The Labor Ledger: a blockchain solution for allotment of rights on jointly developed projects

Paper v.1.0 (29.06.2020)

Vadim Konstantinov, creator of the Labor Ledger
vadim.konstantinov@gmail.com
Vasily Sumanov, researcher
v.d.sumanov@gmail.com
Michael Klimusha, advisor
miklimusha@hotmail.com

Introduction **2**

Foundations of the Labor Ledger **2**

The Labor Ledger logic in a nutshell **4**

The Labor Ledger: use cases **8**

Conclusion **8**

Appendix 1: Pollen.id and Audacity.id 10

Appendix 2: Smart Contracts description 11

Introduction

The problem of fair proprietorship allocation for jointly developed projects is as old as time immemorial. Historically, this was solved by legal agreements, and financial compensation in the form of salaries, bonuses and company shares or in conditional format, such as employee share options. The latter approach is widely used in the startup, tech, and financial sectors. Web3.0 addresses this problem differently: the development of projects has become distributed, and co-founders and employees often do not even meet each other personally. In many cases legal agreements are ineffective. Most work is now delivered and accounted for in the digital form, so it's quite logical to create a special digital tool for this purpose. The decentralized autonomous organization (DAO) concept, which is widely explored and tested acts as a catalyst: such organizations require mechanisms for fair contribution measurement and rights allocation.

One of the most known initiatives in this space is SourceCred.io - a tool for communities for measuring and rewarding value creation. It is for tracking member contributions in software development projects. And has been tracking the results of the prototype project since early 2018[1]. The model tracks Github repo activity, posts in communication channels, and reactions to them (such as likes). This tool is integrated with Aragon open-source software platform[2] that allows for the creation and management of decentralized organizations and disputes resolution (The Aragon Network Jurisdiction, etc.). SourceCred tracks a lot of variables to calculate contributions of participants. But, for the majority of real-world projects such algorithms can be too complex and not relevant to the actual value creation process. For example, a lot of projects are not based on a program code and GitHub commit indicators cannot help them to track contributions practically.

This paper is devoted to an overview of the Labor Ledger - a solution built on top of Ethereum, which is being developed for accounting contributions to jointly developed projects. It is not aimed to compete with SourceCred, AraCred, or any other project with similar principles.

Foundations of the Labor Ledger

What is the Labor Ledger?

The Labor Ledger is a framework of Ethereum smart contracts, tools for developers, and a legal concept which allow parties, who are jointly developing a project (a “collaboration”), to register and possess legally tangible “stakes” in the project (essentially, “rights to share deliverables of the project”) resulting from and depending on their labor inputs and other contributions.

[1] https://cred.sourcecred.io/timeline/@sourcecred/
[2] https://aracred.github.io/website/

The name “Labor Ledger” is given as a reference to the central smart contract of the framework.

Specifically, the Labor Ledger functionality is focused on:
- (1) making a member "inputs" in a project be accurately and fairly accounted for;
- (2) making rules of conversions of “inputs” into “stakes” be clear and transparent;
- (3) making members "stakes" in a project be auditable and reliably registered;
- (4) vesting rights of the members based on their "stakes" (making them legally tangible, enforceable, rules compliant);
- (5) facilitating execution of rights on a "stake" (i.e selling, pledging, etc.).

The Labor Ledger archives it trough:
1. Tokenization (with Project Tokens) of “stakes” (rights) in a “collaboration” (project);
2. Fair and reliable registration of collaborators inputs into the project (mainly, labor inputs, but not limited by that);
3. Providing trust-free mechanisms to convert “inputs” into “stakes”;
4. Auditable and transparent registration of transactions with the “inputs” and “stakes”;
5. Governance mechanics (similar to DAO).

Although the Labor Ledger may serve extreme followers of the “code is a law" paradigm who preach absolute freedom from legislation and regulators, the facilitation of legally compliant “collaborations” and legally tangible “stakes” in “collaborations” (i.e. rights on deliverables of joint efforts) is among the main objectives of the Labor Ledger design.

The document focuses on technical aspects, leaving the legal concept out of the scope.

What do we call a “collaboration”?

Collaboration is a project being jointly developed by a team of like-minded “collaborators” (“co-workers” or “members”). Such development (as it often occurs in the Web3.0 economy) is more and more often undergoing without formal incorporation of any legal entity.

What do we mean by a “stake” (in a collaboration)?

We mean a share in the Intellectual Property and other present/future deliverables created in the course of collaboration.

What kind of “contributions” are supported?

Although the Labor Ledger is primarily focused on labor hours (time which each particular collaborator works over the project), alternative types of inputs (contributions) are also supported, such as:

- Fixed initial shares in a project (for example, 10% share for a founder);
- Direct financial investments into a project (e.g. in form of ETH, DAI, ERC20 tokens - normally, have to be announced and pre-approved by the existing holders of the Project Tokens);

- Meeting of certain conditions and benchmarks (for example, publication of an article or a piece of code, implementation of security audit or other services for the project). It works like an escrow smart contract or, to some extent, as an irrevocable documentary letter of credit having a number of underlying Project Tokens deposited to be transferred to a predefined beneficiaries' account(s) if the on-chain conditions are met or the "arbiters" ("oracles") confirm off-chain conditions.

To properly handle different inputs into a project, the Labor Ledger smart contract is properly integrated with other smart contracts, such as the Project Token contract and the Tokens-for-Labor contract(s).

The Labor Ledger logic in a nutshell

The Labor Ledger is designed to track collaborators contributions and rights arising from such contributions.
The "Project Tokens" and "Labor Units" conception is in the center of the system.

What is the "Project Token"?

It is an ERC-20 compatible token that represents the "stake" in a project (i.e. a certain portion of tokenized rights on the project IP and other deliverables).

In other words, the Project Tokens, essentially, represent rights in regards to results and values a project creates, such as the Intellectual Property or utility tokens.

Each project issues its own Project Token and this token is different from tokens which any other Project issues.

For example, in a project developing a new blockchain protocol with a new token ("utility tokens"), the Project Tokens are used to reward co-workers on a stage prior to the protocol launch and initial distribution of its utility tokens. As soon as the utility tokens get minted, the Project Tokens can be exchanged to the utility tokens from the project team pool.

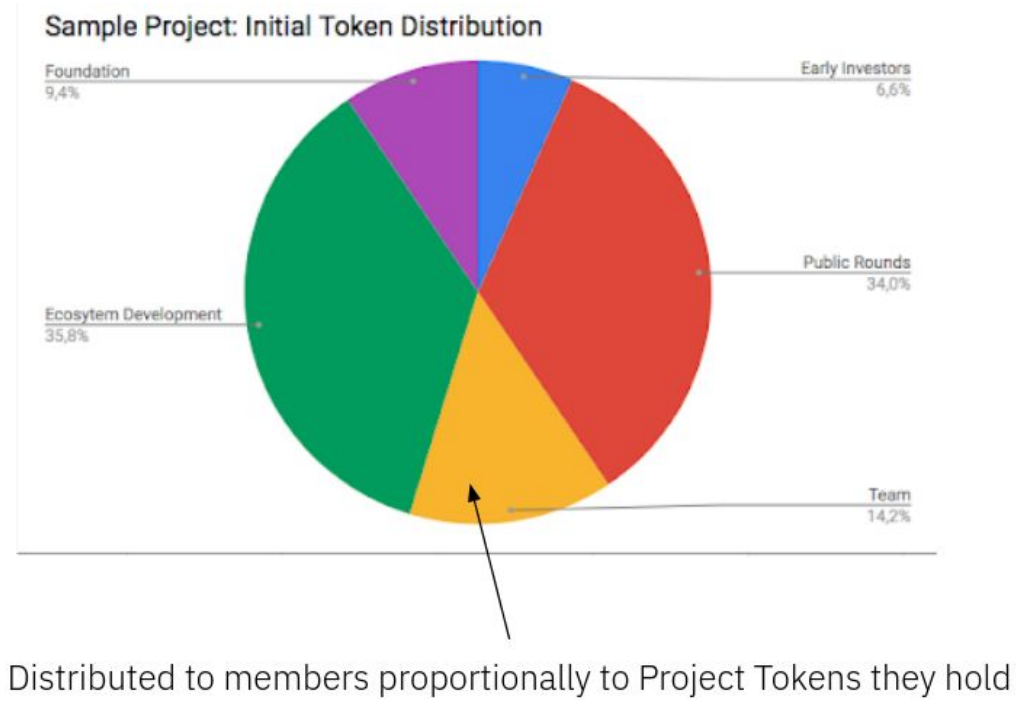

Fig 1. The relation of the Project Tokens and utility tokens the project creates.

The Project Tokens transfers between holders may be restricted (by a whitelist of co-workers and other members of the collaboration; but this option can be altered ot disabled).

How can a collaborator get and use Project Tokens?

Each project collaborator can receive Project Tokens in three ways:

1. Be a member of Founders team, to whom some share of the Project Tokens is initially allocated;
2. Contribute ETH/DAI or any other accepted assets to the project, if the current holders of the Project Tokens opted for such an option (and defined a rate at which crypto-coins will be converted to tokens);
3. Contribute labor into the project (provide useful work) to get Labor Units as a reward, and then exchange Labor Units to Project Tokens.

Later,we will discuss the third way, which is the core of the Labor Ledger system. Each worker gets Labor Units by submitting Time Units (time worked) to the system.

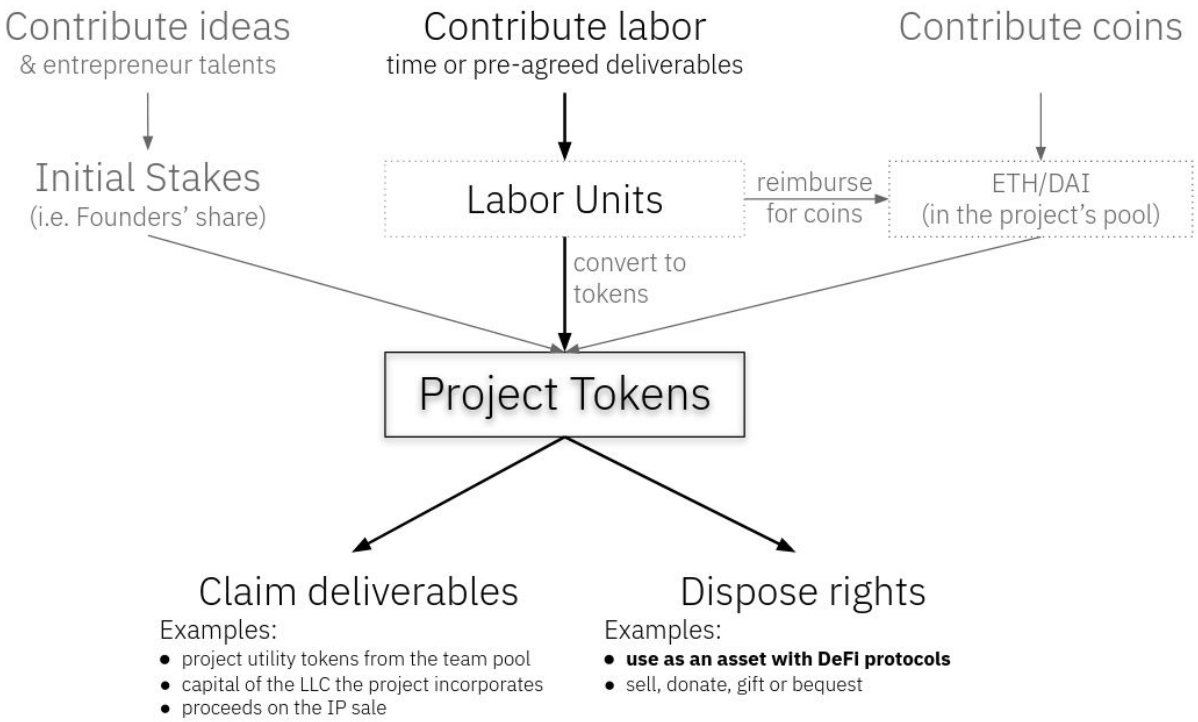

Fig 2. The ways to receive Project Tokens and possible operations with them.

Project Tokens may be used to claim a share of the deliverables the project created or dispose rights in other ways (e.g. sell or donate).

What is the "Labor Unit"?

The Labor Unit is a measure and a "proof" of some work a member did developing a project.

The Labor Units of a member are calculated as the time worked (measured in "Time Units") multiplied by the "weight" (a "seniority" - one of standard labor "prices" the project uses) assigned to the member on joining the project:

$$Labor\ Units\ =\ Time\ Units \times Weight$$

Labor Units grant to a member neither direct rights to claim the project deliverables nor project governance rights (but can be used, in the future versions, as staking assets in team disputes).

How can a collaborator get and use Labor Units?

A member "acquires" Labor Units submitting to smart contracts the labor hours ("Time Units") she worked for the project.

Besides the Labor Units, granted for Time Units, there is an option to receive Labor Units in exchange for a certain agreed work, for example publication of articles or completing other provable jobs.

A member may exchange their “accepted” Labor Units (explained further) into Project Tokens via a smart contract at a certain transparent and pre-agreed rate.

Holders of the Project Tokens would normally want to adjust this rate time after time to account for increasing value as the project progresses.

The smart contract guarantees the fixed exchange rate for a certain period of time and for a certain amount of Project Tokens deposited with the smart contract.

Alternatively, a member may “redeem” his Labor Units for coins should Project Tokens holders decide to provide this option and the collaboration has coins available for that.

The rate for Labor Units in coins tends to depend upon labor market prices.

Just like with Project Token exchange, a smart contract guarantees the fixed (or calculated by a known formula) redemption rate for a certain period of time and for a certain amount of coins deposited with the smart contract.

Labor “acceptance” and “pending” Time Units

Labor Ledger design doesn't provide for a “centralized authority” that decides if Time Units a member reports are fair or not. Instead, the project team is self-regulatory and lets the members themselves decide on it.

Our vision is that only teammates (co-workers) have adequate information (they actively participate in project development) and incentivization (overreported time dilutes their labor value) to accurately measure Time Units reported.

The beta version of the Labor Ledger defines three roles participating in this process:
1. Members (co-worker);
2. The Project Lead;
3. The Project Arbiter(s).

During three weeks followed by the submission, other members (co-workers) can initiate a dispute if they disagree with the Time Units reported.

First two weeks the Project Lead can correct the number, but in the third week only the Project Arbiter can do that.

So, Time Units, which members report to the system get “accepted” in four weeks after the submission, remaining “pending” in the meanwhile. This is called“submission aging”.

A member may submit Time Units for a week ended not later than 4 weeks prior to submission.

In the consequent versions, any member (co-worker) will be able to offer an adjustment to the Time Units number another member submits (to “open a dispute”). To do that, the member

offering the adjustment shall stake some of his Labor Units. After a dispute is initiated, other project participants may vote, staking their Labor Units.

Members will be rewarded or penalized with a part of staked Labor Units for a won or lost dispute. Members will be able to delegate Labor Units to someone they trust.

Such design provides more voting power to members who already have contributed a significant amount of work, and limits Founders with big stakes in Project Tokens to over-influence this process.

The Labor Ledger: use cases

The main use case of the Labor Ledger is to account contributions of distributed team members on early stages of a project development.

Often, projects are started as initiatives of several enthusiasts and when such projects mature, team tokens can become “a bone of contention”. Initial agreements, if they are not embedded into the code can be violated as a result that somebody thinks that he worked more than the other. No doubts, such situations often triggered an endless vicious circle of hostilities.

The goal of Web3.0 is to create an environment, where an individual does not need to trust anyone. And it is true: the “code is a law” principle is being already implemented in the financial sector (DeFi) and decentralized organizations (DAOs). Now we lack an instrument that allows us to feel safe and protected when it comes to investing your personal time. Time is a much more limited resource than money, so we need to treat it with due care and be able to get adequately rewarded for every piece of properly completed job.

So, the first and the most general use case of the Labor Ledger is transparent and reliable registration of teammates contributions to jointly developed projects and protecting their rights and claims on the values the project generates.

The other use cases are connected with rapidly developing DeFi protocols. For example, the Project Tokens (as de-facto an option to future utility tokens) can be used as a collateral[3] for DeFi products.

Conclusion

This document provides a brief overview for the Labor Ledger system.

Our goal is to create a safe and reliable mechanism for collaboration in the Web3.0 era.

The Labor Ledger accounts Founders shares, direct investments and contribution to the project based on work.

[3] Here we are talking about Project Tokens of those projects, which are reliable and trusted by the community.

Accounting of work contribution is based on Time Units (one option) and acceptance of a predefined piece of work with predefined value attributed to it (another option).

All work contributions are to be verified and accepted by the community of actual incumbent co-workers, which already have their work contributions accepted to this project in the past.

In our opinion, systems like the Labor Ledger are necessary tools for organizing, managing and protecting time investments in a rapidly developing decentralized world.

The projects which apply the Labor Ledger in its operation is presented in Appendix 1.

Appendix 1: Pollen.io and Audacity.id

Audacity.id is a SaaS platform that allows to create and operate an "Internet Company" on-line.

It integrates the Labor Ledger into its SaaS platform, and it first applies the Labor Ledger to track contributions of its team members.

Pollen.id is a first decentralized asset pool that plans to add the Project Token of the Audacity team into its asset portfolio.

Appendix 2: Smart Contracts description

Units & tokens

Smart Contracts support the following units and tokens:

Time Units

Working hours, measured in 5-minutes units to avoid floating-point operations.

Time Units may be:

- "accepted" and "pending" (explained further);
- "used" only via automatic "conversion" into *Labor Units*.

Note: *Time Units* do not provide rights of voting (on project development issues).

Labor Units

Labor Units are working hours weighted with a member "weight", i.e. *Time Units* multiplied by the "weight" of the labor value pre-assigned to a member.

To avoid floating-point operations, weights are integers -
instead of the weight of 1.5 for the "Senior" role and 1.0 for the "Standard" role, the weight of 3 is assigned for the "Senior" and 2 to the "Standard" roles.

Labor Units are "accepted" (rather than "pending") as soon as the *Time Units* are accepted.

"Accepted" *Labor Units* may be "spent" for (i.e. exchanged into) *Project Tokens*, if the *Tokens-for-Labor* smart contract is active, and/or into ETH/DAI coins, if *Coins-for-Labor* smart contract(s) is (are) active.

In the upcoming versions, we plan to introduce functions of "staking" of *Labor Units* and rewards/penalties for opening "disputes" and acting in *Project Arbiter(s)* and other roles.

Note: *Labor Units* do not provide rights of voting on project development issues.

Project Tokens

ERC-20 compatible tokens, representing 'de jure' rights (participation) in the project.

ERC-20 functionality extended to impose some limitations (like authorized token holders) on token transfers between accounts.

Note: *Project Tokens* **do** provide rights of voting on project development issues.

Roles

Smart contracts support the following roles:

1. *Member* - may submit labor time and exchange *Labor Units* into *Project Tokens* and/or coins;

2. *Project Lead* - can set the member "status", the member "weight" (once only), maximum working hours allowed for a week, may adjust (decrease) "pending" *Time Units* of a member;

3. *Project Arbiter(s)* - can alter (decrease or cancel) the *Project Lead*'s adjustments of *Time Units*, and alter a labor weight previously assigned to a member;

4. *Inviter(s)* - can invite new member;

5. *Admin* - can manage migration to new versions of smart contracts;

6. *Default Operator* - can submit transactions on behalf of any other user (with any role) unless the user explicitly revokes the allowance;

7. *Operator(s)* - any user can assign an operator(s) able to submit transactions on behalf of the user;

8. *Quorum* - can mint and distribute *Project Tokens*, create *Tokens-for-Labor*, *Coins-for-Labor*, *Tokens-for-coins, Token-escrow* and *Labor-escrow* smart contracts, assign/denounce the *Project Lead*, the *Inviter(s)*, the *Project Arbiter(s)*

9. *Treasurer(s)* - can send transactions (pay coins) from the *Treasury Wallet*

10. *Investors* - can buy *Project Tokens* for DAI/ETH coins, and other coins in the future versions of smart contracts, via *Tokens-for-coins* smart contracts.

Smart Contracts

***Note**: the below mentioned contracts use the "proxy-implementation" design pattern. Technically, there are two smart contracts (the proxy and the implementation) for each of them.*

Collaboration

- orchestrates other contracts;
- allows the *Quorum* to mint and distribute Project Tokens, deploy new instances of *Tokens-for-Labor*, *Coins-for-Labor*, *Tokens-for-coins, Token-escrow* and *Labor-escrow* smart contracts;
- validates transfer of a Project Token from one holder to another holder.

Labor-Ledger

- registers members;
- accounts for labor contributions;
- registers *Time Units* and *Labor Units*;
- processes "pending" and "accepted" units, accounts for "adjustments".

Project Token

- implements the Project Token;
- extends the ERC-20 functionality by limitations on token transfers;
- allows the *Collaboration* contract to "mint" the tokens and validate token transfers.

Tokens-for-Labor

- allows members to "exchange" *Labor Units* into *Project Tokens* deposited with the contract, within a certain period of time, at a fixed rate;
- queries the *Collaboration* contract if a member has enough Labor Units;
- allows the *Collaboration* contract to withdraw back remaining Project Tokens after the time passes.

Coins-for-Labor

- allows members to "redeem" their Labor Units for DAI/ETH deposited with the contract, within a certain period of time, at a fixed rate;
- queries the *Collaboration* contract if a member has enough Labor Units;
- allows the *Collaboration* contract to withdraw back remaining coins after the time passes.

Tokens-for-coins

- allows *Investors* to contribute DAI/ETH (or other accepted tokens) for *Project Tokens,* within a certain period of time, at a fixed rate;
- allows the *Collaboration* contract to withdraw back remaining tokens after the time passes.

Token-Escrow

- allows the *Collaboration* contract to "lock" (deposit) some amount of Project Tokens for a certain beneficiary, for a certain period of time;
- "unlocks" the tokens for the beneficiary against confirmations by pre-defined "arbiters" (or "oracles") that some pre-agreed conditions have been satisfied;
- allows the beneficiary to withdraw the unlocked tokens;
- allows the *Collaboration* contract to withdraw back unclaimed tokens after the time passes.

Labor-Escrow

- against confirmations by "arbiters" (or "oracles") that some pre-agreed task is fulfilled, registers with the *LaborLedger* contract, via the *Collaboration* contract, a certain amount of Labor Units to a member;
- allows the *Collaboration* contract to set the amount of Labor Units, "arbiters" (or "oracles"), and a period of time the contract is active.

Treasury Wallet

- multisig wallet keeping coins (DAI/ETH) of the project.

Admin-Proxy

- implements the *Admin* role.

Other contracts (WIP)

- *Quorum* contract - a multisig and "voting" smart contract, currently, the *Project Lead*'s EOA[4];
- *Arbiter* contract - a multisig smart contract, currently an EOA of a member acting in the *Arbiter* role;
- *Member* contract - a member's EOA, or, in the upcoming versions, a proxy/identity smart contract of the member.

Time acceptance and adjustments

The *Labor-Ledger* smart contract implements "aging" of time submission as follows.

1. A member may submit hours (i.e. *Time Units*) for a week ended no later than 4 weeks ago (e.g., for weeks #14 …#17 should a member submit hours on the week #18)

2. *Time Units* get converted into *Labor Units* no sooner than in three weeks after the submission week.
   (e.g. on the week #21 should a member submit the week #17 on the week #18)
   During these three weeks the member's *Time Units* (and future *Labor Units*) are "pending". After that, with adjustments mentioned below, they are "accepted".
   Note: member rights, i.e. voting, are based on the *Project Tokens*. Only the "accepted" *Labor Units* may be exchanged into the *Project Tokens*.

3. During two weeks following the submission week, the *Project Lead* may make an adjustment - decrease the *Time Units* submitted by a member.

4. During the 3rd week following the submission week, (any of) the *Project Arbiter(s)* may decrease or cancel the *Project Lead*'s adjustments.

5. The adjustments noted above alter (decrease) *Time Units* (and hence - the *Labor Units*) of members.

The *Project Arbiter* acts as a moderator.
A teammate (or a few teammates collectively, via a multisig smart contract) may act in the *Project Arbiter* role.

Example:

A member submits 160 hours for a week.

The *Project Lead* believes it is too much and submits the adjustment for minus 80 hours (thus

[4] "EOA" stands for an "Externally Owned Account", which is, unlike an account a smart contract control, is an account controlled by a user.

striking out 50% of the member hours).

The member does not agree with the adjustment and opens a dispute applying to the *Project Arbiter*.

The *Project Arbiter* considers the issue and lowers the adjustment to minus 40 hours. So the member will finally have 120 hours accepted for the week.

Contract interaction

The following example illustrates contract interaction.

- Founders decide to issue 1 million Project Tokens, of which -
    - (a) 0.2M to the founders team;
    - (b) 0.1M to a member contributing some IP rights;
    - (c) 0.5M for members contributing labor (“co-workers”);
    - (d) up to 0.2M for possible contributors in form of ETH/DAI (“investors”);
- Founders initially anticipate 1000 Labor Units needed for a beta version *(let’s assume 1 Labor Units is equivalent to 1 hour of work fairly priced at 20$/hr)* and another 500 Labor Units needed to tune the beta version into an MVP;
- Founders believe investors will join the project as soon as the beta version is ready, at a price for a Project Token expressed in ETH/DAI to be defined as soon as the beta version is ready;
- Founders suggest Project Tokens from the pool (c) to contributors (co-workers) into the beta version, and ETH/DAI from investors, when the beta is ready, for co-workers upon the MVP;
- As it turns out later, the beta version needs 400 more Labor Units to be completed;
- The Project Tokens holders - founders and contributors of the IP and labor hours - decide to issue additional 0.1M Project Tokens to cover the additional 400 Labor Units;
- When the beta is ready, Project Tokens holders and ETH/DAI contributors/investors define the rate at 125 Project Tokens for 1 DAI regarding to 0.125M Project Tokens (from the pool (d));
- Co-workers contribute 500 Labor Units to complete the MVP and receive 10.000 DAI for it.

It results in the following **transactions flows**.

1. The *Quorum* account, or a contract, calls the *Collaboration* contract to mine 1M *Project Tokens* distributing them as follows:

- 0.2M to the accounts of the founders team members;
- 0.8M tokens to the *Collaboration* contract address.

The *Collaboration* smart contract deploys a new instance (the "proxy") of the *Project Token* smart contract and calls it. The latest mints and transfers tokens to the addresses of the founders team members and the address of the *Collaboration*.

2. The *Quorum* calls the *Collaboration* contract to deploy new contracts:
    - *Tokens-for-Labor* contract, in respect to 0.5M Project Tokens of the pool (c), with the rate 500 tokens for one Labor Units;
    - *Tokens-escrow* contract, in respect to 0.1M Project Tokens of the pool (b), with the (off-chain) condition the oracles shall confirm (on-chain) being "the IP has been contributed".

    The *Collaboration* contract creates new instances ("proxies") of the said contracts, and transfers 0.5M and 0.1M Project Tokens from its own address to the addresses of the two new contracts.

3. Member accounts, directly or via "*operators*", call the *LaborLedger* contract to register time worked for the project and receive *Labor Units*.

4. Member accounts, directly or via "*operators*", call the *Tokens-for-Labor* contract to exchange *Labor Units* into *Project Tokens*, until all deposited 0.5M Project Tokens are exchanged.

    Every time, the *Tokens-for-Labor* contract calls the *Collaboration* contract to get approval and, if approved (a member has enough *Labor Units*) and there are enough *Project Tokens* in the *Tokens-for-Labor* address, it sends the *Project Tokens* to the member account.

    The *Collaboration* contract calls the *Labor-Ledger* contract to check available *Labor Units* and register the "settled" (or "used") *Labor Units* of the member.

5. The *Quorum* calls the *Collaboration* contract to deploy a new instance of the *Tokens-for-labor* contract. This time for 0.1M Project Tokens to exchange with the rate of 250 Project Tokens for the Labor Unit.

    The *Collaboration* contract deploys the new contracts and transfers Project Tokens from its own address to the new contract address.

6. Member accounts call the *LaborLedger* contract and the (second instance of the) *Tokens-for-Labor* contract to get Project Tokens from the pool (c), in the same way as described above.

7. The *Quorum* calls the *Collaboration* contract to deploy a new *Tokens-for-coins* contract, in respect to 0.125M Project Tokens with the rate of 125 tokens per DAI.

The *Collaboration* contract creates a new instance (the "proxy") of the *Tokens-for-coins* contract, and transfers 0.125M tokens from its address to the new contract.

8. Investors send 10.000 DAIs to the *Tokens-for-coins* contract.
   The *Tokens-for-coins* contract calls the *Collaboration* contract to get the approval of token transfer to an investor address and, if approved, the contract transfers *Project Tokens* from its own address to the investor, calling 'transfer' method on the *Project Token* contract, and DAI to the *treasury wallet* contract.

9. The *Quorum* calls the *Collaboration* contract to deploy a new *Coins-for-Labor* contract with the rate set to 20 DAIs per 1 Labor Unit.

   The *Collaboration* contract creates a new instance (the "proxy") of the *Coins-for-Labor* contract.

10. The project treasurers send 10.000 DAI to the address of the *Coins-for-Labor* contract.
    The *Coins-for-Labor* contract is ready to redeem Labor Units.

11. Member accounts call the *LaborLedger* contract to get Labor Units for the work over the MVP, and the contracts work in the same way as described above.

    This time, instead of calling the *Tokens-for-labor* contract, members call the *Coins-for-Labor* contract to exchange their Labor Units.

    The *Coins-for-Labor* smart contract calls the *Collaboration* smart contract to get transaction approval and, if approved, sends DAI to the member.

    The *Collaboration* contract calls the *Labor-Ledger* contract to check available *Labor Units* and register the "settled" *Labor Units* of the member.