DAOstack

The Operating System for DAOs

White Paper V1.0

October, 23rd

***** WORK IN PROGRESS *****

Comments are very welcome

fmatan@gmail.com

Abstract

DAOstack is an operating system for a new form of web organization: the DAO. DAOs have captured the imagination of the best minds in the blockchain space, but despite their promise have yet remained an abstract idea. One key point of failure is the lack of solid framework for decentralized blockchain governance. At the base of the stack, Arc[1] is an open, universal framework of smart-contracts for decentralized governance and collective value-management over the blockchain. Just as HTTP allows the creation and interoperability of websites and web applications, Arc allows the creation and interoperability of web-companies, collaboration apps and DAOs. The result is a new Web of Collaborations, in which collectives can self-organize around shared goals and values, not limited by pure economic growth. We believe this would be a critical step in the evolution of society towards a more cooperative and sustainable future.

[1] Derived from the Greek word Arche, meaning “method of government”.

Table of Content

1. Introduction

Since first appearing on the planet, mankind has been constantly inventing new ways to organize and increase the scale of cooperation towards more and more effective structures; from the nuclear family and tribes to corporations, states and the global economy. The most advanced organization thus far, the Internet, has opened the door for real-time information exchange at a worldwide scale, but it lacks the economic means for general-purpose coordination and global peer-production. The Blockchain made this possible by providing a reliable, open and programmable accounting system, consequentially leading to the invention of the Decentralized Autonomous Organization (DAO).

DAOs are open collectives coordinated by economic incentives and code, self-organized around shared goals. Powered by the network effect, DAOs provide a revenue model and incentive for the production of open, shareable resources (such as open-source code). With the creation of more open resources, the DAO is able to scale indefinitely, while keeping its agility and coherence, and can thus outcompete the existing corporate structures. DAOs have attracted top talents in the blockchain space, holding promise for more efficient and resilient organizations. Despite this, they have lacked critical elements to be successfully deployed so far, and in particular an adequate decentralized governance system.

DAOstack is an operating system for DAOs. With DAOstack, thousands of open-source creators can jointly produce decentralized applications (DApps), while distributing individual ownership in the product to contributors of value. Crowd curators can cooperatively own and manage multi-valued ranking systems to compete with Yelp, TripAdvisor, or YouTube. And autonomous networks can run their collective investment or insurance fund. DAOs can radically change the way we organize, from startups to corporations, to non-profits. DAOstack forms the foundation for this transition to the future of organization.

In the next chapter we will describe the future of DAOs: an acentric[2], anti-rival cooperation. In the third chapter we will present and explore the idea of blockchain, and in particular DAO governance. In the fourth present we will present the technology DAO stack that enables this new form of organization, and will give multiple examples for its usage. In the fifth and last chapter we will elaborate on the DAOstack economy and its engine of growth.

[2] The common term is '***decentralized',*** but we find '***acentric'*** (= has no center) more elegant and accurate.

2. DAO: the future of organization

Legacy organisations

Cooperation of agents increases their efficiency with respect to external competing market forces. This is the basic origin of the company[3] and the reason organizations want to grow. However, coordination of large number of agents is difficult and costly, and that is why organizations cannot grow indefinitely.

When growing, organizations needs more rigid structure in place, and thus face a growing challenge to: a) maintain agility with respect to rapidly changing conditions, and b) preserve alignment of interests, trust and engagement among their members. In short, the larger an organization is, the more internal friction it needs to cope with; the smaller it is, the more external competition is dominant. The size of a company is the sweet spot balancing between these two forces.

Once in awhile, the introduction of a new technology or paradigm shift enables the reduction of coordinational cost, pushing up the scale and efficiency of organizations to new levels. It triggers a transition in the landscape of work and business, and thereafter a social change too, as was exactly the case with the invention of crowdsourcing and the Internet itself.

The Internet allowed for an open, real-time and peer-to-peer information exchange on a global scale. As such, the Internet media has become more efficient, and scaled better, than traditional media outlets, and has quickly assimilated the latter. However, the Internet itself does not support open, peer-to-peer exchange of value and general-purpose coordination, and is thus limited in its potential to power global cooperation.

The Blockchain

Blockchain is the second internet revolution, doing to value and business what the Internet has done to information and media. It allows unprecedented levels of crowd coordination by eliminating altogether the issues of fault and trust, and consequently forms the technological basis for Decentralized Autonomous Organizations (DAOs). A DAO is a new form of scalable, self-organizing cooperation, that is operated by smart contracts on the blockchain. Many believe that DAOs hold the promise for the future

[3] This idea was formalized by Coase in his famous paper “The Nature of the Firm”.

of business and work, but despite a lot of traction in the blockchain community around this subject, a successful governance system and operational basis for DAOs is still missing.

Agencies

The building blocks of DAOs are ***smart companies***, or ***agencies***. An agency is an atomic governance unit that is managed and operated with smart contracts on the blockchain. It has its own token —related to benefits of the company's resources, its own reputation systems —related to credibility and influence in the company matters, and its own governance system —its "bylaws", encoded in smart contracts.

The governance protocol embedded in the smart contract can be anything one can come up with —and we will give a handful of examples in the next chapter, but the simplest example would be a yes/no majority vote on proposals —such as token distribution, when each vote would be weighted by the voter reputation. In a heuristic visualization it could look like this:

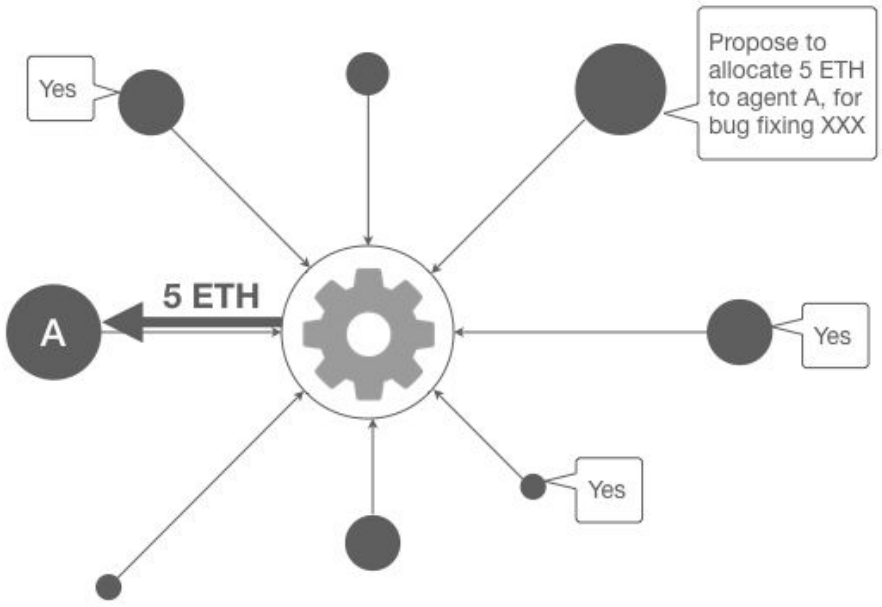


Image 2: A ***schematic illustration of a blockchain agency. The solid balls visualize agents in the company; their distance from the center reflects their influence power, or reputation (the closer they are the more influence they have); and their size reflects their native token possession (the larger the ball the more of the company's tokens they have). One agent is proposing to allocate 5 ETH to agent A for her valuable contribution of fixing bug XXX. The agents of the company vote, with their vote weighted by their reputation, and as soon as a majority of reputation holders agree with the proposal the contract automatically executes the suggested token allocation.***

DAOs

Agencies may or may not be autonomous —depending on their chosen governance system: they are not regulated by external forces, and rather operate faultlessly with smart contracts on the blockchain. They follow verifiable rules that cannot be broken and are changeable only in accordance with the rules themselves.

DAOs are in addition also centerless, meaning that there is no single point of control (or failure) in the organization. Instead of central management there is indirect coordination between agents, also known in biology as stigmergy. The DAO is a self-organizing entity, and at large, resembles an organism rather than an organization.

DAO Topology

There are two ways to think of DAOs with respect to agencies:

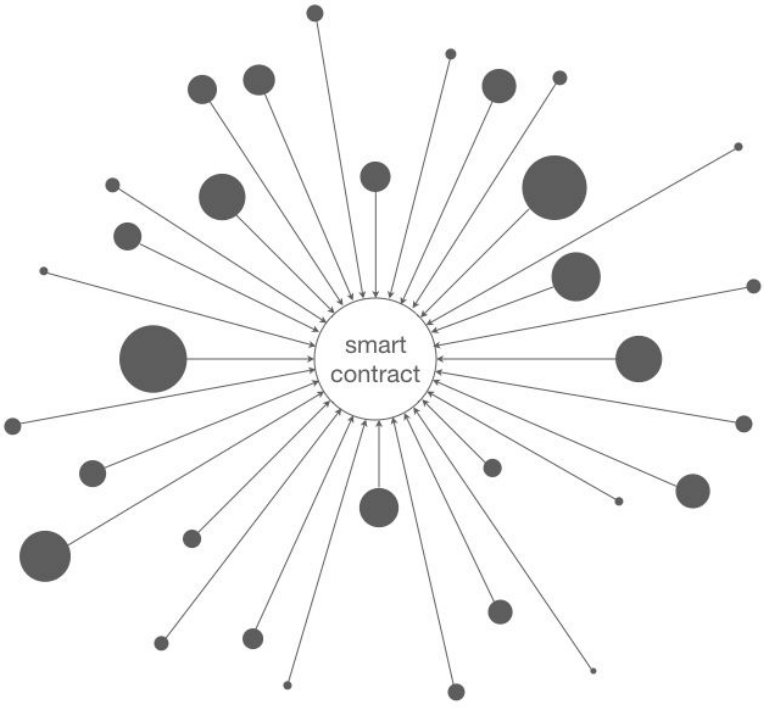


Image 3: ***The two DAO edge modes, an assembly and a fractal federal governance.***

The first way resembles an ***assembly***, having a large number of agents interacting in decision making within a single agency, so that power is distributed. The second way is a ***fractal federal governance***, where

an agency is managed by small number of agents, some of which are possibly themselves agencies, and so on and so forth. In reality, a DAO would probably be somewhere in between, but the bottom line is that power should be well distributed in a ***meritocratic*** way in order to effectively utilize its collective wisdom and benefit the organization as a whole.

Familiar DAOs

DAOs are nothing new. The human body is a DAO made of organs, sub-organs and sub-sub-organs, all the way to the atomic cells, which themselves have their internal structure. The functionality of the body is pretty decentralized and no cell instruct other cells what to do. Rather, each cell is autonomously operating according to inputs it receives from its environment. The sense of an organism —an autonomous and sential human being, is an emergent phenomena apparent only at the collective level.

An ant colony is a DAO too. It functions without any central management or control (no, the queen doesn't decide about the colony, it just lays eggs), and individual ant behaves in reaction to the conditions of its nearest environment. The sensible colony is an emergent phenomena at the collective level, derived from an indirect coordination of ants which need not even communicate directly with one another.

We are also very familiar with a human-colony DAO — the Internet. It is a acentric system which has gracefully scaled to more than 2bn users over fifty years of existence. Its dynamic self-governance helped it evolve and upgrade itself over time just like a living system. The Internet does not support internal value distribution though, so it lacks an inherent economic incentive model for engagement. Its functionality is thus limited to the distribution of information.

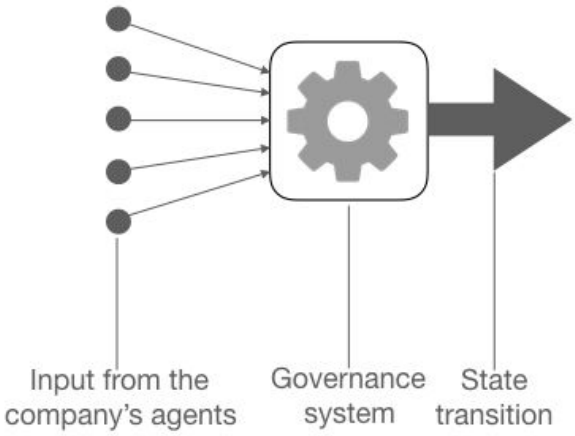
In fact, the blockchain itself is a DAO (or rather, DApp). And it is the first value-based DAO. It is a centerless, living organism operated by a wide crowd of engaged members (AKA ***miners***). With a new form of internal economic incentive model, it opens the door for growth and adoption levels never seen before. At the time of writing, the Bitcoin blockchain network has grown from zero to almost $100bn in value without any central management or coordination. The Ethereum blockchain has similarly grown to $30bn in three years! (The lucky investors of Ethereum's Crowd Sale have seen their investment going up about X1200 times in that time window.) But the functionality of these value-based DAOs is limited, and an additional element is needed to enable general-purpose DAOs that could organize around general purposes. The DAO stack is that missing element.

3. Blockchain Governance

In this chapter we describe what we mean by blockchain, and in particular DAO governance, and provide several examples.

The Agency

We denote by ***agency*** the basic unit of governance over the blockchain. A blockchain governance system is a state-transition function, that collects inputs from blockchain addresses and under certain rules spells out a global-state transition.



Instead of unwrapping this statement in a formal language, let us explain it via an example.

Simple fund

The simplest possible company is a company that manages an ETH fund, and internally only has its native FND token. We will call it Fundis. There are only three rules in the Fundis governance system:

1. The only collective decision that Fundis makes is sending ETH from its main wallet to a certain blockchain address (i.e. make an investment). This action is initiated by a proposal that is put up by an agent —say, agent X proposes to send 1000 ETH to address A. Then agents can vote yes or no on that proposal. At any time, votes are weighted by the number of FND that the voter has in her address:

Address	FND
0x00eFBC5D0A96e…	6
0x67248B65b6E0f7…	7.5
0x5F42801ac21677…	0.21
0x70f67cD70A50DA…	120
0xEA3C6A6b31dAD…	1.4
⋮	⋮

Once a majority of FND holders approve a proposal, it is automatically executed.

2. The second rule of Fundis is that upon its establishment, and for a period of one week, FND tokens are issued and purchased for buyers, one FND per ETH.

3. The third and final rule is that at any time, any FND holder can send his FND to the main Fundis wallet, and get back his share of the ETH existing in that wallet at that moment, pro rata. His FND is destroyed.

There are a few immediate issues with this governance system, such as:

1. It might be very hard to recruit a majority of FND holders to vote on one proposal. This issue is related to governance scalability, which we will discuss in length below.
2. It might be corrupted. Say, there is $100m worth of ETH in the Fundis wallet. And, say, that it costs $60m to buy half of all FND tokens in the market. Then, an agent with enough liquid capital can purchase just above half of the FNDs in the market, and immediately thereafter make a proposal to send all ETH to his own address. That's an easy $40m hit. This problem is related to resilience, which is the second topic we will cover below.
3. It is not clear (and probably not true) that those who hold more tokens would make the best investment decisions. It is not a deal-breaker, but might be leading to a pretty inefficient investment fund. This is in tension with meritocracy, which is another important criteria.

4. Another simple attack is made by making the proposal to distribute all of Fundis ETH into addresses that will vote yes on this proposal, pro rata. Perhaps in the beginning decent agents will refrain from approving this cunning proposal; but once the first approvals will come, there will be a growing pressure for approving it (and not losing all of your money), until an avalanche effect will take place and the cunning proposal will be approved. This to demonstrate that resilience is a tricky issue.
5. Finally, even after the Fundis creators have realized that its governance system if heavily flawed, they cannot do much about it since Fundis governance system is not upgradable, which is another criteria we would like to impose.

This simple governance system highlights almost all of the problems of blockchain governance, which we will discuss below. It also demonstrates the initial statement above: "A blockchain governance system is a state-transition function, that collects inputs from blockchain addresses and under certain rules spells out a global-state transition."

Indeed, there are four types of inputs that agents can inject into the company:

1. Submitting a proposal;
2. Voting yes or no on an existing proposal;
3. Sending ETH to Fundis and buying FND in the first week of its establishment;
4. And sending FND to Fundis and claiming back one's ETH share.

There are three possible actions (i.e. global-state transitions) Fundis can make, and three rules to trigger them correspondingly:

1. Issuing and sending FND to senders of ETH in the first week (1 FND for 1 ETH);
2. Sending ETH from Fundis wallet to senders of FND, pro rata (and burning the FND);
3. Sending ETH to an address following a successful proposal (as described above).

Reputation systems

Instead of weighing votes of agents with their native token stake, a company can have different balance sheets to denote the influence of agents inside votes. Generally, reputation scores are non-transferable assets —an agent cannot transfer its reputation to another agent. The simplest case is when a company has a single reputation system —meaning, the same influence score for all types of decisions; more generally, a company can maintain multiple reputation scores that are used for different cases. Reputation is used here interchangeably with influence power, and can form the basis for a meritocratic governance system, where those who are most appreciated —due to their past

contributions— have the most influence. One way to allocate reputation to agents would be via proposals. For example, one can propose to allocate 100 FNDr (Fundis reputation score units) to agent A for her recent valuable contribution C. Reputation can also be algorithmically tied with economic remuneration of the company to a contributor of value; as well as with one's votes and evaluations in relative to the collective. This topic is called reputation flow, and will be covered somewhere else.

Reputation system alone solves most of the problems of the previous token-based governance system, but it still remains unscalable.

Basic functionalities

An agency deployed on the Ethereum blockchain via the DAO stack can, in principle, do anything that can be done on the blockchain. In particular:

- **Token distribution**. Each agency or DAO can issue and distribute its own native tokens to contributors of value, as valued by the organization. The issuance of native tokens enables the organization the creation of its own separate economy. The utility or benefit of the tokens can be anything that the agency decides about, such as entitling access to the DAO's product (we will call these ***utility tokens***), or entitling a share of the agency's revenue (we will call those ***share tokens***).

- **Funds allocation**. An organization can earn, or collect via its own-token sale, external tokens such as ETH, STK or other DAO's tokens. It can keep them in reserve, and distribute to third parties in exchange of a particular effort or contribution. This is somewhat analogous to an agency using its funds to compensate contributors, employees or other service providers.

- **Reputation assignment**. Each agency can assign ***reputation scores*** to its members. Reputation is a representation of one's professional credibility, and thus influence, within the organization. As opposed to traditional blockchain-based tokens, reputation is not transferrable. It is awarded to or earned by specific members, according to their merits and contributions to the organization. Since reputation is tied with decision-making power in the organization, more reputation should be allocated to those who the organization believes make the best decisions. However, in order not to lock up decision-making power over time, the organization might decide that reputation dissipates over time.

- **Collective data curation**. An organization can manage its own collective databases of objects, and maintain their curation. It can be the curation of articles, website, organizations or anything else. The power of a shared database lies in its network effect; if everyone are looking at the same spot (because it's well-curated), then that spot is valuable (and monetizable). We will see below a few examples with the collective DAOstack registries, the ArcHives.

- **External activity**. An agency can act within another agency as a single entity. For example, an agency can submit proposals inside another agency (or DAO), and vote on others' proposals.[4]

- **Governance structure**. Each organization can configure and update its own governance system. By approving or removing certain governance modules, the agency defines how the it functions, what it can or cannot do, and what are the mechanisms for changing these governance schemes. Each organization can also adopt specific schemes imposing a number of limitations or global constraints on its operations.[5]

Governance structure

The governance of an agency can be divided into two types of actions, the ***do's*** and the ***don'ts***:

1. The ***do's*** are the logical and operational rules under which the above actions can be triggered in an agency. For example: if a majority of reputation holders in a DAO approve the issuance of new tokens, token issuance will be automatically triggered by the DAO's smart contract. We call these operational logics: ***schemes***.

2. The ***dont***'s are the restrictions and limitations that must absolutely be respected by an agency, and that cannot be violated even by an approved scheme. For example, if a DAO approves an upper cap of one million tokens, token issuance schemes will operate only as long as the total number of tokens issued is less than a million. We call these limitations: ***constraints***. Constraints can be absolute, or they can be designed to be upgradable under specific conditions.

[4] More generally, a DAOstack agency can act anywhere within the blockchain domain, thus preserving full interoperability with other systems. E.g. an agency in DAOstack can open a user in a company in Aragon :)

[5] The agency can also upgrade its technology stack itself, either to an upgraded version of DAOstack, or a whole new architecture.

We collectively denote schemes and constraints by ***elements***. Given the elements of an agency, its entire governance protocol (including the protocol to change the protocol) is unambiguously defined.

Schemes

Schemes are logical functions made of a series of instructions that take a particular set of inputs, and process them to generate a particular set of outputs. Schemes can be designed to do virtually anything, although most of them will trigger one of the above basic functionalities of an agency (i.e. token distribution, funds allocation, reputation assignment, etc.). Common schemes are based on ***proposals*** —the proposition to trigger some of the basic actions of an agency in a particular way, that will then be voted on and possibly approved, resulting in the automatic execution of the proposal.

A simple rewarding scheme

For example, a simple rewarding scheme in a DAO can be as follows:

- An agent submits a proposal to the DAO to reward 150 G-tokens and 200 G-reputation bits (***G-reps***) to another agent with address W, for contribution made to the DAO.
- Anyone can vote ***yes*** or ***no*** on this proposal, to be weighted by their DAO reputation score.
- Once a majority of the DAO reputation holders vote ***yes,*** the DAO's operating system executes the proposal and allocates 150 G-tokens and 200 G-reps to address W. Note that W can also be the address of a smart contract, and in particular the address of yet another agency.

Fundis schemes

Similarly, when looking back at the Fundis example above, it had three rules, which could be written as three distinct schemes. Thus, the entire governance system of Fundis is describable via those three elements. We will use this modularized formalism in the next chapter, when describing the DAO stack and in particular the ***Arc*** governance framework.

A more complicated example

For the sake of illustration, a fairly more complicated reward scheme could be described as follows:

- An agent submits a proposition requesting to be rewarded for a particular contribution.
- The scheme triggers an alternative voting system, enabling anyone holding reputation in the DAO to propose a specific number of tokens to award the agent with.

- Votes are weighted by the reputation score of each voter, and the total amount of tokens to be awarded equals the weighted ***median*** of all voters.[6]
- As soon as 20% or more of reputation holders have expressed a vote, and not more than 2% of reputation was engaged in the vote during the last day, the DAO will allocate the amount of tokens and a proportionate amount of reputation to the agent, following the median at that moment.

More generally, the spectrum of possible scheme's design is nearly endless. Schemes can activate different actions (including the basic functionalities mentioned above), incorporate a variety of different logics, and rely on a variety of different voting systems which might associate voting power with reputation, tokens, or a combination of both. Another common typology of schemes is the one allowing DAOs to offer their native tokens for sale under some conditions (the so-called token sales).

Global Constraints

Global constraints are specific conditions that can be attached to a particular agency or DAO and that will limit its functionalities. As a general rule, for any operation to go through within a particular agency, it must comply with the list of global constraints configured by the organization.
Regardless of the list of schemes that have been approved by the agency, none of these schemes will be able to trigger a particular functionality within the organization, if doing so would not conform with even a single one of the global constraints.

For instance, common global constraints that might be adopted by a large variety of DAOs include:

- The upper cap imposed on the total amount of tokens that can be issued by the DAO.

- The rate of inflation that will determine the dynamic cap of tokens that can be issued for any given time period (e.g. 2% per month).

- The burn rate of usage of a DAO's (say, ETH) fund .

- The maximum amount of reputation that can be issued within a particular time period.

[6] This is the largest number that 50% or more of voting reputation has assigned a larger or equal number of it.

- The identity of one or more agents that can request the issuance of more tokens or reputation (i.e.. whitelisting).

- The registries from which the DAO can approve new schemes.

- The conditions that all propositions must conform to in order to be deemed as valid for submission to any approved scheme.

An organization can define certain global constraints to be permanent and other to be modifiable. In the latter case, the organization will also define under what conditions these constraints can be updated (e.g. by a 75% decision of token holders, or only via this or that specific scheme).

Decentralized Governance

We understand now what is a blockchain (or more generally, algorithmic) governance system. What do we mean then by a decentralized governance system? Intuitively we think of many agents participating in every decision, but that is not necessarily accurate, nor well defined.

To understand it better let us look at the one decentralized governance system we already know well —the blockchain itself. The blockchain is a decentralized system that at each instance decides about the order of events in a computing network. Events can be triggered by any node on the network, but a priori, since the ***gossip*** about events is whispered from one node to another, and there is no central body to overview the entire process, there is no objective way to determine which events occurred before and which after. The blockchain is this magical mechanism for a network to reach a consensus about the order of events. This is a very particular set of decisions, but it is a decision-making system, and it is decentralized.

One immediate observation is that the nature of decentralization of the blockchain protocol is not originated from the fact that there are little or many participating agents, but rather that there is a fixed set of rules under of which anyone can participate democratically. So the essence of decentralization is that it operates under an ***open protocol***.

Scalability

The scalability of decentralized governance systems is in inherent tension with resilience. By decentralized we means that we want to allow any participant to play the game (i.e. use an ***open***

protocol). By resilience we mean that we need enough participants to review every decision. But this is clearly in tension with the scarce resource of participants' attention, whether it is computing power —in the case of blockchain, or human attention —in case of DAO governance.

We argue that there are generally three ways to resolve (to some degree) the tension of scalability and resilience in a decentralized governance system, which are: ***compositionality***, ***attention monetization,*** and ***relative majority***. As mentioned above, the blockchain itself is a form of decentralized governance system, in which these mechanisms are somewhat analogous to: ***sharding***, ***gas***, and ***off-chain*** computations.

Compositionality

Consider the following two cases. In the first, nine agents of a company with equal votes. In the second, three agents of a company with equal votes, each of which is by itself a company, composed of three agents with equal votes. A sub-company casts its votes in the mother company as soon as it develops its decision via an internal majority.

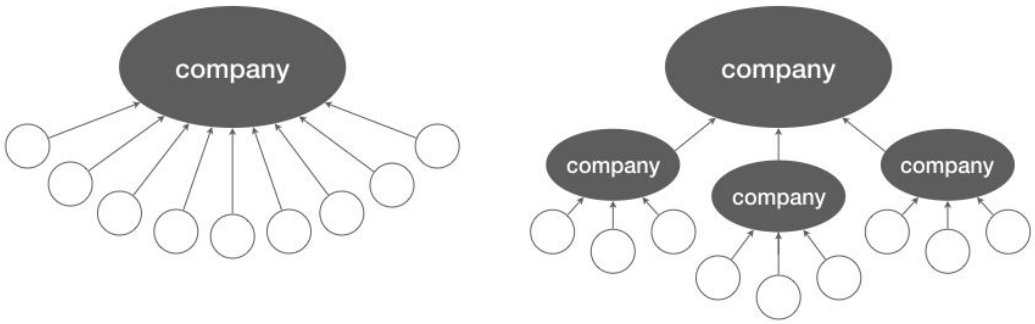


Fig X: The assembly vs. federated governance.

It is easy to see that in the first case, it requires the attention and consensus of five human agents in order to cast a decision in the mother company; whereas it is sufficient to have the consensus of four human agents in the second scenario to form a decision in the mother company.

This very simple example illustrates the fact that ***compositionality***, or ***fractalization***, of governance systems makes them more scalable. One can argue that by this we lost resilience (since less human agents can now take over the system); but we can also observe that not any configuration of four human agents in the second scenario can form a decision in the mother company. If we take this into

account it can be shown that under the assumption of a fixed probability of faulting an agent, the resilience in the two examples actually remains the same.

Attention monetization

Forming a decision requires the attention of voters. The more decisions are there to evaluate, the more attention is needed. However, attention is a scarce resource. In one way or another, to form a resilient decentralized decision-making system, attention should be monetized to reflect its inherent scarcity. In the Bitcoin blockchain it is formed in terms of ***transaction fees***, and in the Ethereum blockchain as ***gas***. For DAO governance, attention needs to be monetized with the token that reflects the community of agents whose attention is being called for. For internal DAO decisions that would likely be the DAO token, whereas for inter-DAO activities we would use Stacks, the DAOstack ecosystem token. Note that monetization of attention does not buy decisions, but only buys the attention over proposals; we will elaborate further on this point below.

Relative majority

Naively, a straight-forward way to form scalable decision-making process is by demanding only a relative majority to approve a decision. By relative majority we mean that, the majority of approvals is only with respect to those who cast their vote (and not all potential voters in the system).

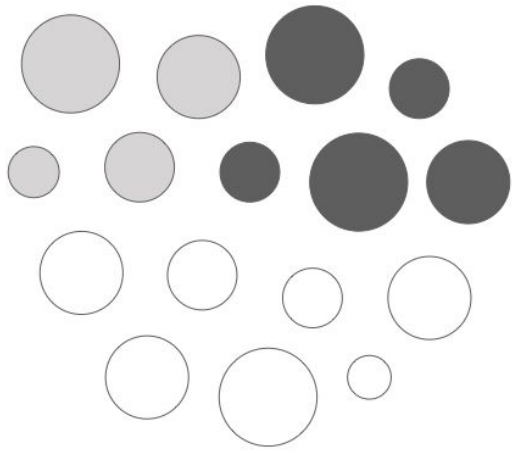


Figure X: The size of the balls reflect the voting power of agents. The empty balls are those who have not cast their vote on a specific proposal; the darker balls voted yes, and the lighter balls voted no. Clearly, there is no absolute majority of approvals, but there is a relative majority of approvals out of all voting agents.

Note that relative majority votes requires a finite time window for the proposal to be considered.

To solve this problem, previous trials of governance systems introduced a ***quorum*** —the minimal amount of voting power put into a decision above of which the vote is considered legitimate. The problem with a quorum is that it is impossible to fix the right amount of it, and will either damage resilience —if fixed too low, or scalability —if fixed too high, and possibly both. If at all, quorum needs to be dynamic.

In the proposal below, we provide a resilient governance system, without a quorum, that is based on relative majority and attention monetization.

Minimal Viable Protocol

In this section we propose the simplest decision-making protocol that is potentially decentralized, resilient and, to some degree, scalable. We describe the protocol in the following seven steps:

1. **Proposals**. Decisions are initiated by proposals submitted by agents that are voted on with a yes or no. This is not the most general decision-making process, and we'll explore more possibilities below, but for now we stick with the simplest example.

2. **Based on reputation**. Agents' votes are weighted with their reputation. We will stick for now with a single reputation system per company. In a DAO there will be many sub-companies, and sub-sub-companies, each of which will be focused on a specific branch of production or decision making. In this fractal framework, the idea of a single reputation system per company is not inconceivable.

3. **Finite time**. Once a proposal is ***opened*** —as we will define below— it has a finite time to be closed (say, 2 weeks). Which means, by the end of this time interval a decision (yes or no) is taken according to the relative majority of votes (out of all voters).

4. **Quiet ending**. To avoid finalization attacks, the effective decision (majority of yes or no) cannot be changed in the ending interval (of, say, 1 day). Meaning, if on the last day of opening the majority changed from yes to no (or vice versa), the opening interval is extended by another day. The vote is closed only once there is no change of decision during the last day of voting.

5. **Opening stack**. At each point in time there can be only a finite number of open proposals (say, 10). All other proposals stand in a queue, and are ordered by a ranking system. Every time a proposal in the opening stack is finalized, the proposal with the highest rank in the queue is opened and enters the opening stack.

6. **Boosting**. Anyone can propagate a proposal up in the queue by ***boosting*** it. Boosting is done by putting tokens on stake.[7] If the proposal is successful the booster gets back her tokens, and otherwise they are destroyed.

7. **Ranking system**. The ranking system of proposals in the queue can be pretty general, but a simple sensible choice would be: $R_+^2 \cdot B$, where R_+ is the amount of reputation already voting yes on this proposal, and B is the amount of tokens boosted that proposal.

4. The DAO stack

DAOstack provides the foundational tools for the creation, operation and governance of DAOs, internally and externally within a broader ecosystem. In a nutshell, it can be regarded as an analogue of ***Wordpress*** for DAOs —it does for blockchains what ***WordPress*** has done for the web. This vision is made possible with the following stack of components in place:

[7] Which tokens will be made clearer in the next two chapters.

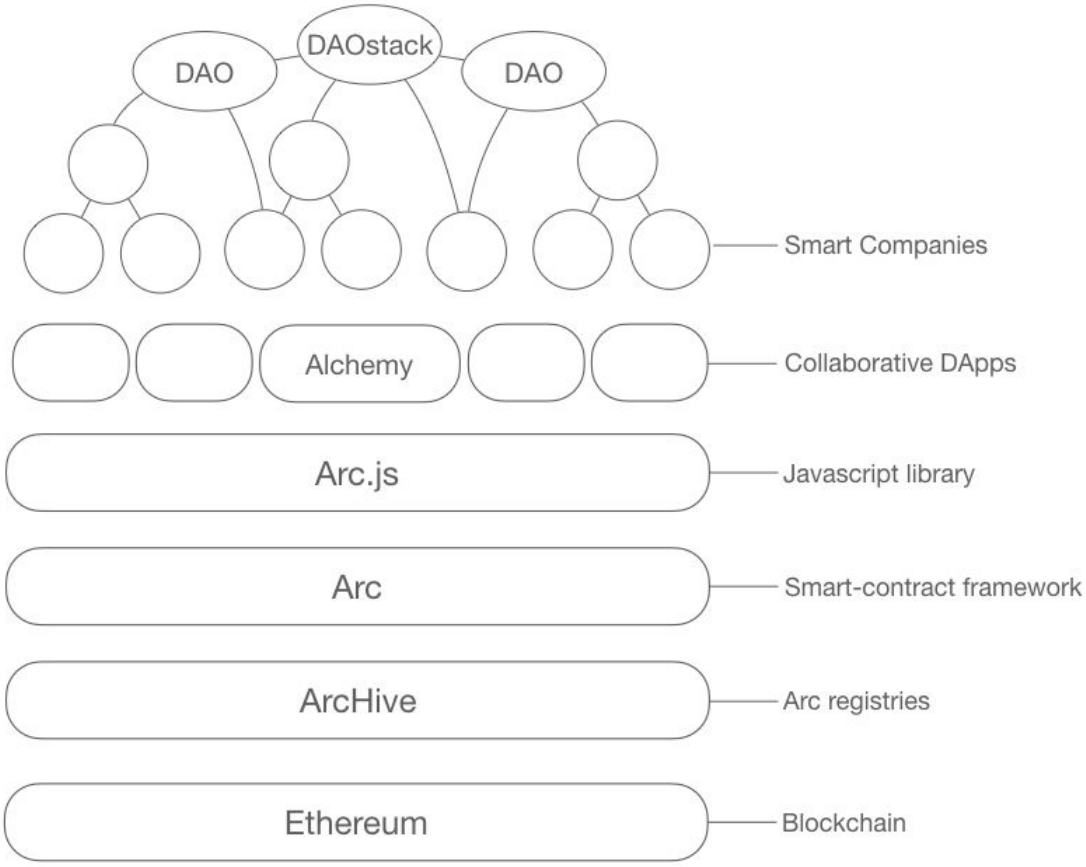


Image X: A rough sketch of the DAO stack.

The DAOstack ecosystem is made of a multitude of distinct but interoperable DAOs, interacting with one another in order to maximize the potential benefit of open and distributed collaboration. At the technical level, all DAOs are made of a series of smart contracts, deployed through ***Arc***: a Solidity framework of governance modules allowing to create, configure, deploy and operate DAOs onto the Ethereum blockchain,[8] possibly relying on IPFS as an overlay network for data storage and retrieval.

People can interact with these DAOs either directly, through the execution of blockchain transactions, or indirectly, by relying on a particular front-end to the underlying blockchain ecosystem. ***Alchemy*** is the front-end developed internally by DAOstack, enabling anyone, without technical knowledge, to create a new agency or DAO and start collaborating with others in the DAOstack ecosystem. It relies on ***Arc.js***, a JavaScript library that operates the Arc Solidity framework via Web3.js. It is designed to make

[8] While Arc is currently based on the Ethereum blockchain, the framework will eventually become blockchain agnostic once other technology platforms mature and interoperability is achieved.

the interaction with Arc and the blockchain accessible to front-end JavaScript developers; easily creating collaborative applications on top of Arc without the need to interact directly with the Solidity code, or the Ethereum blockchain. The ***ArcHive*** is a public set of registries, curated by the DAOstack community and serving its global ecosystem; it is where the ecosystem, and network effect, builds up.

Arc

Arc is a general governance framework for an interacting internet of blockchain agencies, the basic operating system for DAOs. It is an open-source, modular and general-purpose framework by design, and it comes with an open library of template governance modules, or ***element***, that will evolve by the needs of its users. It also allows an easy upgrade and modification of a governance system to better fit the organization's specific needs over time.

Arc does not favor any specific governance structure over another, and quite on the contrary, it makes it easy for third parties to create their own ***elements*** per their agency needs. By combining available ***elements***, each agency can implement its own governance structure that specifies the rules for the issuance, management and assignment of scarce resources —including transferable assets (such as blockchain-based tokens and other smart assets) and non-transferable assets (such as reputation and influence scores).

Architecture

Arc elegantly implements in smart contracts the basic decomposition of governance systems discussed in the previous chapter, with: ***actions, schemes*** and ***constraints*** that every agency can be built of. Below is a visual representation of the logic and smart contract architecture of the Arc framework:

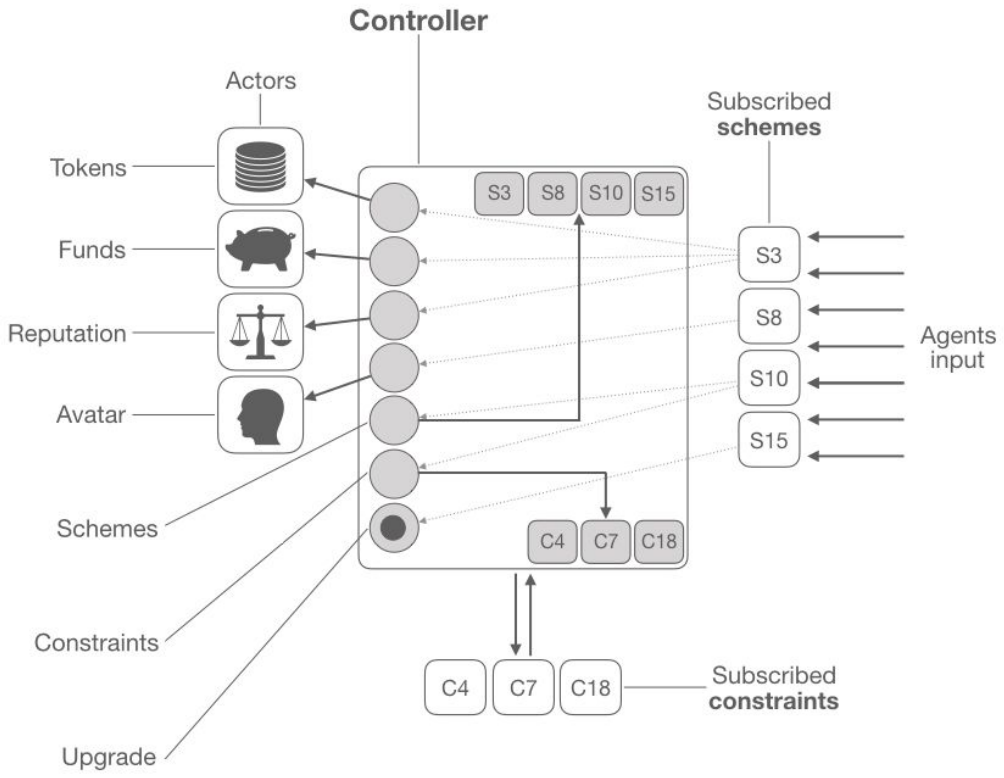


Image 6: A sketch of the Arc contract bundle per agency.

Upon the deployment of a naked agency, it has only one scheme that can only register more elements. We call this first scheme —which also unfolds the ***Controller*** and the ***actor contracts***— the ***Genesis Scheme*** (denoted by S10 above).

The Controller

The controller contract is the main engine of the agency. It is "owned" by and gets commands from the ***subscribed schemes*** alone, which operate its functions. Via its functions its sends commands to the ***actors*** contracts: the ***token*** and ***reputation*** printers, the funds ***wallet***, and the ***avatar***.

Subscribed elements

Subscribed schemes and constraints are elements that have been registered by a previous scheme which is allowed to register schemes. Every agency begins its life with one such scheme, the ***Genesis Scheme***, but can add new ones that can register or unregister certain elements under certain conditions.

Agent's input

The only interface for agents to interact with the agency is via its subscribed schemes. Each scheme comes with its specific "knobs" (functions) which can be operated by external agents (blockchain addresses) via transactions that call and operate these functions.

Token printer

The token printer issues and allocates the native tokens of the agency to agents. It is "owned" by and gets commands from the controller alone.

Reputation printer

The reputation printer assigns the agency reputation to agents. It is "owned" by and gets commands from the controller alone.

Wallet

The ***wallet*** will be implemented in a next version (at the moment the ***avatar*** is also the wallet). It holds external funds owned by the agency and distributes them to agents. It is "owned" by and gets commands from the controller alone

Avatar

The ***avatar*** is the "face" of the agency (and its address), which acts externally and is capable of doing —via the help of particular schemes— anything that can be done on the blockchain. In particular, it can participate as an agent in other agencies —e.g. submit and vote on proposals— on behalf of its agency.

Protocol upgrade

The controller can —when receiving commands from the ***subscribed schemes***— register new ***schemes*** and ***constraints***, or unregister old ones. Since the governance protocol of an agency is fully specified via its ***elements***, every agency can easily upgrade and modify its governance system (if this ability is rooted in its current protocol). The conditions to perform a protocol upgrade are defined in subscribed schemes that can operate the elements-(un)registration. The agency can have different conditions to register or

unregister different elements. (E.g. it might require a 60% majority of reputation holders to modify the schemes and most of the constraints, but a 75% of token holders to increase the token supply.)

Technical upgrade

The controller has one special function that transfers its ownership over all of the ***actor*** contracts to a new address. Whatever would be this new address, it will have a complete control over the assets (transferrable and non-transferrable, such as reputation in other agencies) of the agency. It can be an upgrade to an improved architecture of Arc, or to a whole new architecture, if desired so by the agency.
Example
We present below an illustration of the interaction between above various elements:

- An agent wants to activate a certain "reward scheme" of a particular DAO.
- The agent will submit a transaction to the smart contract of that scheme (possibly via one of the DApps integrated with the DAO stack), along with the relevant input required for that scheme (e.g. agency, number and type of tokens, recipient address, etc.).
- Other agents may approve this request with their vote.
- Once enough votes approve the request —as defined by that specific reward scheme— the reward scheme will command the controller to execute the proposal (with the specific parameters), and the controller will command the relevant acting contract to execute the reward allocation.
- All functions governed by the controller are subject to the global constraints registered by the DAO. Global constraints act as both an ex-ante and ex-post modifier. Before executing any function, the controller is required to run all global constraints stored in the array to ensure that they all return false at this particular point in time. Given that the state of the DAO might change after a function has been executed, once execution is completed, the controller is required to go—once again—through all global constraints, and if any of them returns false, the controller will revert everything to the previous state.

Universal elements

Elements —schemes and constraints— are universal by design. Meaning, several DAOs can rely on the same universal element contracts rather than deploying their own contract each time. Such a design contributes to the scalability, functionality and security of the framework.

Design Principles

Arc was designed with the following design principles in mind:

- ***Generality***. Arc is a general framework that supports an indefinite number of governance elements. Over time, Arc's library of schemes and constraints will grow, with the addition of many new templates and modules, possibly developed by third parties and the open-source development community. When combined, it will allow for people to experiment with an increasingly large number of governance protocols, and consequently for the successful governance protocols to float up by natural evolution.

- ***Modularity***: The Arc framework is highly modular by design. Every DAO's governance structure is made of small building blocks (the governance modules, or ***elements***) that can easily be added, combined, edited or removed. This modularity becomes a point of efficiency in that these building blocks do not need to be redeployed onto the blockchain, rather only referred to, saving both storage and operation cost. Moreover, it makes development of sophisticated governance protocols easier and easier the more building blocks are produced. Last but not least, it makes contract security management possible (whereas writing entire protocols each time from scratch is simply inconceivable security wise).

- ***Simplicity***: Arc has been designed with a strong emphasis on simplicity — in terms of both technical design and usability. The modularity of the framework makes it possible to focus on individual building blocks, whose complexity can be kept to the minimum. As the building blocks are modular and re-used, each will receive a lot of individual scrutiny, affording a higher level of vetting, and the corresponding greater level of security. To facilitate ease of use, in Arc, bundles of smart contracts can be deployed with a single transaction.

- ***Upgradability***: The governance structure of each DAO can be easily upgraded to use new schemes or different parameters of existing schemes. More granularly, each DAO created through the Arc framework comes with a particular set of rules, which by default include rules to change the rules. In addition, the DAO also comes with the ability to upgrade its technical architecture and to evolve to a better and better architecture over time.

- ***Interoperability***: The entire system is designed to facilitate interactions and promote interoperability between different agencies and DAOs. In practice, it means that agencies can interface with other agencies in a seamless manner, exchanging tokens with one another, acting as individual contributors or agents within another agency, and possibly even acquiring influence over other agency. The framework makes it possible for a mesh network of

interconnected agencies and DAOs to emerge, spontaneously creating the DAO ecosystem.

- ***Openness***: Open frameworks such as Wordpress, Google add-ons and Android invite independent developers to create their own apps, templates and integrations — thereby instantiating and supporting a vibrant development community and which in turn benefits the framework with a wide set of applications. Similarly, we expect DAOstack to attract developers from the Ethereum community to develop their own governance modules or frontend DApps—thus creating a thriving plethora of templates, modules, and applications for this emergent ecosystem. The DAOstack code is fully open source, backend and frontend.

Arc.JS

Arc is a pretty extended Solidity framework, the integration with which requires a sufficient acquaintance with the Solidity smart-contract programming language and the Ethereum blockchain. To make the integration of third-party apps easy for enthusiastic front-end (JavaScript) developers, we have designed Arc.js as an easy JavaScript gate.

Arc.js is a JavaScript library built on top of Web3.js (Ethereum's JavaScript API), with which any functionality of the Arc framework can be called directly from within the JavaScript environment, without knowledge of the Solidity programming language.

We expect this additional layer to make the Arc framework (and the rest of the DAO stack) extensively more accessible for the open-source development community, and third parties, and thus greatly promote the early adoption of the DAO stack and the growth of its ecosystem.

The ArcHives

Arc is the operating system for the DAOstack ecosystem, and the ArcHives are its shared, curated records. Shared, curated databases can be very powerful, providing a central locus for data integrity and data quality; but more so, they are the locus for the ecosystem network effect, which is also the source for an open-ecosystem monetization (i.e. business model). We will describe below the basic ArcHives of DAOstack: the Agency Index, the Elements Index, and the Hub Registry, and will explain the monetization model behind them. Beyond these, each agency or DAO can have their own independent registries, and in particular DAOstack can also have many other registries in the future.

The Archery

Arc's open framework makes it possible for anyone to deploy new governance ***elements*** —schemes and constraints. For the purpose of security, DAOs will likely decide to limit themselves to schemes that have gone through exhaustive and professional audit, and more so have been heavily battle tested. For that purpose, DAOstack has implemented its own app store for elements, the Archery, which records all governance modules that have been approved by the DAOstack expert community.

The Archery provides additional utility to the Stacks and revenue model for DAOstack, as it costs Stacks to deploy a governance module onto the registry. Moreover, in order to encourage independent developers to develop further elements, DAOstack will allow element developers to implement a universal business, collecting a particular amount of Stacks (voted on by the DAOstack community) upon every subscription of new agency to a module. All of the elements developed by the DAOstack founding team will remain free of charge for the community.

Once there is a network effect of users using the Archery for the elements they subscribe to, the value proposition of this registry will appreciate and accordingly its monetization capability. Basically, it will be more profitable for developers to publish their elements on this registry, and thus pay the registration fee. The registration fee is also useful to filter spam publications, as it will only be profitable if the element will be approved, and widely used by the DAOstack ecosystem.

Agency Index

DAOstack also implements an organization registry that records a list and metadata of all agencies and DAOs deployed through the Arc framework. This database is critical for the interoperability of all collaborative DApps and organizations in the DAOstack ecosystem. This registry will also be curated by the DAOstack community, which also implements a search engine to facilitate discovery. Just like in the case of the the Archery, it costs Stacks to register or promote an organization on the Agency Index.

The Hive

The Hive is where innovators, professionals and stakeholders can meet each other within multiple organizations. It is an open billboard that everyone can use to post requests or offers, which are registered and can be promoted by spending Stacks. The Hive registry is curated by the DAOstack community, which, again, also implements a search engine to facilitate discovery.

Alchemy

More Protocol examples