University of Zurich^UZH

Bruno Rodrigues, Eder J. Scheid, Jonas Brunner
Calvin Falter, Guilherme Sperb Machado, Thomas Bocek,
Burkhard Stiller

FlatFeeStack: a Blockchain-based Sustainable Public Funding of Open Source Projects

TECHNICAL REPORT – No. IFI-2022.05

February 2022

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

FlatFeeStack: a Blockchain-based Sustainable Public Funding of Open Source Projects

Bruno Rodrigues[1], Eder J. Scheid[1], Jonas Brunner[1], Calvin Falter[1],
Guilherme Sperb Machado[2], Thomas Bocek[3], Burkhard Stiller[1]
[1]Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH—8050 Zürich, Switzerland
[2]AxLabs, Zürich, Switzerland
[3]Ostschweizer Fachhochschule OST, Informatik
Oberseestrasse 10, CH—8640 Rapperswil, Switzerland
E-mail: [rodrigues¦scheid¦stiller]@ifi.uzh.ch, [jonas.brunner¦calvin.falter]@uzh.ch, guil@axlabs.com, thomas.bocek@ost.ch
`https://flatfeestack.io`

***Abstract*—Recent studies have shown a decrease in the number of Open Source Software (OSS) projects and their contributions due to several reasons, including the increased complexity of OSS, a lack of time from contributors, a lack of OSS funding, and also pointing to the need for extrinsic motivations for contributors to invest time and effort in OSS.**

Thus, the design and development of `FlatFeeStack` were performed to offer a blockchain-based platform that promotes incentives to sponsors and contributors to foster OSS based on three main innovations: *(i)* a sponsorship model based on flat fees including multipliers that allow for the predictability of the amount sponsored to projects and encourage third-party sponsors; *(ii)* an engine allowing for the evaluation of individual contributions within a project to establish incentives based on individual contributions; and *(iii)* enabling the payout to contributors in cryptocurrencies simplifying the payment flow to contributors and removing intermediaries to process payments, which also allows for reinvesting (*i.e.*, staking) cryptocurrencies.

***Index Terms*—Blockchain, Distributed Ledger, Open Source Software, Incentives, Funding Schemes, Cryptocurrencies**

I. Introduction

Open Source Software (OSS) is essential to foster innovation through transparent, collaborative contributions without copyright restrictions restricting their use and distribution. However, recent studies [11], [24], [6], [12] have shown a decrease in the number of OSS projects and their contributions due to several reasons, including the increased complexity of OSS, a lack of time from contributors, a lack of OSS funding, and also pointing to the need for extrinsic motivations for contributors to invest time and effort in OSS [20], [21], [32]. An analysis in [12] presented the overall decline in the number of OSS commits from 2014 onward, considering 224'342 OSS projects with a total of 180'937,525 commits (*cf.* Figure 1), and confirms the reasons for the decline in contributions for OSS as described in [11] and [24].

In this regard, intrinsic and altruistic motivations have been for many years the driving motivation for OSS contributors, but the increased complexity that often requires more time for relevant contributions has also exposed a need for extrinsic (mainly financial) incentives. The major factors are also exposed in the same studies [11], [24], [12], showing that the increasing number of OSS projects also influences the competition and the complexity of these projects, which often seek details to differentiate itself from others.

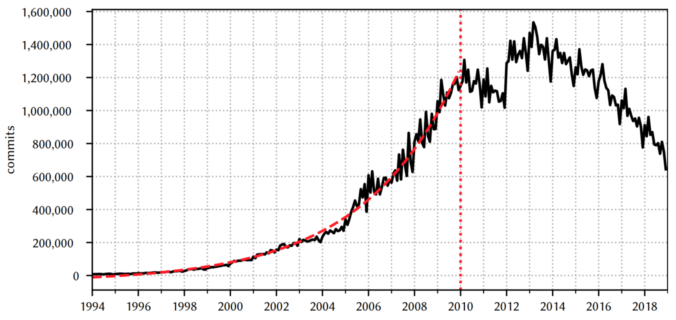

Fig. 1: Decline in Commits in OSS Projects from 2013 [12]

The need for incentives is not different within the Blockchain community. Vitalik Buterin, the founder of Ethereum, recently highlighted the need to create a sustainable and structured public funding scheme to retroactively incentivize Blockchain projects [4], [30] (*i.e.*, "give back to the community"). This reinforces the need that extrinsic motivations (*i.e.*, financial incentives) are necessary not only to foster contribution in traditional OSS, but to create a form of sustainable income for these contributors [20]. In this sense, the Blockchain community not only requires measurable incentives to solve open challenges, but it can also offer a solution to the general problem on "how to create a retroactive public funding of OSS", as it provides a decentralized platform for developing financial applications [35].

Platforms to raise funds are not a novel concept to entrepreneurs, content creators, and artists. Patreon [34], BuyMeACoffee [5], and Ko-Fi [23] are platforms offering a way to offer an alternative source of income based on fiat currency. Yet, studies from [11], [6], [24], [12], [20] showing the decrease of OSS projects and their related contributions indicate that such platforms do not efficiently stimulate OSS contributors. For example, such platforms *(i)* are typically centralized and not transparent to contributors, specifically affecting their traction in the Blockchain community (favoring decentralized solutions) [4], *(ii)* often present high-fees that hinder the income of contributors and affecting all types of contributors, and *(iii)* lack sponsoring models that incentivize sponsors (individuals or companies) to continuously support OSS projects.

`FlatFeeStack` builds upon the principle that sponsoring (*cf.* definitions in Section II) platforms necessarily need to consider as a fundamental basis of their design a *public and transparent* way to create a structured and sustainable source of income that *fairly* incentivizes contributors.

`FlatFeeStack` provides incentives for both contributors and sponsors to boost cooperative behavior allowing for the following key contributions:

- Enabling the sponsoring of multiple OSS projects instead of individual ones. Thus, it simplifies the sponsoring process by avoiding multiple registrations of sponsors and contributors over different OSSes.
- Enabling individual and corporate sponsors to provide a periodic flat fee to support selected projects, including the option to corporate sponsors for including multipliers.
- Distributing donations based on a ranking of contributions per OSS. Thus, contributors would receive a donation proportional to the relevance of their contribution.
- Enabling an optional pay-in or payout to contributors in cryptocurrencies, which simplifies the payment flow by removing intermediaries and fees, and allows contributors to reinvest (*e.g.*, by staking).

A flat fee in different categories allow sponsors to know beforehand the amount to be distributed within an OSS. Also, multipliers allow corporate sponsors to multiply a donation of individual sponsors up to a certain pre-allocated budget. In addition, enabling the sponsoring of multiple OSS further simplifies the sponsoring side by allowing a sponsor to select one or more OSS, and contributors associated with those projects would receive an amount proportional to the relevance of their contribution.

A proportional distribution of incentives per OSS incentivizes contributors to not only increase the number of contributions, but also their quality. `FlatFeeStack` introduces such an approach to determine the relevance of contributions based on the assumption that each OSS determines the relevance of contributions in their Git repository. Lastly, the alternative payment flow in cryptocurrency allows for a simpler payment flow with reduced fees in contrast to traditional fiat payout methods.

The remainder of this paper is organized as follows: Section II overviews fundamentals and related work. While Section III describes the design and analytics and details experiments on each component, Section V summarizes the work, extended by future steps.

II. Background and Related Work

Incentives for entrepreneurs, content creators, software developers, and artists are not necessarily the same, rendering platforms effective or ineffective. Thus, definitions on fundraising platforms, different models that influence open source and Blockchain communities are compared.

A. Definitions

Fundraising: A broad definition encompasses all types of funds coming to individuals or organizations, including the mission and its methods to raise funds [15].
Crowdsourcing: A type of fundraising that leverages a network of people to support (*i.e.*, financial donations or other social contributions) a cause or business [15], [6].
Sponsoring: Another form of fundraising concerned with a long-term relationship between sponsors (donate) and contributors (receive donations) [3].

`FlatFeeStack` provides an alternative where developers from OSS and Blockchain communities continuously receive funds depending on their contributions toward OSS, fostering innovation and code maintainability, and incentivizing participation. In this sense, the *fundraising* and *crowdsourcing* models do not apply for the `FlatFeeStack` case due to their short-termed nature, where funds are gathered and shared once. Hence, the *sponsoring* model applies for `FlatFeeStack`, which presents a long-term and recurrent cycle of funding.

B. Payment Models

Different sponsorship models and payment structures can make platforms interesting (*i.e.*, incentivize) to sponsors. Depending on the target community, platforms may opt for a one-off model with variable payments, a subscription model with a fixed payment [20], or a hybrid approach. While one-off models are suitable to individual sponsors, a fixed (flat) fee subscription model is more suitable to corporate individuals, once it allows to predict costs and explore the social-responsibility image [6].

C. Incentives

Incentives are typically categorized as intrinsic or extrinsic. On one hand, while individual sponsors are typically motivated by intrinsic values (*e.g.*, sense of community and ideological belief), corporate sponsors can be driven by intrinsic and extrinsic values to retroactively help a community (*e.g.*, a handy open-source library) and get rewarded by exploiting the social-responsibility image [6] publicly. On the other hand, contributors are incentivized by both intrinsic and extrinsic motivations, as observed in [20], [11]. In addition, technical characteristics, such as the level of decentralization, open-source code, transparency, and payment (pay-in and pay-out) in cryptocurrencies, are relevant factors to incentivize, in particular, OSS and Blockchain communities.

TABLE I: Comparison of Fundraising Platforms

Platform	**Type**	**Pay-in Model**	**Pay-out**	**Sponsoring Structure**	**Proportional Sponsoring**	**Architecture**	**Open Source**
Patreon [34]	Crowdfunding	One-off	Direct transfer, Payoneer, PayPal	1 to 1 (individuals)	✗	Centralized	✗
BuyMeACoffee [5]	Crowdfunding	One-off	Direct transfer, PayPal, Stripe	1 to 1 (individuals)	✗	Centralized	✗
Flattr [14]	Sponsoring	One-off and Subscription	Direct transfer	1 to N (multiple projects)	✗	Centralized	✗
Ko-Fi [23]	Crowdfunding	One-off	PayPal, Stripe	1 to 1 (individuals)	✗	Centralized	✗
Liberapay [26]	Crowdfunding	One-off and subscription	PayPal, Stripe	1 to 1 (individuals)	✗	Centralized	✓
Goteo [19]	Crowdfunding	One-off	Direct transfer, PayPal	1 to 1 (individuals)	✗	Centralized	✓
Gitcoin [16]	Crowdfunding	One-off	Cryptocurrency	1 to 1 (individuals)	✗	Decentralized	✗
GitHub Sponsors [17]	Sponsoring	One-off and subscription	Direct transfer	1 to N (multiple projects)	✗	Centralized	✗
Issuehunt [22]	Sponsoring	One-off and subscription	Stripe, direct transfer	1 to 1 (individuals)	✗	Centralized	✓
Tidelift [37]	Crowdfunding	Subscription	Stripe	1 to N (multiple projects)	✗	Centralized	✓
Open Collective [27]	Sponsoring	One-off and subscription	Stripe, direct transfer PayPal	1 to N (multiple projects)	✗	Centralized	✓
FlatFeeStack [8]	Sponsoring	One-off and subscription	Stripe, Cryptocurrencies	1 to N (multiple projects)	✓	Hybrid	✓

D. Related Work

In order to encourage individual and organizational sponsors as well as contributors to OSS and Blockchain projects, Table I compares fundraising platforms considering their type, payment models and structure, architecture, and whether they allow for a proportional sponsoring of contributors. In this sense, the first important distinction is their type. Most of these crowdfunding platforms are relatively recent and based on the sponsoring model.

The sponsorship model based on periodic subscriptions (*e.g.*, Flattr [14], Liberapay [26], GitHub Sponsors [17], and `FlatFeeStack`) aligns with the goal of creating continuous public funding for contributors. By directly integrating with payment providers such as Stripe or PayPal, the money is directly transferred to the creator's payment service provider account. Conversely, the one-off models (*e.g.*, Patreon [34], BuyMeACoffee [5], Goteo [19], Gitcoin [16]) may be more suitable to artists or to raise funds to *kickstart* a project, but they are not ideal for fostering continuous contributions to running projects.

Open Collective[27] is a fully transparent donation platform based on monthly recurring payments. While many of the registered "collectives" represent open source projects, the platform can also be used for other non-profit projects. Issuehunt [22] is a platform which is specifically tailored to open source software donations. Open issues can be funded (*i.e.*, creating a bounty) to incentivize more developers to work on a task. The developer who resolves an issue first gets the funded amount. Tidelift [37] provides a managed open source solution for software development teams. With the cheapest subscription being 1500$ per month and the direct integration into the CI workflow, the platform is built for enterprise-level customers.

While financial rewards are the most obvious incentive, [25] shows that not all developers are willing to accept rewards, especially when they feel that the financial support could take influence on the innovation process. A recent study by Overney [31] shows that only a small fraction (0.04%) of public repositories on GitHub ask for donations. Still, the demand for open source donations has increased significantly during the past three years, with PayPal being the top requested donation method across all `Readme.md` files on GitHub [31]. This is especially interesting because PayPal is a general digital payment service that is not specifically tailored towards open source funding.

The architectural design of these platforms also determines a vital role to incentivize sponsors and contributors. For instance, characteristics such as crypto donations, transparency, and decentralization are relevant factors for the Blockchain community. However, the possibility of a payment in fiat currency necessarily involves a factor of centralization and reliance on third parties (*e.g.*, to process credit card transactions). Gitcoin [16] and `FlatFeeStack` are highlighted, in which the latter was designed in a hybrid architecture by also including the possibility of fiat payments.

However, `FlatFeeStack` as designed differs from related work including a contribution analysis component, which allows for the evaluation of the relevance of contributions (*e.g.*, commits to a Git repository) and a proportional distribution of sponsored resources to a project. Also, sponsors can include a multiplier model that allows sponsoring companies to multiply the donation of individual sponsors up to a specific pre-allocated budget in the contract.

III. Blockchain-based Public Funding Solution

The payment flow of the newly designed FlatFeeStack's (*cf.* Figure 2) outlines a sponsor determining the type of donation (*i.e.*, subscription or a single one-off) for one or more projects and the proportion to which the donation will be distributed among projects. In this sense, FlatFeeStack operates as a multi-asset wallet in which the contributor receives the asset that the sponsor sends without performing conversions to a specific currency. After the sponsor donates in the first stage, the amount is distributed in the proportion determined among the projects flagged by the sponsor. The second stage (*i.e.*, project fanning) is illustrated as an example with two projects, where the donation is equally distributed to each project.

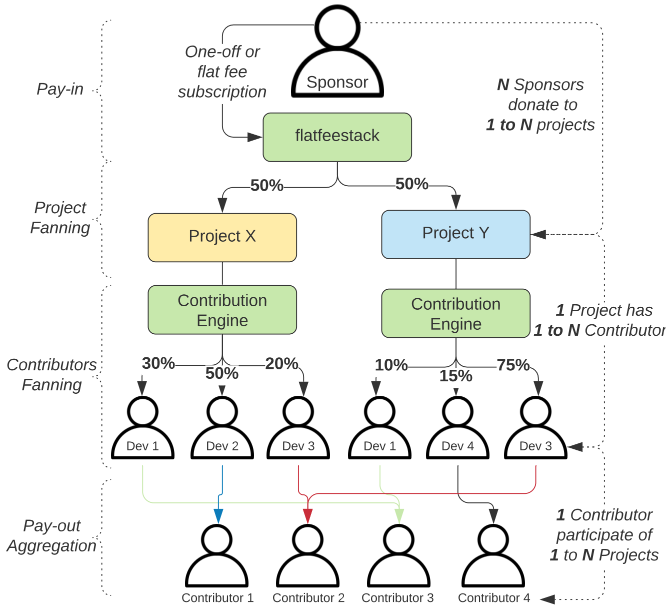

Fig. 2: Example Sponsoring Flow

In the third stage (*i.e.*, contributors fanning), the contribution engine analyzes the Git repository associated with each project to assess each contributor's contribution associated with the project. A contributor's weight is calculated using the commit evaluation engine, which computes a score based on git contributions. The weight calculations are based on a window of three months, and the weights are updated monthly.

A multiple-week interval prevents unequal distribution on less active repositories. In a weekly window, a single modest edit to a README.md file (with no other commits this week) may yield the same reward as a major pull request the week before. A broad window, suggests that a contributor's compensation is spread across several weeks. This means that the incentive for a single commit is paid out in three monthly installments. Since a contributor may be affiliated with one or more projects, the payment is aggregated before the distribution in the final step.

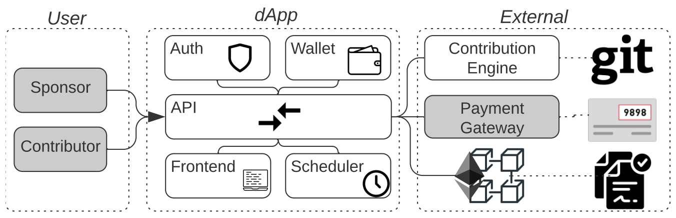

Fig. 3: FlatFeeStack Architecture

A. Architecture

The FlatFeeStack's architecture (*cf.* Figure 3) manages payments and funds of users (sponsors or contributors). While sponsors can be corporate or individual donors, contributors receive sponsorship through their contribution to open source Git repositories. The prototype is based on a Web dApp (decentralized application) that connects with the contribution engine oracle (*cf.* Subsection III-C), an external payment gateway service in case of credit card donations, and the EVM-compatible Blockchain (*e.g.*, Ethereum [38] or Binance chain [2]). The dApp has distinct functions based on type of user, for example, a sponsor may donate to certain projects and the percentage donations are distributed. These values must be transferred to the wallet, in which the scheduler triggers transactions in the selected period. Major dApp components include the following ones:

- **API**: The API (Application Programming Interface) connects to the database and provides various endpoints to manipulate the data, and checks if incoming requests are authenticated. Also credit card details are not handled by the API, only payment metadata;
- **Auth**: The authentication service manages the identity of users and grants access to request data from the API. It is based on a stateless (token-based) authentication, not being connected to the authentication service, but the identity of a user can be validated using a shared secret. This means that the API does not need to connect to the authentication service, but the identity of a user can be validated using a shared secret;
- **Frontend**: The frontend provides the user interface in order to register and sponsor projects. It uses the endpoints provided by the API to query data from the database;
- **Scheduler**: Triggers repeating tasks (*e.g.*, payouts) and aggregates data in the database; and
- **Wallet**: hold funds of sponsors or contributors.

In the contributor's role, the dApp works as a wallet in which the contributor enters his/her information about the projects he/she participates in, as well as the periodicity of the fund verification that the scheduler can automatically request. In the role of sponsor, the user configures the payment type, periodicity, and sponsoring multiplier parameters to incentivize the donation of individual sponsors (*cf.* Subsection III-B).

B. Sponsorship Multipliers

The addition of multiplier factors is a feature that allows corporate sponsors to incentivize the participation of individual sponsors. Lack of incentive donations is a major problem in the OSS community and, more recently, Blockchain, as described in a review of incentive mechanisms used by GitHub Sponsors [39]. A corporate sponsor allocates an initial budget IB to be distributed between K selected projects according to the pre-determined percentages per project Pct_x. Each selected project will have a starting budget B determined as $B_n = \sum_{n=1}^{K} IB * Pct_n$.

Algorithm 1 Multiplier of Donations

```
Require: $B_n = \sum_{n=1}^{K} IB * Pct_n$
Require: $MBI$ max. available budget per individual sponsor
Require: $AB$ max. available budget
 1: for each $P \in B_n$ do
 2:     $L \leftarrow$ list of new donations
 3:     for each $D, i \in L$ do                  ▷ Loop donations in L
 4:         if $D \le MID$ then                   ▷ Ensure small donations
 5:             $DN \leftarrow D * MF$             ▷ Multiply donation
 6:             $AB \leftarrow AB - DN$
 7:             $MBI[i] \leftarrow MBI[i] + DN$
 8:             if $AB \le 0 \,||\, MBI[i] \le 0$ then
 9:                 abort()                        ▷ Check upper bounds
10:             else
11:                 pay(DN) to P                   ▷ Pay multiplied donation
12:             end if
13:         end if
14:     end for
15: end for
```

Then, a multiplier factor MF can be applied to small donations up to a certain threshold MID as detailed in Algorithm 1 applied to multiply donations. The goal of MID in such an algorithm is to prevent multiplying large donations and incentivize a higher number of individual sponsors. Also, two upper thresholds are defined to ensure that corporate sponsor funds are not maliciously depleted. A maximum available budget AB defines an upper bound on the maximum sponsored value across all projects (*i.e.*, overall budget that a corporate sponsor is willing to donate), and an upper bound for individual sponsors termed AIB to prevent several small donations from the same individual sponsor (*i.e.*, prevent a single individual sponsor from consuming the entire AB budget).

An example based on Figure 4 considers a corporate sponsor making available a budget of 200 coins (*i.e.*, $AB = 200$). It was considered that 20 individual sponsors donated randomly generated values to evaluate the thresholds (*i.e.*, MID, MIB and AB) in each scenario. In addition, the corporate sponsor specifies a multiplier factor of 2 to incentivize individual sponsors. The first scenario in Figure 4 upper left considers the ideal scenario where donations do not exceed MID and MIB and are multiplied by the corporate donor (AB in blue donates the total budget being reduced). In the second scenario in Figure 4 upper right, the randomization range is slightly increased, implying that some donors exceed the specified MID, and thus their donations are not multiplied.

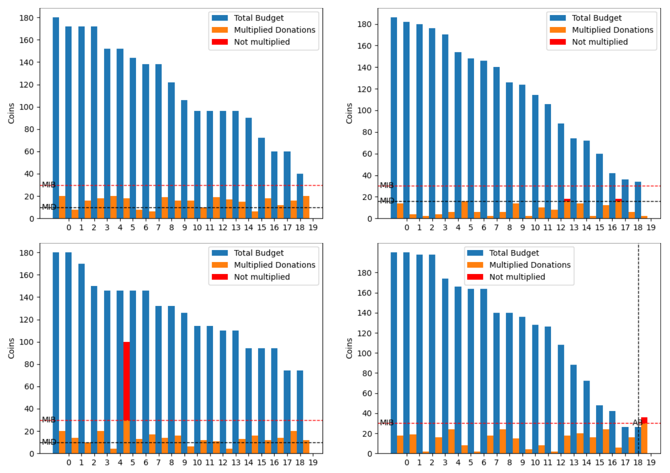

Fig. 4: Multipliers Analysis: Ideal Scenario (upper left) and Ignoring Donations Above MID (upper right). Ignore Donation (in red) Above MIB (lower left) and Scenario Where AB is Expired (lower right).

The multiplication factor provides an extra incentive for individual contributors to donate more and provides a safeguard against malicious acts that can deplete the budget allocated by the corporate sponsor. For example, if the ABI threshold is not specified in the third scenario (*cf.* Figure 4 lower left), another corporate sponsor can exhaust the allocated budget (*e.g.*, red bar) that would be multiplied, and prevent the donation from individual sponsors from being multiplied. Lastly, the last scenario presents the moment when the total budget specified by the corporate sponsor is reached (AB), and the multiplication factor is no longer applied.

C. Contribution Engine

One of the major incentives for contributors to be more active is the transparent and proportional distribution of donations according to their contribution to the repositories they participate. However, the assessment of contributions is not a straightforward task, in which simple quantitative metrics cannot translate the real relevance of contributions. Thus, it is necessary to extract specific information from the project in order to establish the relevance of, for example, a commit based on tags that allow the identification of how critical that change can be in the code. Thus, a developer no longer contributes only through written code but also through other activities such as communicating and coordinating with other people involved in the project.

The process of analysis in the contribution engine is structured in three major steps. The first step retrieves Git data, which is equivalent to cloning a Git repository. In a

TABLE II: Git and Platform Metrics

Git-Specific		**Platform-Specific**	
Changes	- Additions - Deletions	**Issues**	- Author of Issues - Comments on Created Issues - Written Comments
History	- Commits - Merges	**Pull Request**	- Author of Pull Request - Activity on Created Pull Request - Performed Code Reviews

second step, required metrics are extracted and collected to be offset against each other in the third step. Finally, each contributor is given a share of the total contribution.

1) Git Iterator: The engine only clones the part of the repository to be analyzed. The engine first update the repository and if this process fails, the repository is cloned. The cloning is executed so that the cloned repository will only contain one branch, which is the one defined earlier. To achieve this effect, the repository is configured as a single branch repository with the specific branch as this single branch. This means only the history leading to the tip of a single branch is cloned. Otherwise, if the repository already exists, it will only be fetched to update the remote-tracking branch so that Git will update the remote changes in the local repository.

2) Metrics: Two categories of metrics are distinguished (*cf.* Table II). One is Git and repository-based and contains metrics that can be read from a repository with Git. Since the analysis engine is based on a Git repository, this Git analysis is possible for queries. The second source of metrics is platform-specific information. This subdivision and allocation to categories allow a better overview of the weighting of the individual metrics. The weighting of the individual metrics is done in two stages. In the first stage, the weighting within a category is defined, and in the second stage, the weighting among categories. A weight within a stage is always defined so that weights can be added up to 1.

The contribution is then calculated in two stages in accordance with the defined weightings. In the first step, the developer's contribution to be analyzed is calculated as a percentage of the category's total contribution. In the second step, these calculated percentage contributions of a category are balanced against the categories' previously defined weightings. This results in the following formula for the contribution of a contributor.

$$C_{developer} = \sum_{c=0}^{c_{total}} \omega_c \times \frac{\sum_{m_c=0}^{m_{c,total}} \omega_{m_c} \times \theta_{m_c}}{\sum_{m_c=0}^{m_{c,total}} \omega_{m_c} \times \Theta m_c} \quad (1)$$

where:

$C_{developer}$	Calculated contribution of the developer
c_{total}	Total amount of categories
ω_c	Weight of the category c
c	Index of the category
m_c	Index of the metric inside category c
$m_{c,total}$	Total amount of metrics within category c
ω_{m_c}	Weight of the metric inside category c
θ_{m_c}	Value of the metric for the analyzed developer
Θ_{m_c}	Total value of the metric for all developers

3) Analysis: Analyzing the repository for each developer's contributions using Equation 1, which result in a list of each contributor with their percentage of the total contribution to the project within the analyzed time frame. Table III provides an overview of the categories used as well as metrics with the corresponding weights. The weightings were developed in a process in which they were initially determined based on [33] and self-assessment. They were adjusted through applications in real projects with the agreement of open source contributors.

TABLE III: Weights for Analysis (Based on [33])

ω_c	**Category**	ω_{m_c}	**Metric**
0.36	changes		
		0.7	additions
		0.3	deletions
0.30	history		
		0.7	commits
		0.3	merges
0.14	issues		
		0.5	author of issues
		0.2	comments on created issue
		0.3	written comments
0.20	pull requests		
		0.7	author of pull request
		0.3	performed code reviews

Table III shows the distribution of weights if no platform information is to be analyzed. Additionally to this metric weighting, different weights of pull requests exist due to the activity on a pull request. A closed request counts 0.6; an open one counts as 1.0. If a pull request is merged, it counts 1.5 times. In addition to these states, the activity of a pull request is also checked for approval. If this is the case, the above value is multiplied by 1.4. For example, an approved request, which was closed again afterward, receives a weight of 0.84 and an approved, and merged request receives a weight of 2.1. This multiplier is intended to promote agreement among the developers and assure the code quality since there is no other representation of code quality in the evaluation.

To assess the engine's results, the contributors involved in the repository evaluated the overall results. Since personal acquaintances to the main contributors of the *neow3j* repository were available, it was chosen as an application and evaluation example. For the repository analysis, the request was sent to the analysis engine. With this request, the *neow3j* repository on the master-3.x branch (the current master branch) should be analyzed from January 1st to July 1st. Tests in Figure 5 (left) confirmed the assumption that the time needed for mapping increases with the number of contributors. Nevertheless, it is interesting to note that the mapping duration is not constant when calculated down to one Git user. With a duration of 278 ms to 577 ms per Git user, it is so different that no reliable prediction can be made for a different number of users.

Responses indicate which Git contributor was responsible

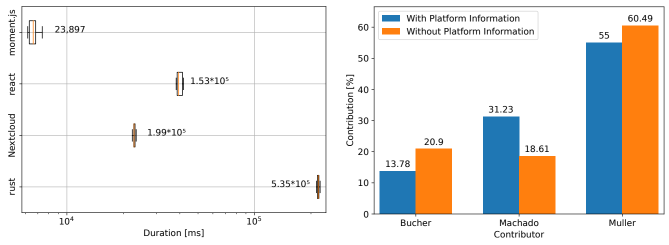

Fig. 5: Contribution Engine: Execution Time for Collecting Platform Information over 3 Months (left) and Analysis of Contributions in the *neow3j* Project (right)

for which partition of the total contribution in the selected period. This contribution's summary is conducted within multiple iterations, wherein these results were discussed with the repository owner at each iteration. Consequently, weights were adapted. The critical insight from the feedback and adaptation was that the platform information was not sufficiently weighted. Thus, this analysis is considered as representative of the actual contribution. Figure 5 shows the contribution per developer in the first semester of 2020, where platform information was considered once and not at the other time.

D. Payment Flow

The FlatFeeStack's payment flow is designed on two stages (contributor fanning and payout aggregation) as shown in Figure 2. Within the payout aggregation, a project contributor is identified by its Git email address. Once a user can have multiple Git email addresses configured (*e.g.*, a work email on the office computer and another email on the personal laptop), there exists a one-to-many relationship between users and contributors. Furthermore, a single contributor can be involved in multiple projects and generate income from multiple sources. Therefore, a payout aggregation is needed to map the rewards of contributors to the platform's registered users.

Receiving payments is as simple as validating an email address and setting a payout address now that cryptocurrencies can be paid out. Because there is no need to first register a project or perform identification verification, the procedure of getting money as a contributor is simplified. Paying out in cryptocurrency, on the other hand, necessitates a fiat-to-crypto conversion. To generate an exchange request, the balances of all registered contributors with a payout address are totalled weekly. The administrator in charge of transferring fiat to crypto receives an exchange request. The administrator enters the exchange rate into the system after the fiat money has been swapped. Each balance (in fiat money) is tied to a request for an exchange.

A "pull payment" mechanism is used to pay the registered user. Instead of creating transactions for users with a balance larger than zero on a regular basis, the balances are deposited into a SC on a regular basis (Smart Contract). Users can receive money by calling a method on the SC that triggers the transfer of monies. The transaction fee is passed on to the payment recipient when payments are pulled. Other platforms include payout transaction fees in their platform charge. However, because of the fanning at the project and contributor levels, a sponsor who supports five projects should pay a smaller platform charge than a sponsor who supports twenty projects. To take it a step further, a sponsor supporting a project with only one contributor should pay lower fees than a sponsor supporting a project with multiple contributors.

Further, using a smart contract increases transparency and security for the contributors concerning but not yet paid out balances. Even if a contributor decides to claim his/her balances only yearly (to reduce transaction costs), his/her funds are secured and only claimable by him/herself in the smart contract. Even though the pull payment process would also work without a smart contract, there was no guarantee that a donation platform is actually in possession of this money. Thus, using a smart contract with a pull payment process provides the optimal level of transparency and security from a user perspective.

The balances are stored using a mapping between **address** and **uint256**. Because arithmetic operations in Solidity "wrap" on overflow (*i.e.*, if an integer overflows, the most significant bits are lost), OpenZeppelin's SafeMath [29] library is used for all arithmetic operations. The SafeMath library throws an error if an overflow happens and thereby ensures that incorrect transactions are rolled back.

1) Fill: The *fill* updates the balances of addresses (*i.e.*, contributors) and can only be called by the owner of the contract. Thus, on SC deployment, the owner's address is stored in the _owner variable. This variable is used to ensure that only the owner can call the *fill* method. Further checks ensure that the input arrays addresses_ and balances_ have the same length, as each balance needs to be assigned to a specific address. The for-loop (line 20) updates the stored balances of the smart contract based on the two input arrays of the method. Additionally, the sum of all input balances is calculated in the loop.

2) Release: A user can claim the balances that are assigned to his/her address. In the form of gas, the transaction costs are thereby covered by the user who requests the funds. Before a transfer is initiated, it is verified that the balance of the requested address is greater than zero (line 33). To avoid race conditions when the function is called rapidly multiple times, the balance of the address is set to zero before the transfer is initiated. The order of lines 35 and 36 is important, as race conditions have already been exploited to hack the DAO, which eventually led to a hard fork of the Ethereum blockchain to restore stolen balances [13].

3) BalanceOf: Returns the balance of an address. Since the default value of possible mappings in Solidity is 0, the balanceOf method always returns a value, even if the requested address has never been registered in the SC.

d

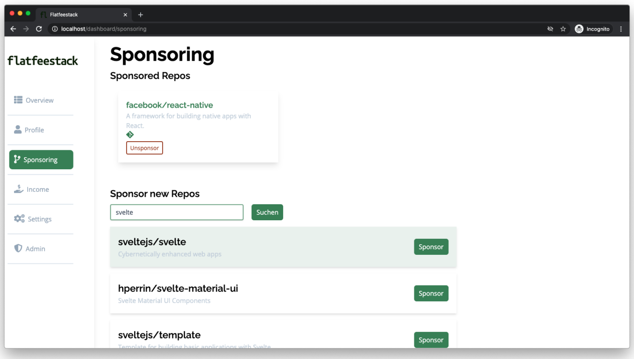
Flatfeestack
localhost/dashboard/sponsoring
Incognito
flatfeestack
Overview
Profile
Sponsoring
Income
Settings
Admin
Sponsoring
Sponsored Repos
facebook/react-native
A framework for building native apps with React.
Unsponsor
Sponsor new Repos
svelte
Suchen
sveltejs/svelte
Cybernetically enhanced web apps
Sponsor
hperrin/svelte-material-ui
Svelte Material UI Components
Sponsor
sveltejs/template
Template for building basic applications with Svelte
Sponsor

Fig. 6: Selection of repositories to sponsor [7]

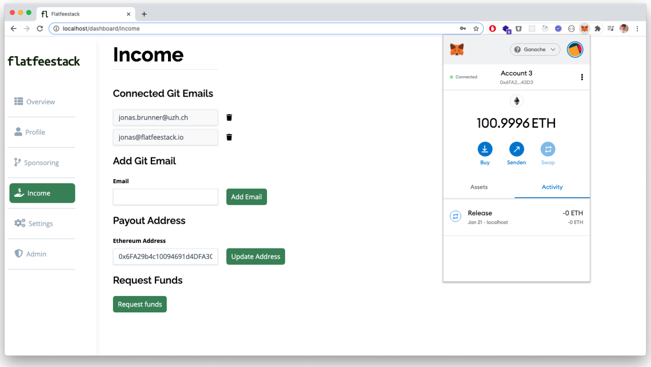

Fig. 7: Balance after release function has been called [7]

```
1   contract coreContract {
2       [...]
3       function fill(address[] memory addresses_,
            uint256[] memory balances_) public
            payable {
4           require(msg.sender == _owner, "Only the
                owner can add new payouts");
5           require(addresses_.length == balances_.
                length, "Addresses and balances
                array
6           must have the same length");
7           uint256 sum;
8           for (uint i=0; i < addresses_.length; i
                ++){
9               balances[addresses_[i]] = balances[
                    addresses_[i]].add(balances_[i])
                    ;
10              sum = sum.add(balances_[i]);
11          }
12          if(sum > msg.value) {
13              revert("Sum of balances is higher
                    than paid amount");
14          }
15      }
16      function release() public {
17          require(balances[msg.sender] > 0, "
                PaymentSplitter: account has no
                balance");
18          uint256 _balance = balances[msg.sender];
19          balances[msg.sender] = 0;
20          msg.sender.transfer(_balance);
21          emit PaymentReleased(msg.sender,
                _balance );
22      }
23      function balanceOf(address address_) public
            view returns (uint256){
24          return balances[address_];
25      }
26  }
```

Listing 1: Ethereum SC code available in [7]

Before a transfer is initiated, it is verified that the balance of the requested address is more significant than zero (line 17). To avoid race conditions, when the function is called rapidly multiple times, the balance of the address is set to zero before the transfer is initiated. The order of lines 20 and 21 is essential, since race conditions have already been exploited to hack the DAO, which eventually led to a hard fork of the Ethereum Blockchain to restore stolen balances [13].

An SC increases transparency and security for the contributors in terms of assigned but not yet paid out balances. Even if a contributor decides to claim his/her balances only yearly (to reduce transaction costs), his/her funds are secured and only claimable by him/herself in the SC. Even though the pull payment process would also work without an SC, there was no guarantee that a donation platform actually possessed this money. Thus, using an SC with a pull payment process provides the optimal level of transparency and security from a user perspective.

Although, the use of cryptocurrencies provides advantages in transparency and execution speed of payments, drawbacks in terms of costs remain (*cf.* further discussed in Section IV). After a successful payment/deposit of cryptocurrencies with an active subscription, a user can start to sponsor repositories (*cf.* Figure 6). The selection of projects happens independently of the payment and can be changed at any time.

From a contributors' point of view, only a few steps are required to receive a donation using the same registration and dashboard as the sponsors. The *Request funds* button (*cf.* Figure 7) uses Web3 [9] to claim the user's balance by calling the `release()` function of the SC. This requires the user to have a Web3 enabled browser or an installed extension such as MetaMask. A transaction to the SC is created by clicking on the button that can be reviewed and signed in a Web3 modal (*e.g.*, Metamask). Therefore, a user can claim the balance at any time from the SC. The MetaMask extension even presents an error if a user tries to claim a non-existing balance from the SC. After the release function has been called, the user receives the funds. A drawback is that the user needs to have ether to pay the transaction cost to claim his/her balances.

A solution to this problem is the use of the Ethereum Gas Station Network (GSN) [28]. Based on the GSN, a so-called *Paymaster* contract implements the business logic to decide which transactions to pay for. This enables, for example, the use case where the donation platform pays the transaction cost of the first claim. However, the use of an SC has already been shown to "burn" a significant amount of a sponsor's donation amount. With the fanning donation architecture, covering contributors' first transaction fee would thus result in an even more significant portion of the donation amount being used for transaction fees.

IV. Discussion and Limitations

`FlatFeeStack` argues that there is the need to provide financial incentives to foster the production and maintenance of open source software. In this regard, different platforms (*cf.* Section II) support open source projects at different stages (*e.g.*, to crowdfund or sponsor projects), wherein `FlatFeeStack` sustainably creates incentives for both, contributors and sponsors, based on a ranking of contributions per project.

A. Contribution Engine and Definition of Metrics

While the `FlatFeeStack` presents a hybrid and operational technical basis, a proportional distribution of donations fosters relevant contributions, it also challenges determining what constitutes “relevance". Once different projects evaluate metrics differently, each project's selection of metrics and related weights must go through an internal discussion and voting process to determine relevance. For example, a DAO (Decentralized Autonomous Organization) could be deployed within registered contributors of a project to vote on metrics, weights, and contributions most relevant to this project. However, such a decentralized decision-making model can be slow to converge to a decision effectively and cumbersome in the sense that the voting process could be manipulated [10]. An approach to mitigate such effects involves an election with representative project members in a permissioned DAO, resulting in transparent definition metrics and weights.

Furthermore, the fact that metrics and weights for distributing incentives are deployed on-chain might have a significant impact (*i.e.*, it is open and verifiable). On one hand, knowing which type of contribution is relevant to distribute incentives proportionally is relevant. On the other hand, contributions may be purposefully targeted to maximize contributors' profits, while other actions (such as reviews, comments, or bug fixing) may be ignored, since they do not necessarily maximize profit. In this sense, the `FlatFeeStack` proposed enables projects to reach their definition of relevance through the definition of metrics and weights, and provides the technicalities of this platform.

B. Crypto Payout and Transaction Fees

Enabling payout in cryptocurrencies has advantages and disadvantages. While, on the one hand, it is possible to simplify the sponsoring process by reducing intermediaries (and additional fees) for paying contributors, on the other hand, high transaction fees can impair its operation. For example, to claim funds, a contributor first needs funds to pay the transaction costs, significantly increasing the entry barrier to receiving donations.

Implementing an SC provides transparency for contributors when it comes to amounts that have been allotted but not yet paid out. Even if a contributor chooses to claim his or her balances just once a year (to save transaction fees), his or her assets remain safe and only claimable by the contributor. However, `FlatFeeStack` cannot be fully decentralized once it integrates traditional payment services, which need to be handled in a centralized way by a payment service provider (*e.g.*, Stripe or PayPal), typically involving fees split into two parts, *(i)* a fixed amount per transaction and *(ii)* a percentage of the transaction amount. Thus, `FlatFeeStack` implements a hybrid-architecture providing crypto payments as an alternative as outline in Table I.

In the case of crypto payment, each request made to the Smart Contract (SC) must be paid by the contributor to verify and request payments (*e.g.*, by calling the `release()` method where a user requests to retrieve the balance). In this regard, the use of a blockchain platform with lower transaction fees (in contrast to Ethereum), such as the PoS-based (Proof-of-Stake) Tezos [18], is needed to reduce the operational costs required to run the SC's logic (*e.g.*, contribution fanning and payout aggregation) whose fees would be shifted toward the contributor requesting the payment. Therefore, using a smart contract could potentially “burn" a significant amount of a sponsor's donation amount.

Still, the transaction fees of blockchains are significantly lower than automated bank transfers (considering PoS-based chains). For example, Flattr [14] charges 3$ per payout and the Stripe connect platform [36] charges fees within the same range (2$/month per user + (0.25% + 25c) per payout). Furthermore, since crypto payments are final, there is no need for a lengthy onboarding process with identity verification, and there is no need to deal with denied or returned payouts. The usage of cryptocurrencies for payment is limited because there is no method to set up recurring payments in the form of a subscription.

With the upgrade to Ethereum 2.0 and the transition to PoS [1], fees in the Ethereum network could potentially develop more toward low fees (such as Tezos [18]), making Ethereum a solid choice in the future. Also, there is an additional configuration effort outside the donation platform for users who have never used cryptocurrencies. However, creating a new Ethereum account is comparable to the effort of full customer identification (Know Your Customer, KYC) required for other donation platforms.

V. Summary and Future Work

`FlatFeeStack` is an open-source (code available in [7] and front-end at [8]) donation platform that provides a new optimistic sponsoring approach, where sponsoring and receiving donations for open source projects can be achieved. Key contributions are *(i)* an on-chain sponsoring model based on a flat fee subscription and multipliers, *(ii)* a contribution engine distributing donations based on a contribution ranking, and *(iii)* payout in cryptocurrencies simplifying the payment flow and reducing intermediaries fees. While *(i)* concerns a change in the sponsoring business logic to incentivize further corporate and individual sponsors, *(ii)* and *(iii)* provide the technical support to incentivize further contributors making a statement about the developers' contribution and to simplify the payment and receiving of donations.

Future work includes improving the availability of metrics by extracting platform-specific information that can be assessed and applied to all repositories, regardless of language or technology attributed to individual contributors. For example, considering the use of tags determined (*e.g.*, via DAO) by projects themselves, adjusting weights in the contribution calculation will be possible. Concerning `FlatFeeStack`'s pull payment, an approach using an EVM-based SC can allow for a significant reduction of fees in contrast to automated bank transfers. Transaction fees must still be entirely shifted toward contributors to keep sponsor fees low and enable the fanning-out donation architecture. In this regard, next steps will address a fee optimization of payouts for other Blockchains (*e.g.*, Binance, Tezos, or NEO), because with such a fanning out donation architecture the ratio of payouts to sponsors is significantly higher than on existing platforms.

References

[1] "Upgrading Ethereum to radical new heights," https://ethereum.org/en/eth2/, (Last accessed December 2020).

[2] Binance, "Binance Smart Chain," Dec. 2021, https://www.binance.org.

[3] L. Brennan, W. Binney, and E. Brady, "The Raising of Corporate Sponsorship: A Behavioral Study," *Journal of Nonprofit & Public Sector Marketing*, Vol. 24, No. 3, pp. 222–237, 2012.

[4] V. Buterin, "Review of Optimism Retro Funding Round 1," Nov. 2021, https://bit.ly/3DAdRnm.

[5] BuyMeACoffee, "A Supporter is Worth a Thousand Followers," Nov. 2021, https://bit.ly/31N4xj6.

[6] G. Cecere, F. Le Guel, and F. Rochelandet, "Crowdfunding and Social Influence: An Empirical Investigation," *Applied Economics*, Vol. 49, No. 57, pp. 5802–5813, 2017.

[7] Flatfeestack, "Flatfeestack GitHub," Dec. 2021, https://github.com/flatfeestack.

[8] FlatFeeStack, "Flatfeestack - On the Shoulders of Giants," Dec. 2021, https://flatfeestack.io/.

[9] ChainSafe, "ChainSafe/web3.js," https://bit.ly/3DBIQjd, (Last accessed December 2021).

[10] U. W. Chohan, "The Decentralized Autonomous Organization and Governance Issues," *Available at SSRN 3082055*, 2017.

[11] 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*, ser. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 186–196. [Online]. Available: https://bit.ly/3oyaadz

[12] M. Dorner, M. Capraro, and A. Barcomb, "Quo Vadis, Open Source? The Limits of Open Source Growth," *arXiv preprint arXiv:2008.07753*, 2020.

[13] S. Falkon, "The Story of the DAO – Its History and Consequences," 2017, https://bit.ly/3lNobSZ, (Last accessed January 2021).

[14] Flattr, "Support Creators with One Easy Subscription or With One-time Contributions," Nov. 2021, https://flattr.com/.

[15] M. M. Gierczak, U. Bretschneider, P. Haas, I. Blohm, and J. M. Leimeister, "Crowdfunding: Outlining the New era of Fundraising," in *Crowdfunding in Europe*. Springer, 2016, pp. 7–23.

[16] Gitcoin, "Gitcoin: Build and Fund the Open Web Together," Nov. 2021, https://gitcoin.co/.

[17] GitHubSponsors, "Invest in the Software the Powers Your World," Nov. 2021, https://bit.ly/3dueKmP.

[18] L. M. Goodman, "Tezos - A Self-Amending Crypto-Ledger," September 2014, https://tezos.com/whitepaper.pdf.

[19] Goteo, "Crowdfunding the Commons," Nov. 2021, https://en.goteo.org.

[20] I.-H. Hann, J. Roberts, S. Slaughter, and R. Fielding, "Economic Incentives for Participating in Open Source Software Projects," *ICIS 2002 Proceedings*, p. 33, 2002.

[21] ——, "Why do Developers Contribute to Open Source Projects? First Evidence of Economic Incentives," in *2nd Workshop on Open Source Software Engineering, Orlando, FL*, 2002.

[22] Issuehunt, "Where Open Source Grows," https://issuehunt.io/, (Last accessed January 2022).

[23] Ko-Fi, "Make an Income Doing What You Love," Nov. 2021, https://ko-fi.com/.

[24] S. Krishnamurthy, S. Ou, and A. K. Tripathi, "Acceptance of Monetary Rewards in Open Source Software Development," *Research Policy*, Vol. 43, No. 4, pp. 632–644, 2014.

[25] ——, "Acceptance of Monetary Rewards in Open Source Software Development," *Research Policy*, Vol. 43, No. 4, pp. 632–644, 2014.

[26] Liberapay, "Liberapay Recurrent Donations Platform," Nov. 2021, https://bit.ly/3DyiTAL.

[27] Open Collective, "Make Your Community Sustainable," https://opencollective.com/, (Last accessed January 2022).

[28] OpenGSN, "Gas Station Network," Dec. 2021, https://opengsn.org/.

[29] OpenZeppelin, "SafeMath.sol," Dec. 2021, https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.so.

[30] OptimismPBC, "Retroactive Public Goods Funding," Jul. 2021, https://bit.ly/3oA5rZa.

[31] C. Overney, "Hanging by the Thread: an Empirical Study of Donations in Open Source," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, 2020, pp. 131–133.

[32] C. Overney, J. Meinicke, C. Kästner, and B. Vasilescu, "How to Not Get Rich: An Empirical Study of Donations in Open Source," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1209–1221.

[33] R. M. Parizi, P. Spoletini, and A. Singh, "Measuring Team Members' Contributions in Software Engineering Projects using Git-driven Technology," in *IEEE Frontiers in Education Conference (FIE 2018)*, San Jose, CA, USA, USA, October 2018, pp. 1–5.

[34] Patreon, "Change the Way Art is Valued," Nov. 2021, https://www.patreon.com/.

[35] B. Rodrigues, T. Bocek, and B. Stiller, "The Use of Blockchains: Application-driven Analysis of Applicability," in *Advances in Computers*. Elsevier, 2018, Vol. 111, pp. 163–198.

[36] Stripe, "Stripe Connect: Online Marketplace Payments Platform," https://stripe.com/connect, (Last accessed January 2022).

[37] Tidelift, "A Managed Open Source Subscription Backed by Creators and Maintainers," https://tidelift.com/, (Last accessed January 2022).

[38] G. Wood *et al.*, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum Project Yellow Paper*, Vol. 151, No. 2014, pp. 1–32, 2014.

[39] X. Zhang, T. Wang, Y. Yu, Q. Zeng, Z. Li, and H. Wang, "Who, What, Why and How? Towards the Monetary Incentive in Crowd Collaboration: A Case Study of Github's Sponsor Mechanism," 2021, https://bit.ly/335h2qF.