Evaluating the software frameworks for developing Decentralized Autonomous Organizations

María-Cruz Valiente[1], Juan Pavón[1], Samer Hassan[1,2]

[1]Knowledge Technology Institute, Universidad Complutense de Madrid, Spain
mcvaliente@ucm.es, jpavon@fdi.ucm.es

[2]Berkman Klein Center for Internet and Society, Harvard University, USA
shassan@cyber.harvard.edu

Abstract. First Bitcoin in 2008, and later Ethereum in 2014, held a powerful promise: online decentralized governance, without servers or central controllers, not just for finance applications like crypto-currencies but for any organization. The so called Decentralized Autonomous Organizations (DAOs) were expected to fulfill such a promise, enabling people to organize online relying on blockchain-based systems and smart contracts automatizing part of their governance. In 2016, three DAO software frameworks —Aragon, Colony and DAOstack— emerged aiming to facilitate development and experimentation in this field. To which extent do they facilitate DAO development today? This paper performs an analytical comparison of these three frameworks, focusing on their current functionalities for building DAOs. We find Aragon to be the most complete in several aspects. In order to provide more details on the challenges on building DAOs with current frameworks, we present a case study using the Aragon framework. Through this case study, we have piloted DAO development using this framework, and thus we may highlight the benefits, limitations and problems that developers face when adopting it. Our findings show that, even if Aragon does provide superior capabilities to other frameworks, it is still highly challenging to build a DAO with the current tools. Today, problems include issues on software engineering, instability, localization, documentation, lack of formalization and standards, and interoperability. Complementarily, this paper aims to provide some guidance to those developers aiming to face the challenges in developing a DAO, and to those aiming to fix the major weak points that make DAOs the organizations of a still distant future.

Keywords: Blockchain, DAO, decentralized autonomous organization, Ethereum, smart contract, software framework.

1. Introduction

Blockchain was initially introduced as the technology enabling Bitcoin [1], as a new form of digital currency and, in short, it can be seen as a distributed database that stores transactions grouped into blocks, which results in a decentralized public digital ledger [2]. The blocks represent the history of the blockchain and the transactions change its

state. In this context, a transaction could be seen as an atomic write operation triggered by a user that may alter one or several records of the corresponding database. For example, a currency transfer between two accounts will reduce the sender's balance and will increase the recipient's balance.

Another definition is found in Antonopoulos & Wood [3], where this paradigm is defined as "a fully-distributed, peer-to-peer software network which makes use of cryptography to securely host applications, store data, and easily transfer digital instruments of value that represent real-world money."

Aiming to move beyond cryptocurrency applications, in 2014 Vitalik Buterin co-founded Ethereum, a general-purpose blockchain-based distributed computing platform [4]. In a white paper [5], Buterin introduces the *Decentralized Autonomous Organization* (DAO) term as a way to explore those new organizations' governance rules that could be automated and transparently embedded in a blockchain.

A relevant feature of DAOs is that they operate without central control/management. That is, the participants of a DAO typically hold some voting power and can submit proposals that will be approved or rejected through several decision making mechanisms [2, 6]. Besides, as a decentralized organization, a DAO can "provide services (or resources) to third-parties or even hire people to perform specific tasks. Hence, individuals can transact with a DAO in order to access its service, or get paid for their contributions." [7].

During the last five years, Ethereum and DAOs have gained increasing attention in the industry. They have been discussed in Economics, Law, Organization Theory or Computer Science as a means to support non-hierarchical organizations that are concerned with ensuring sharing, security, transparency, and auditability, enabling global business models without a central authority or middle-man controlling them.

Nowadays, MolochDAO [8], a DAO created to fund Ethereum 2.0 grants, has become the facto reference for people concerned about DAOs. However, the complexity to build a DAO from scratch is significant and, even for experienced software developers, it is highly complicated to grasp all the issues related to it. In fact, there is a lack of both accepted standards and of widespread use cases of DAOs deployed on the Ethereum blockchain (at the time of writing), specifically if we compare their added value, for example, in terms of efficiency or new services, with traditional, centralized organizations.

As a response to this issue, some open source software frameworks have emerged to facilitate the implementation of DAOs, of which the major ones are Aragon [9], Colony [10] and DAOstack [11]. However, even if the three frameworks were established around 2016, it is still difficult to find comprehensive material to understand the architecture and many aspects of these frameworks needed to build a DAO.

Thus, in this paper the research questions are: What kind of problems do developers find when implementing a DAO using a DAO framework? What advantages are provided by the use of a DAO software framework in order to adopt a decentralized approach? To the best of our knowledge there are currently no works that attempt to evaluate existing software frameworks as toolkits for building DAOs. This paper analyzes these frameworks and illustrates one of them, Aragon, which is currently significantly more popular than the other two. This is done with the case study of a sample DAO that provides support to researchers participating in a common project to manage the different tasks that they have to carry out.

The paper proceeds as follows. Section 2 covers the technical issues to take into account when building DAOs. Section 3 reviews and compares the three main frameworks mentioned earlier concerning the DAO development process. Section 4 presents the methodology followed to implement a DAO with the Aragon framework using a practical example. Finally, we summarize the experience and provide an overview of the results in Section 5, together with some recommendations

2. The development of a DAO

2.1. Smart contracts

In 1994 Nick Szabo introduced the term 'smart contract' as "a computerized transaction protocol that executes the terms of a contract" [12]. This definition was adopted in the context of Ethereum differing from current web applications, in the sense that they can run autonomously (i.e., run without requiring a trusted third party).

In Ethereum, smart contracts can be defined as "computer programs that run deterministically in the context of an Ethereum Virtual Machine (EVM) as part of the Ethereum network protocol" [3]. Regardless of it being referred to as a smart "contract", it has no legal meaning in this context, although there is some debate concerning their potential recognition by Law [13]. In fact, as described in Filippi & Hassan [14], smart contracts aim to emulate the logic of contractual clauses.

2.2. Contract accounts vs Externally Owned Accounts (EOAs)

When a smart contract is deployed to the Ethereum network, the code is stored in the blockchain and generates an address that acts as the identifier for that smart contract. This type of address associated with smart contracts is known in Ethereum as *contract account*.

On the other hand, in order to access Ethereum blockchain and run specific smart contracts, it is required another type of account known as *Externally Owned Account* (EOA). An EOA is an account managed by an individual or organization that has a private key which is used to control funds and contracts. An EOA is obtained through the use of an Ethereum wallet. The wallet term comes from the original use in Bitcoin, as the individual's software that stores the cryptocurrency, akin to the physical wallet that contains physical money. Therefore, in Ethereum, this wallet is a software application that enables the management of EOAs providing a key vault, secure login, and token wallet. The most popular browser extension for creating a web-based wallet is MetaMask [15].

Similarly to contract accounts, EOAs are represented by addresses, they can send money and store data in the Ethereum network, but only EOAs can initiate transactions. Besides, each interaction (i.e., contracts deployment and transactions) has associated a cost, named as 'gas', and EOAs must pay ether (ETH), Ethereum's native cryptocurrency, in order to perform them. In this respect, Ethereum provides two types of networks: (i) *mainnet* where real ether is used in the transactions (i.e., production environment); and (ii) *testnet* which is the network used by developers to test the blockchain and interact with their applications (i.e., test environment).

2.3. Governance

When we refer to ‘governance’, it is more than just the decision-making process. For example, in Economics, governance “is the use of institutions, structures of authority and collaboration to allocate resources and coordinate the effort and activity in society or in the economy” [16]. Thus, governance rules force organizations to consider and formalize their understanding of their current decision making processes.

In Ethereum, through the use of smart contracts, DAOs can implement their governance model, encoding at least part of their rules implemented in one or several smart contracts and stored in the blockchain. Depending on the DAO, voting can occur for taking decisions, for example, distributing funds or implementing new rules.

2.4. Decentralized applications (Dapps)

In Ethereum, developers can program their web applications, known as *Decentralized applications* (Dapps), without knowledge about the underlying mechanisms of peer-to-peer networks, blockchain, consensus rules, etc.

Dapps are composed of at least: (i) a smart contract that acts as a software agent running on the blockchain performing predefined or pre-approved tasks without any human involvement, and which is under the control of a set of business rules [2]; and (ii) a web user interface.

From the Ethereum point of view, a DAO could be defined as a Dapp that may be composed of other Dapps, which are all running on the Ethereum platform, and whose business logic is encoded in terms of one or more smart contracts. In this way, the smart contracts enforce at least part of the governance rules (i.e., decision making rules) of a specific DAO.

3. Frameworks for building DAOs

Each of the three frameworks described in this section has a different definition of DAO, and thus a distinct approach and focus. Their software and tools reflect that, stressing different features in each of them. We aim to describe each exposing their own approaches, and what they find relevant, according to their documentation. For a comparison of features, see the subsection 3.4.

3.1. Aragon

Aragon is a software framework oriented to the development of DAOs built on the Ethereum platform and aimed to create highly configurable governance structures [17].

From the point of view of Aragon, a DAO is seen as an organization made up of smart contracts (i.e., software programs as defined above) named ‘Aragon apps’, where each smart contract is associated with a web user interface (i.e., a Dapp).

Aragon provides several pre-configured templates for DAO creation (i.e., pre-configured smart contracts for different types of organizations).

3.2. Colony

Colony is a DAO framework based on a reputation system (i.e., decision power is weighted by the user reputation). It aims to help organizations create their own DAOs, named 'colonies', providing financial management, ownership, structure and authority.

The Colony network is composed of a suite of smart contracts which are deployed on the Ethereum blockchain. However, although it is planned to be included in the near future, at this moment, organizations cannot customize colonies with smart contract modules in order to implement their specific governance model.

According to Colony's whitepaper [18], the structure of a colony is based on *domains* and the *permissions* that accounts may have in each domain.

3.3. DAOstack

DAOstack supports the process of DAO development by providing a library of governance protocols and user interfaces that facilitates their creation and management.

From the point of view of DAOstack, a DAO is seen as a network of stakeholders who make non-hierarchical decisions about shared resources [19]. In DAOstack, decisions are initiated by *proposals*. The framework of DAOstack is composed of a set of several modules or layers [11].

3.4. Aragon, Colony and DAOstack comparison

Implementing DAOs with rich functionality requires the next mechanisms: (i) Financial management (i.e., funding); (i) A voting system; (ii) Tokens for membership management and voting power; (iii) Ability to create new governance models relying on smart contracts; (iv) Templates of organization models; and (v) Permissions, which are of major importance when different roles are accessing to a DAO. The comparison among the three frameworks (Aragon, Colony and DAOstack) in relation to these main mechanisms associated with the process of building DAOs are summarized in Table 1.

As can be seen in Table 1, among these three frameworks, only Aragon offer prototypes of DAOs (organization templates), which can be configured, and provide mechanisms to add smart contracts that enable the definition of new governance models. On the other hand, DAOstack does not support the definition of permissions and roles. Therefore, Aragon is more flexible as it satisfies all the above requirements for DAO development.

Besides, based on our findings searching the Web (blockchain-based project websites, documentation, social media and forums, and overall dissemination), the Aragon community is the most active in the field of DAOs [20, 21]. Furthermore, this framework is the most developed and most widely adopted among developers.

In this vein, it is worth highlighting the case of 1Hive [22], one of the communities more concerned about DAOs and their adoption using the Aragon framework. 1Hive has released DAO profiles in Apiary [23], their DAO explorer. Apiary aims to help people explore and understand the Aragon ecosystem indexing all the Aragon organizations and Aragon apps on the Ethereum mainnet network and listing all the results in the Apiary Browser [24]. 1544 Aragon organizations were indexed in this browser at the time of this writing. Note in comparison, DAOstack has ~60 public DAOs; and Colony does not provide a number at this time. [25, 26].

Table 1. Comparison of Aragon, Colony and DAOstack.

Mechanism	Aragon	Colony	DAOstack
Token	✓	✓	✓
Reputation	✓	✓	✓
Funding	✓	✓	✓
Permissions	✓	✓	✗
Voting system	✓	✓	✓
Organization templates	✓	✗	✗
New governance models	✓	✗	✗

4. DAO development: Task management case study

Although we have implemented several DAOs focused on testing and validating the Aragon framework, the case study in this section aims to be just complex enough to illustrate DAO development using the Aragon framework, as it shows how to incorporate new functionality. Thus, in this section the development process of an example Aragon DAO will be described.

In order to facilitate replicability, and to exemplify the current state of the framework, we will make explicit the points in the development process that were more challenging due to the lack of resources, documentation or relevant issues in the framework. In addition, this detailed description may serve well to computer scientists and software developers intending to use Aragon for practical purposes.

The objective of the case study is to implement a DAO that supports collaborative activities of researchers, located in different countries, who participate in common research projects. The DAO has to include a Dapp that manages those tasks necessary to obtain the deliverables associated with a specific project.

Although a formalized overview of the Aragon architecture is missing in its available documentation, as mentioned earlier, we have experimented with some implementations, and we have modelled an approach in order to formalize its understanding, that is shown in Fig. 1 as a UML class diagram.

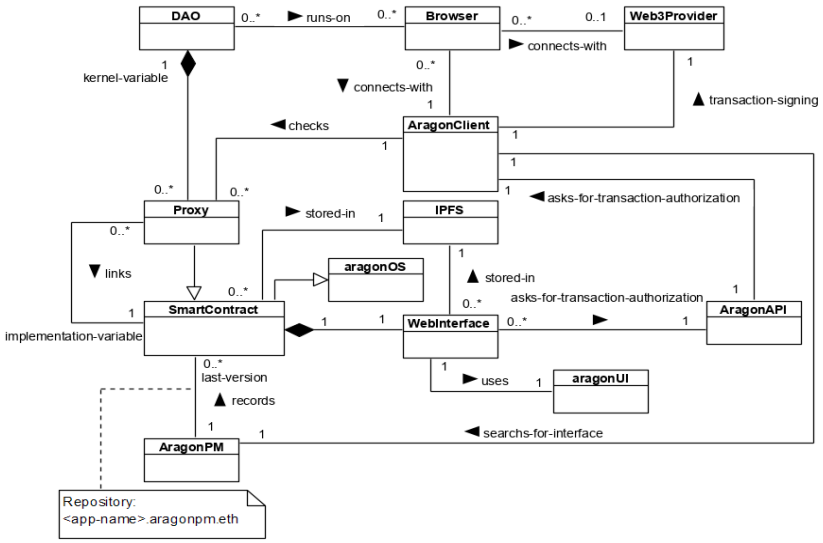

Fig. 1. General overview of the Aragon framework architecture, in an UML class diagram.

The process to implement a DAO with Aragon consists of a set of several steps that starts with the installation of several modules. This phase is apparently easy to tackle, but in practice it can be cumbersome, as different confusing errors may arise throughout the process.

We followed the same steps in different machines and using different operating systems, and the results and errors obtained were different in each case. Most of all were concerned about connection failures, versions that were incompatible among them and missing modules.

Therefore, we had to exert a significant amount of effort and apply different mechanisms in each case in order to solve these errors. In this sense, it was difficult to find related documentation that explains how we have to proceed in such situations.

An Aragon project consists of four main elements that determine the phases for the Aragon app development process: *Aragon Buidler plugin* [27], *configuration files*, *contracts*, and *app*.

The first stage consists in obtaining the code template of our project. The Aragon code template is conformed to the Aragon Buidler plugin, which provides the basic artifacts that constitutes an Aragon app for our DAO.

The second stage consists in filling the configuration files which include, among others, the Aragon-specific metadata and web-specific configurations for our Aragon app, for example, definition of the user roles. This requires some working around as they are not well documented with enough detail.

In the third stage, new smart contracts associated with the Aragon app are created and deployed using aragonOS. The smart contract defines the permission roles for the new functions, which in the particular case study for a project management DAO are: (i) create a task; (ii) delete a task; (iii) change the priority of a task; and (iv) mark a task as complete.

The fourth stage consists in generating the front-end (i.e., web user interface), which is implemented as React components [28], and integrated with aragonUI [29]. This phase requires a good understanding and knowledge in the use of the Aragon React components. In this case, although the documentation in Aragon is well structured and defined, we suffered from a lack of references and support to tackle it for several issues. For example, we had several problems when we tried to apply some additional styles and behavior to some Aragon UI controls since they do not run depending on the version of the framework and the new updates. In addition, it is not easy to include other UI controls apart from those provided by the Aragon framework.

Finally, once we have implemented and deployed our Aragon app in the Ethereum network, the last two steps are to include this application in our DAO through an Aragon proxy.

To do that, first, we have to create our DAO following the Aragon organization template that matched with our governance or business model. In our case, we selected the category of 'Reputation'. That is, an organization that uses non-transferable tokens to represent reputation (i.e., reputation-weighted voting is used to make decisions, and one reputation token equals one vote).

Second, we have to execute several Aragon commands. This process required some expertise on the Aragon platform in order to perform complex tasks of permissions assignments and resolve several conflicts and unknown errors that usually arise during the integration of a DAO with the implemented Aragon app.

5. Results and conclusions

The implementation of a DAO from scratch is a complex task for any organization. In this paper we have presented a preliminary analysis of the three most popular software frameworks oriented to the development of DAOs: Aragon, Colony and DAOstack. We have shown a comparison among these three frameworks in relation to the main functionalities associated with the process of building DAOs.

Since Aragon is currently the most complete and widely used, in order to provide more details on the challenges on building DAOs with current frameworks, we have presented a case study using the Aragon framework. Through this case study, we have piloted DAO development using Aragon, and thus we may highlight the benefits, limitations and problems that developers face when adopting it.

We have realized that there are still many issues that make its usage far from reaching an acceptable level of maturity. The main findings include:

- The creation of a DAO involves a lot of mechanisms that are not easy to understand and follow, even when using a framework such as Aragon, which intends to facilitate this task.
- The maintenance and adoption of smart contracts by organizations is a complex process since a deployed smart contract's code cannot be directly changed (i.e., the code of a smart contract is immutable, as any data recorded in a blockchain). For example, Ethereum still faces this issue, which is not yet solved in Ethereum 2.0. However, using different smart contract versions enable upgradability,

using a proxy contract to point to the last version. Aragon supports upgradability through aragonCLI commands which significantly facilitates the task.

- Aragon provides more flexibility since it enables the selection of a specific version for a Dapp through aragonPM.
- Aragon still has to address several DAO programming issues and some modules of the framework are not working properly across versions and operating systems.
- There is no support for multiple languages (internationalization and localization) in Aragon, only English, which is surprising in today's software landscape.
- Due to new requirements and evolution of Ethereum, the Aragon code templates often change without being backward compatible (i.e. losing support to DAOs that have been implemented with previous versions of the templates).
- More useful and detailed documentation is missing for both expert and non-expert blockchain programmers. It is often difficult to understand all the involved files in the implementation of a DAO in terms of the Aragon framework since the documented architecture provided is generally insufficient.

Even a simple case study as the one illustrating this article, shows that the support for the understanding, generation and implementation of DAOs in the context of blockchain-based solutions remains a challenge.

Moreover, those not completely familiarized with blockchain and Ethereum, or who have not yet implemented a Dapp before outside available frameworks, will have a steep learning curve, on top of the inconsistencies and failures that could appear as a consequence of the inevitable and logical evolution of these frameworks.

In summary, even though Ethereum (in 2014) and DAO frameworks (in 2016) are aimed to facilitate the development of DAOs in multiple ways, we think that there is still a long way to provide robust, usable and reliable software tools for their development. The available frameworks provide different approaches, conceptions and tools for building DAOs, without attempting to share understandings or provide interoperability.

6. Acknowledgment

This work was partially supported by the project P2P Models (https://p2pmodels.eu) funded by the European Research Council ERC-2017-STG [grant no.: 759207]; and by the project Chain Community funded by the Spanish Ministry of Science, Innovation and Universities [grant no.: RTI2018-096820-A-100]. The authors would also like to thank Paulo Colombo at UCM for the useful feedback.

7. Referencias

1. Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.* https://bitcoin.org/bitcoin.pdf (Retrieved April 16, 2021).
2. Swan, M. (2015). *Blockchain: blueprint for a new economy*. First edition. ed. O'Reilly, Beijing : Sebastopol, CA.

3. Antonopoulos, A.M.; Wood, G. (2019). *Mastering Ethereum: building smart contracts and DApps*. First edition. ed. O'Reilly, Sebastopol, CA.
4. Buterin, V. (2014a). *Ethereum: A Next-Generation Cryptocurrency and Decentralized Application Platform*. Bitcoin Magazine. https://bitcoinmagazine.com/business/ethereum-next-generation-cryptocurrency-decentralized-application-platform-1390528211 (Retrieved April 16, 2021).
5. Buterin, V. (2014b). *A next-generation smart contract and decentralized application platform*. https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf (Retrieved April 16, 2021).
6. El Faqir, Y.; Arroyo, J.; Hassan, S. (2020). *An overview of decentralized autonomous organizations on the blockchain*. Proceedings of the 16th International Symposium on Open Collaboration, OpenSym 2020. Association for Computing Machinery, New York, NY, USA, pp. 1–8. https://doi.org/10.1145/3412569.3412579
7. Hassan, S. (2017). *P2P Models white paper. Decentralized Blockchain-based Organizations for Bootstrapping the Collaborative Economy*. https://www.dropbox.com/s/c9lyx0r6lq3fw7p/whitepaper_p2pmodels.pdf?dl=0 (Retrieved April 16, 2021).
8. MolochDAO (2021). https://www.molochdao.com/ (Retrieved April 16, 2021).
9. Aragon main site (2020). https://aragon.org/ (Retrieved April 16, 2021).
10. Colony (2020). https://colony.io/ (Retrieved April 16, 2021).
11. DAOStack (2020). https://daostack.io/ (Retrieved April 16, 2021).
12. Ethereum Docs (2020). *Ethereum Classic Technical Reference*. https://ethereum-classic-guide.readthedocs.io/en/latest/docs/world_computer/smart_contracts.html (Retrieved April 16, 2021).
13. Herian, R. (2018). *Legal Recognition of Blockchain Registries and Smart Contracts*. http://oro.open.ac.uk/59481/ (Retrieved April 16, 2021).
14. Filippi, P.D.; Hassan, S. (2016). *Blockchain technology as a regulatory technology: From code is law to law is code*. First Monday. https://doi.org/10.5210/fm.v21i12.7113
15. MetaMask (2020). https://metamask.io/ (Retrieved April 16, 2021).
16. Bell, S. (Ed.) (2002). *Economic governance & institutional dynamics*. Oxford University Press, South Melbourne, Vic. ; New York.
17. Aragon (2020). Aragon Dev. Portal. https://hack.aragon.org/ (Retrieved April 16, 2021).
18. Rea, A.; Kronovet, D.; Fischer, A.; du Rose, J. (2020). *Colony Whitepaper*. Colony Blog. Retrieved November 9, 2020, from https://blog.colony.io/whitepaper-update/ https://colony.io/whitepaper.pdf (Retrieved April 16, 2021).
19. DAOstack White Paper (2018). https://daostack.io/wp/DAOstack-White-Paper-en.pdf (Retrieved April 16, 2021).
20. What's a DAO? (2020). https://aragon.org/dao (Retrieved April 16, 2021).
21. Powered By Aragon (2020). https://poweredby.aragon.org/ (Retrieved April 16, 2021).
22. 1Hive (2020). https://1hive.org/ (Retrieved April 16, 2021).
23. Honey Pot (2021). *1Hive*. https://1hive.org/introducing-profiles-on-apiary/ (Retrieved April 16, 2021).
24. Apiary Explorer (2020). https://apiary.1hive.org (Retrieved April 16, 2021).
25. DAO analyzer (2021). http://dao-analyzer.science/. (Retrieved April 16, 2021)
26. DAO analyzer GitHub repository (2021). https://github.com/Grasia/dao-analyzer. (Retrieved April 16, 2021)
27. Aragon Buidler plugin (2020). *Aragon One Blog*. https://blog.aragon.one/buidler-plugin/ (Retrieved April 16, 2021).
28. Banks, A.; Porcello, E. (2017). *Learning react: functional web development with react and redux*. First edition. ed. O´Reilly, Bijing Boston Farnham.
29. aragonUI (2021). https://ui.aragon.org/ (Retrieved April 16, 2021).