but both independently pointed to settings where the same information would not already be known, like successors moving into established projects, and large distributed communities where no single contributor holds the full picture. One participant explicitly suggested deploying DISTRAC on a separate project they were less embedded in. *The tool's value is conditional on the audience, and our interview sample sat outside that audience.*

Predictions are not yet connected to practice. When asked whether a K-day forecast would have changed how they handled a recent departure, both participants answered no. The reason was not prediction error: the departures they encountered were driven by funding cuts, job changes, or family priorities, events that an upstream alert does not give a maintainer the means to act on. One participant noted that reframing the forecast to active instead of inactive *could* redirect retention effort toward developers predicted to stay. *The value of inactivity prediction depends on intervention mechanisms that OSS communities may not have in place.*

The simulation views carried more signal than the prediction. Asked which view was most useful, one participant singled out the Collaboration Network with simulated removal, calling it "exactly accurate" and asking to remove more than one developer; the other singled out the activity-history timeline exploration. Both flagged the Knowledge-at-Risk file list as concrete input for successor onboarding in cases when one developer dominates a file's ownership. Both also asked for analysis beyond Truck Factor developers — a scope our current pipeline does not support.

VI. LIMITATIONS

Technical. DISTRAC ingests only public GitHub data; mailing lists, Slack, and private code review tools are not yet integrated, and developers active in those channels appear less engaged than they are. The prediction target inherits the assumptions of the Calefato et al. four-state heuristic [6]: contributors with irregular cadence are harder to classify, and the label is a proxy rather than a measured disengagement event. The current deployment is retrospective and single-repository at a time; a continuously updated, multi-repository service is ongoing work.

From the interviews. Two gaps showed that the tool does not yet close. *Scope*: maintainers asked to analyze contributors beyond the Truck Factor set, whose trajectories matter for community health but who fall outside our current modeling target. *Action*: it was not clear how a K-day forecast translates into maintainer action when the underlying departure drivers (funding, job changes, life events) lie outside the maintainers' control. Closing this gap requires co-designed intervention mechanisms (lightweight check-ins, successor pathways, workload-reassignment prompts, retention-focused reframings of the forecast) that OSS communities currently lack. Both are explicit directions for the next version of the tool.

VII. RELATED WORK

OSS sustainability has been studied extensively regarding contributor retention, knowledge concentration, and project

fragility. DISTRAC synthesizes several lines of this research, adapting prior ideas at nearly every stage of its pipeline.

The premise that a project depends on a few contributors is formalized by the *truck factor* of Avelino et al. [4], whose authorship-based algorithm we use to select the core developers for DISTRAC to model. Our forecasting target is itself borrowed; the four-state activity model (*Active*, *Non-Coding*, *Inactive*, *Gone*) was introduced by Iaffaldano et al. [7] and Calefato et al. [6], and DISTRAC learns to forecast a causal re-implementation of their labels, the basis for this research.

That such departures matter is established by Miller et al. [2] and Coelho and Valente [5], who document how the loss of central contributors stalls or abandons projects. DISTRAC's three feature dimensions reuse further work: Knowledge contribution builds on the *Degree of Expertise* of Bird et al. [16] and the per-file operationalization of Cury et al. [14]; the sociotechnical network is from collaboration structure under project decline [11], [12] and on Oliveira et al.'s [3] account of authority concentration in OSS.

DISTRAC's contribution is connecting the prediction with a simulation of its consequences. The closest recent work is Xu et al. [17], which forecasts OSS project abandonment from temporal traces of repositories. Xu et al. predict at the *repository* level whether an entire project will die. DISTRAC predicts at the *developer* level, using the inactivity framework by Calefato et al. [6] to forecast disengagement before it aggregates into project-level decline.

VIII. CONCLUSION

We presented DISTRAC, a tool for forecasting core-developer inactivity and visualizing the consequences of a predicted departure. The tool combines project health, sociotechnical network, and knowledge distribution features into a single dashboard and pairs each prediction with a simulation view that surfaces at-risk files, affected collaborators, and the overall scale of impact. Across 100+ repositories and 226 core developers, the underlying model is able outperform basic models on true inactivity-onset windows with appropriate binary thresholds.

Interviews with OSS maintainers sharpened our view of when DISTRAC is useful and when it is not. Long-tenured maintainers of small projects already hold the ownership and collaboration structure in their heads, and noted that the same information becomes valuable to successors and members of large distributed communities that lack mutual awareness. They also exposed an *action gap*: a prediction is only as useful as the intervention it triggers, and OSS communities lack the management levers that enterprise settings rely on.

Future work includes a continuously updated multi-repository deployment and the integration of contribution signals beyond GitHub, broadening the visibility into off-platform work. Our interviews indicate that inactivity prediction and a maintenance dashboard are useful only in context. OSS communities lack some traditional management features to implement large-scale deployment of a managerial application. Therefore, we will co-design lightweight intervention mechanisms (check-ins, successor pathways, retention prompts) with maintainers, so that a prediction lands as the start of an action rather than the end of an observation.

IX. REPLICATION PACKAGE

The replication package is available at https://zenodo.org/records/20435459. A demonstration video is available at youtu.be/AnKz4ZnoVs0.

X. ACKNOWLEDGMENT

Work supported by TRIF Faculty Research and Creative Activity Support Grants program at Northern Arizona University.

REFERENCES

[1] T. L. Foundation, "What is open source software?" https://www.linuxfoundation.org/blog/blog/what-is-open-source-software, 2017, accessed: 2025-11-06.

[2] C. Miller, D. Widder, C. Kästner, and B. Vasilescu, "Why do people give up flossing? a study of contributor disengagement in open source," in *International Conference on Open Source Systems*, 2019, pp. 116–129.

[3] P. Oliveira, T. Conte, M. Gerosa, and I. Steinmacher, "Governance in practice: How open source projects define and document roles," *CHASE 2026*, p. 255, 2026.

[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] J. Coelho and M. T. Valente, "Why modern open source projects fail," in *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*, 2017, pp. 186–196.

[6] F. Calefato, M. A. Gerosa, G. Iaffaldano, F. Lanubile, and I. Steinmacher, "Will you come back to contribute? investigating the inactivity of OSS core developers in GitHub," *Empirical Software Engineering*, vol. 27, no. 3, p. 76, 2022.

[7] G. Iaffaldano, I. Steinmacher, F. Calefato, M. Gerosa, and F. Lanubile, "Why do developers take breaks from contributing to OSS projects? a preliminary analysis," *arXiv preprint arXiv:1903.09528*, 2019.

[8] A. Alami, R. Pardo, and J. Linåker, "Free open source communities sustainability: Does it make a difference in software quality?" *Empirical Software Engineering*, vol. 29, no. 5, p. 114, 2024.

[9] J. Sun, "Sustaining scientific open-source software ecosystems: challenges, practices, and opportunities," in *46th International Conference on Software Engineering: Doctoral Symposium*, 2024, pp. 234–236.

[10] M. Hoffmann, F. Nagle, and Y. Zhou, "The value of open source software," *Harvard Business School Strategy Unit Working Paper*, no. 24-038, 2024.

[11] P. Arantes, F. Soupinski, and A. Fontão, "Social networks during software ecosystems' death," in *SESoS 2023*. IEEE, 2023, pp. 9–12.

[12] S. Herbold, A. Amirfallah, F. Trautsch, and J. Grabowski, "A systematic mapping study of developer social network research," *Journal of Systems and Software*, vol. 171, p. 110802, 2021.

[13] CHAOSS, "Community health analytics in open source software (CHAOSS)," https://chaoss.community, 2025, accessed: 2025-11-06.

[14] O. Cury and G. Avelino, "Knowledge islands: Visualizing developers knowledge concentration," pp. 789–795, 2024.

[15] S. Utzinger, P. Oliveira, F. Calefato, M. Gerosa, and I. Steinmacher, "DisTrac: Disengagement Tracking Tool in OSS Projects — Replication Package," 2026. [Online]. Available: https://zenodo.org/records/20435459

[16] C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu, "Don't touch my code! examining the effects of ownership on software quality," in *ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 4–14.

[17] Y. Xu, R. He, H. Ye, M. Zhou, and H. Wang, "Predicting abandonment of open source software projects with an integrated feature framework," *arXiv preprint arXiv:2507.21678*, 2025.