Enhancing Open-Source Contributor Onboarding and Engagement: A Data-Driven Approach Using Personalized Dashboards

Ammar Ali
Dr. Mehjabin Khatoon(Professor)
Mudassir Ahmed Khan
Nikhil
Divyansh Thakur

Scholar, B.Tech. (CSE) 4th Year Department of Computer Science Engineering,
Dr. Akhilesh Das Gupta Institute of Professional Studies, New Delhi

Abstract

Tools and artifacts produced by open source software (OSS) have been woven into the foundation of the technology industry. To keep this foundation intact, the open source community needs to actively invest in sustainable approaches to bring in new contributors and nurture existing ones. We take a first step at this by collaboratively designing a maintainer dashboard that provides recommendations on how to attract and retain open source contributors. For example, by highlighting project goals (e.g., a social good cause) to attract diverse contributors and mechanisms to acknowledge (e.g., a "rising contributor" badge) existing contributors. Next, we conduct a project-specific evaluation with maintainers to better understand use cases in which this tool will be most helpful at supporting their plans for growth. From analyzing feedback, we find recommendations to be useful at signaling projects as welcoming and providing gentle nudges for maintainers to proactively recognize emerging contributors. However, there are complexities to consider when designing recommendations such as the project current development state (e.g., deadlines, milestones, refactoring) and governance model.

1. Introduction

1.1 The Sustainability Challenge in Open Source Software

Despite the ubiquity of open source in our technology infrastructure and products, OSS projects struggle with a variety of challenges that can impact project health and sustainability. Research has found OSS to be a challenging ecosystem to navigate, both for newcomers and for experienced OSS contributors in large mature organizations . We scope our study to help, attract and retain newcomers as this group is essential in ensuring a constant flow of contributors and a sustainable community. Newcomers have been found to face particularly challenging barriers. They often struggle with even finding a task to work on. Even after newcomers have contributed to the project, retaining them is nontrivial. In some projects, as high as 80% of these contributors do not transition to long term contributors . A lack of newcomers can also hurt the health and sustainability of OSS projects. It is thus important to implement proactive measures to help make OSS projects more welcoming for newcomers to join and stay. Previous research has focused on solutions to help newcomers, for example, through mentoring or an information platform that overcomes the common newcomers' barriers . In this paper, we focus on the other end of the spectrum by empowering maintainers to improve their project and community health for newcomers.

1.2 A Socio-Technical Solution: The Personalized Contributor Dashboard

To tackle this significant attrition issue, this paper suggests a socio-technical intervention: a data-driven, Personalized Contributor Dashboard. While human mentorship is a very effective onboarding method, it is a limited resource that cannot be easily scaled in volunteer-based OSS communities. The proposed dashboard is intended to automate and scale the essential functions of a mentor—offering guidance, feedback, and encouragement. It is more than just an information

source; it is a dynamic system designed to actively guide, motivate, and integrate newcomers into a project's community. By delivering personalized task recommendations, real-time progress updates, and clear goals, the dashboard directly tackles the lack of support materials and timely feedback, which are recognized as key factors in newcomer attrition.

1.3 Theoretical Grounding and Research Questions

The design of the Personalized Contributor Dashboard is deliberately based on established psychological theories of human motivation. We argue that an effective onboarding tool must cater to the basic psychological needs of its users. Our approach combines three complementary theories: Self-Determination Theory (SDT), which guides the fostering of intrinsic motivation; Goal-Setting Theory, which offers a structure for organizing actions and effort; and Self-Efficacy Theory, which details how to build a contributor's confidence and resilience. This multi-theoretical framework informs the design of each feature to ensure a cohesive and motivating experience.

This study is shaped by the following three research questions (RQs), which are designed to assess the effectiveness of this approach:

- **RQ1:** Does providing personalized issue recommendations (based on user skills and project data) increase first-contribution success, compared to generic/manual discovery?
- **RQ2:** Does access to real-time progress tracking and gamified elements (badges, streaks, stats) lead to higher retention/frequency of contributions versus contributors without such feedback?
- **RQ3:** Does dashboard-driven onboarding increase new contributors' perceived self-efficacy and satisfaction compared to traditional approaches?

By exploring these questions, this paper aims to offer empirical backing for the benefits of personalized, data-driven interventions in enhancing the health and sustainability of open source communities.

2. Background and Related Work

2.1 The Newcomer's Journey: A Review of Onboarding Barriers in OSS

The challenges that contributors face include not finding a good task to start with , a lack of mentors , not being recognized for one's work, a mismatch between contributors' motivation in joining a project and the project goals, as well as a mismatch in career goals and expectations. Scaffolding newcomer learning or encouraging contributors is not an easy task, as community leaders (e.g., maintainers) have limited time to attend to both the project and community needs. Past research has identified strategies to overcome some of these challenges. Several works have researched mechanisms to help newcomers start to contribute. For example, newcomer-friendly issue labels that explicitly highlight starter tasks can help newcomers make their first contribution. Santos et al provided an approach to automatically identify the skills needed for an OSS tasks, which could then be used to label issues. On the other hand, Huang et al. have shown that certain project topics, especially those that relate to social good help in attracting a diverse set of contributors. Other works have shown the benefit of badges as an attractor for projects. For example, code quality badges that signal project quality can help create positive impression among contributors. The challenges that contributors face include not finding a good task to start with, a lack of mentors, not being recognized for one's work, a mismatch between contributors' motivation in joining a project and the project goals, as well as a mismatch in career goals and expectations. Scaffolding newcomer learning or encouraging contributors is not an easy task, as community leaders (e.g., maintainers) have limited time to attend to both the project and community needs. Past research has identified strategies to overcome some of these challenges. Several works have researched mechanisms to help newcomers start to contribute. For example, newcomer-friendly issue labels that explicitly highlight starter tasks can help newcomers make their first contribution. Santos et al. provided an approach to automatically identify the skills needed for an OSS tasks, which could then be used to label issues. On the other hand, Huang et al. have shown that certain project topics, especially those that relate to social good help in attracting a diverse set of contributors. Other works have shown the benefit of badges as an attractor for projects. For example, code quality badges that signal project quality can help create positive impression among contributors.

2.3 State of the Art in Contributor Support Tools

2.3.1 Gamification and Self-Tracking in Software Development

The use of game-design elements in non-game contexts, known as gamification, has been investigated as a way to boost motivation and engagement in software engineering. Common elements include points, badges, levels, and leaderboards, which can make routine or difficult tasks more enjoyable and competitive. Studies have indicated that these elements can improve student engagement in software engineering education and motivate developers in professional settings. The most frequently used elements are points, badges, and leaderboards, which offer clear feedback and a sense of accomplishment.

At the same time, self-tracking and developer productivity dashboards have become more widespread. These tools offer metrics on activities such as deployment frequency, cycle time, and commit volume. However, a major issue with such tools is the risk of focusing on easily measured but potentially misleading metrics (e.g., lines of code) that favor quantity over quality and can lead to burnout. Additionally, manual time-tracking can be seen as tedious and a threat to developer autonomy. An effective dashboard must therefore provide meaningful metrics that show genuine progress without turning into a tool for micromanagement.

2.3.2 From Generic Labels to Personalized Task Recommendation

The most common method for guiding newcomers is the manual labeling of issues by maintainers with tags like good first issue or help wanted. While this approach is well-intentioned, it has considerable drawbacks. Research indicates that these generically labeled issues are often not numerous enough or are a poor fit for the varied skills and backgrounds of individual newcomers, leading to a high rate of unsuccessful contribution attempts. A "good first issue" for an experienced developer who is new to a project is quite different from one for a student making their first-ever contribution.

This discrepancy underscores the critical need for personalization. Recent research has shifted towards creating automated, personalized first issue recommenders. For instance, the PFIRec model by Wang et al. uses machine learning to rank potential issues for a specific newcomer by considering features of both the individual (e.g., expertise preference, prior OSS experience) and the issue. This approach is in line with broader trends in recommendation systems, which employ techniques like content-based filtering (matching item attributes to user profiles) and collaborative filtering (using the behavior of similar users) to offer customized suggestions. Our proposed system expands on this line of research by incorporating a personalized recommendation engine as the foundation of a comprehensive, motivation-focused onboarding experience.

3. System Design: The Personalized Contributor Dashboard

The Personalized Contributor Dashboard is a web-based application created to offer a complete and motivating onboarding experience for new contributors to OSS projects. Its architecture and features are a direct application of the theoretical principles discussed in the previous section.

3.1 System Architecture and Technology Stack

The system is designed as a modern, single-page application (SPA) that adheres to the principles of Clean Architecture to ensure a clear separation of concerns and ease of maintenance. This layered design separates business logic from framework-specific details, making testing and future updates easier.

- **Frontend Framework:** The user interface is developed using **Next.js**, a widely used React framework. Next.js was chosen for its robust features, such as a file-system-based App Router for clear and intuitive routing, and its support for both Server-Side Rendering (SSR) for dynamic data and Static Site Generation (SSG) for fast-loading static content. This hybrid rendering approach ensures both high performance and current information.
- **Language: TypeScript** is utilized across the entire codebase. Its static typing leads to better code quality, improved developer tools, and enhanced maintainability, which is essential for a project of this kind.
- **Backend-for-Frontend (BFF):** The system uses a BFF pattern, with **Next.js Route Handlers** acting as API endpoints. These server-side functions manage all interactions with the external GitHub API, process and combine data, and handle sensitive credentials, ensuring that the client-side application receives only the necessary

data in the correct format.

- **Styling:** The user interface is styled with a modern CSS framework like Tailwind CSS, which enables the quick development of a responsive and visually appealing design.
- **Deployment:** The application is designed for containerization with **Docker**, which simplifies the setup and deployment process in various environments.

3.2 Data Layer: Harnessing the GitHub API

The dashboard relies exclusively on data obtained from the official GitHub API. This method guarantees that all information is accurate and current, reflecting the real-time status of the target OSS project.

- **API Strategy:** A dual-API approach is used. The **GitHub REST API** is employed for fetching specific, well-defined resources, such as repository statistics, commit activity, and user profiles. For more intricate data requirements, like retrieving nested information about issues, comments, and related pull requests in a single request, the **GitHub GraphQL API** is used. GraphQL's flexible query language allows the application to request precisely the data it needs, avoiding the over-fetching often seen with REST APIs and enhancing performance.
- **Data Points:** The system gathers a broad range of data points to create a detailed view of both the contributor and the project. This includes user data (public repositories, languages, starred projects), repository data (issues, labels, pull requests, commit history), and community interaction data (comments, reviews).
- **Authentication:** Users authenticate through GitHub's OAuth flow. The application then uses the user's access token to make authenticated API requests, ensuring access to user-specific data while respecting GitHub's API rate limits.

4. Project Growth : Attracting Newcomers

For project growth, the dashboard uses two recommendations to help attract newcomers: newcomer-friendly issue labels, and OSS goal(s)project tags.

4.1 Newcomer-Friendly Issue Labels.

The dashboard analyzes the issue tracker data to identify:(1)the percent of issues that are labeled with a newcomer friendly label by string-matching against the set of newcomer-friendly labels in “MunGell / awesome for-beginners” GitHub repository. This list is curated from 187 repositories in 22 programming languages. Additionally , the dashboard detects if the issue labels the newcomer friendly labels used by other OSS projects . Based on the above data, the dashboard suggests adding newcomer friendly label(s) such as “good first issue”.

4.2 OSS Goals

We used the approach from Huang et al.To collect OSS4SG projects’ goals. The dashboard recommended projects that are about “social good” to signal their goals through README.MD badges to help attract contributors. Maintainers found this recommendation can help their project in the following ways.Increases visibility... Maintainers felt “those meta tags allow it to be more contextually meaningful ” and “make it more more visible”, which was considered “important, particularly, for young projects” and attracts diverse contributors and users. Maintainers felt this recommendation would help “engage a broader spectrum” of contributors, especially from “traditionally underrepresented people in computer science”. found that this could be useful to not only attract contributors but also users by indicating “there’s this project out there that can be interesting and useful to you. Never mind contributing to it, but use it”. Maintainers also shared that this can emphasize that “working in the social good ones doesn’t diminish the experience for the developers”

5. Conclusion and Future Work

5.1 Summary of Contributions

In this work, we took a first step in providing a systematic mechanism through which OSS community leaders (e.g., maintainers) can be proactive about their community's health and sustainability. We designed and evaluated a dashboard that provides recommendations for maintainers to help them attract (e.g., advertise project goals) and retain (e.g., recognize contributors) newcomers to their project. We worked closely with OEs, who understand the project practices and communicate regularly with OSS participants.

One key feedback when creating recommendations was the need to take into consideration the project's contribution model. With the open source ecosystem increasingly using hybrid models of contribution, different governance models exist. While some projects are driven by a community of volunteers, others are company-driven with paid employees being the main contributors. explained: "because this project in specific was company sponsored...Most of the things that work here , don't work elsewhere".

5.2 Directions for Future Research

To summarize, open source software is a key digital infrastructure that drives many of our products and a venue for workforce development. However, current project maintainers are often resourceconstrained to scaffold newcomers' learning, which is not only detrimental to the project's health, but also to society. Therefore, as researchers we need to identify mechanisms to help maintainers attract and retain (new) contributors. As different OSS projects have different characteristics and needs, it is important to continue to "listen" to maintainers to better understand what tools and support they need. Maintainers' listening tours and industry conferences are great venues to foster industry-academia collaborations.

References

Barriers Faced by Newcomers to Open Source Projects: A Systematic Review. *Proceedings of the 10th International Conference on Open Source Systems*. Steinmacher, I., Conte, T., & Gerosa, M. A. (2014).

Barriers Faced by Newcomers to Open Source Projects: A Systematic Review. *ime.usp.br*. Various Authors. (2023).

The harsh reality of getting contributors for your open source project. *reddit.com*. Trinkenreich, B., et al. (2020).

Recommending Tasks to Newcomers in OSS Projects: How Do Mentors Handle It? *Proceedings of the 16th International Symposium on Open Collaboration*. da Silva, F. Q. B., et al. (2023).

Understanding Self-Efficacy in the Context of Software Engineering. *arxiv.org*. Ushahidi Team. (2019).

How to find (and keep) open source contributors. *opensource.com*. Various Authors. (2019).

What are the biggest challenges when onboarding to a new codebase/project?. *dev.to*. Chen, K.-C., & Jang, S.-J. (2010).

Motivation in online learning: Testing a model of self-determination theory. *selfdeterminationtheory.org*. Jääskelä, P., et al. (2022).

Investigating the relation of higher education students' situational self-efficacy beliefs to participation in group level regulation of learning during a collaborative task. *tandfonline.com*. Berges, M., et al. (2021).

Introductory Programming Self-Efficacy Scale (IPSES). *csedresearch.org*. Wang, Y., Lo, D., & Xia, X. (2023).

Personalized First Issue Recommender for Newcomers in Open Source Projects. *arxiv.org*. Gagne, M., & Deci, E. L. (2014).

Self-Determination Theory in Work Organizations: The State of a Science. *researchgate.net*. Al-Zubaidi, H., et al. (2024).

Confidence in Code: The Relationship Between Self-Efficacy and Participation. *tudelft.nl*. University of Rochester Medical Center. (n.d.). Self-Determination Theory. *urmc.rochester.edu*. Wikipedia contributors. (n.d.). Self-determination theory. *en.wikipedia.org*. Berges, M., et al. (2021).

Measuring Self-Efficacy in Programming Surveys. *csedresearch.org*. Various Authors. (2022).

Gamification in software engineering academic papers. *researchgate.net*. Olafsen, A. H., & Deci, E. L. (2020).

Self-Determination Theory and Organizations. *selfdeterminationtheory.org*. GitHub Docs. (n.d.). About the REST API. *github.com*. Lee, S., et al. (2024).

Enhancing Programming Self-Efficacy and Performance in Interdisciplinary Education. *mdpi.com*. Tufano, M., et al. (2024).

Attracting and Retaining Newcomers to Open Source Software Projects: A Large-Scale Quantitative Study. *arxiv.org*. GitHub Docs. (n.d.). REST API endpoints for repository statistics. *github.com*. Appfire. (n.d.). The do's and don'ts of time tracking in software development. *appfire.com*. Proaction International. (n.d.). How Self-Determination Theory Enhances Employee Motivation. *proactioninternational.com*. Spaeth, S., et al. (2012).

Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *econstor.eu*. Melzar. (n.d.). Clean Architecture with Next.js and Typescript. *github.com*. OpsLevel. (n.d.). A Complete Guide to Goal Setting for Software Engineers. *opslevel.com*. Namespace. (2024).

GitHub API Demystified: A Developer's Handbook. *namespacecomm.in*. Rasheed, M. I., et al. (2021).

The Role of Personal and Motivational Factors in Influencing Value Co-creation Engagement. *frontiersin.org*. Lee, S., et al. (2024).

Self-Efficacy in open source contribution. *mdpi.com*. Appcues. (n.d.). A/B testing user onboarding flows. *appcues.com*. Lark. (2024).

Locke's Goal-Setting Theory for Information Technology Teams. *larksuite.com*. Arounda. (2024).

Top 20 SMART Goals for Software Developers Examples. *arounda.agency*. eLearning Industry. (2023).

The Self-Determination Theory In Online Training: A Practical Guide. *elearningindustry.com*. Sharma, P. (2024). Mastering A/B Testing in Mobile App Development. *medium.com*. Digital Thriving Playbook. (2025).

Self-Determination Theory for Multiplayer Games. *digitalthrivingplaybook.org*. Pinto, G., et al. (2021).

The shifting sands of motivation: Revisiting what drives contributors in open source. *smu.edu.sg*. GeeksforGeeks. (n.d.). Goal Setting Theory: Meaning, Working, Principles, and Examples. *geeksforgeeks.org*. OSS Software. (n.d.). How to Contribute to Open Source Projects: Best Practices. *osssoftware.org*. Vinogradov, K. (2021).

OSS Contributor Engagement Patterns and Metrics. *kvinogradov.com*. Benbya, H., & Belbaly, N. (2005).

Understanding Developers' Motives in Open Source Projects: A Multi-Theoretical Framework. *researchgate.net*. FullStack Labs. (2025).

Recommendation System Development with AWS Personalize. *fullstack.com*. Vercel. (n.d.). Next.js by Vercel - The React Framework. *nextjs.org*. Timalsina, B. (2023).

Intrinsic Motivation for Open Source Development. *medium.com*. Atlassian. (n.d.). Goal setting theory. *atlassian.com*. Vinogradov, K. (2021).

OSS contributor engagement patterns. *kvinogradov.com*. Wang, Y., et al. (2023).

Personalized First Issue Recommender for Newcomers in Open Source Projects. *researchgate.net*. Hertel, G., et al. (2003).

Motivation of Software Developers in Open Source Projects. *researchgate.net*. Rathnayake, P. (2023).

SMART Goals for Achieving Programming Success. *medium.com*. Lark. (2024). Goal-Setting Theory in software development. *larksuite.com*. Riehle, D. (2012).

Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *econstor.eu*. Melzar. (n.d.). Modern Web Dashboard Architecture with React, TypeScript, and Next.js. *github.com*. Benbya, H., & Belbaly, N. (2005).

Motivational theories in open source software. *researchgate.net*. OpsLevel. (n.d.). How to measure and improve developer productivity. *opslevel.com*. Netflix Technology Blog. (2024).

A Foundation Model for Personalized Recommendation. *netflixtechblog.com*. Sharma, A. (2024).

Designing User Dashboards in Next.js. *medium.com*. Torchiano, M., et al. (2022).

Gamify: Gamification in Software Development, Verification, and Validation. *polito.it*. Wrike. (n.d.). Goal-Setting Theory and Frameworks. *wrike.com*. Clearcode. (n.d.). What Motivates a Developer to Contribute to Open-Source Software?. *clearcode.cc*. Graphite. (n.d.). GitHub statistics and analytics for your repositories. *graphite.dev*. Lönn, C. (2012).

Motivation in Open Source Software Development. *lup.lub.lu.se*. Appcues. (n.d.). Best Practices for A/B Testing User Onboarding Flows. *appcues.com*. Teaching Python. (2024).

Setting Goals for Coding Success. *teachingpython.fm*. Sertis. (2024).

Personalized Recommendation Systems. *medium.com*. GitHub Docs. (n.d.). GitHub GraphQL API documentation. *github.com*. Chauhan, H. (2024).

Step-by-Step Guide to Building an Admin Dashboard with Next.js. *dev.to*. M Accelerator. (n.d.). The Ultimate Guide to A/B Testing Onboarding Flows. *maccelerator.la*. OrcDev. (2024).

Building a Dashboard with Next.js 15, Shadcn, TypeScript & Tailwind v4. *youtube.com*. Meegle. (2025).

Personalized Recommendation Algorithms. *meegle.com*. Suryono, R. R., et al. (2022).

Gamification in Software Development: Systematic Literature Review. *researchgate.net*. Ngandu, M. R., et al. (2023).

Capturing student interest in software engineering through gamification. *semanticscholar.org*. Google Cloud. (2023).

Building a recommendation system on Google Cloud. *google.com*. Meegle. (2025)..