The following is a quick exposition of an idea I had for improving the ICO model by merging in some of the benefits of DAOs, but doing so in a way that minimizes complexity and risk.

[

svg

1002×499 36 KB

](https://ethresear.ch/uploads/default/original/1X/d422da54d200e5dc8e119de3a65ec885a542b4c3.png)

The idea is as follows. A DAICO contract is published by a single development team that wishes to raise funds for a project. The DAICO contract starts off in “contribution mode”, specifying a mechanism by which anyone can contribute ETH to the contract, and get tokens in exchange. This could be a capped sale, an uncapped sale, a dutch auction, an interactive coin offering, a KYC’d sale with dynamic per-person caps, or whatever other mechanism the team chooses. Once the contribution period ends, the ability to contribute ETH stops, and the initial token balances are set; from there on the tokens can become tradeable.

After the contribution period, the contract has one major state variable: tap

(units: wei / sec), initialized to zero. The tap determines the amount per second that the development team can take out of the contract. This can be implemented as follows:

```
tap: num(wei / sec) lastWithdrawn: timestamp # Make sure to initialize this to the contribution period end time

@public def withdraw(): send(self.owner, (block.timestamp - self.lastWithdrawn) * self.tap) self.lastWithdrawn = block.timestamp

@private def modify_tap(new_tap: num(wei / sec)): self.withdraw() self.tap = new_tap
```

There is also a mechanism by which the token holders can vote on resolutions. There are two types of resolutions:

- Raising the tap
- Permanently self-destructing the contract (or, more precisely, putting the contract into withdraw

mode where all remaining ETH can be proportionately withdrawn by the token holders)

Either resolution can pass by some kind of majority vote with a quorum (eg. yes - no - absent / 6 > 0

). Note that lowering

the tap by vote is not possible; the owner

can voluntarily lower the tap, but they cannot unilaterally raise it.

The intention is that the voters start off by giving the development team a reasonable and not-too-high monthly budget, and raise it over time as the team demonstrates its ability to competently execute with its existing budget. If the voters are very unhappy with the development team’s progress, they can always vote to shut the DAICO down entirely and get their money back.

Game-theoretic Security

Any vote is subject to 51% attacks, bribe attacks and other game-theoretic vulnerabilities, and any ICO is subject to the risk that a team will be irresponsible or simply a plain fraud. However, in a DAICO these risks are minimized, requiring both the developer and the vote to be compromised to cause any real damage:

- 51% attack maliciously raises tap
- honest developer can just lower the tap again, or not claim excess funds
- Developer starts spending funds on lambos instead of real work
- voters can prevent much of this by not raising the tap too much too quickly, but if it happens anyway they can vote to self-destruct
- 51% attack maliciously self-destructs
- honest developer can just make another DAICO

Notice how two of the potentially most harmful kind of 51% attack: (i) sending funds to some other third party chosen by the attacker, and (ii) lowering the tap to keep funds stuck in the contract forever are both simply disallowed by the mechanism.

Variations

- Denominate the tap

in usd / sec, and use some price feed

- Use DAI as the funding currency instead of ETH
- Experiment with mechanisms other than simple voting