Ethereum Classic Marketplace ICO

S.J. Mackenzie[a)]

(Dated: 13 January 2018)

Currently it is unintuitive for non-technical end users to interact with published cryptocontracts. For this reason cryptocontracts will not easily reach mainstream. End users expect a convenient one-click platform that delivers user friendly yet secure third party applications which are designed to use cryptocontracts.

Keywords: Ethereum Classic, flow-based programming, dataflow, fractalide, HyperCard

The *Ethereum Classic Marketplace ICO* aims to provide a rich set of development and runtime tools (Hyperflow), a marketplace for third party vendors to sell applications to end users (Fractalmarket) and a set of components which grant applications seamless integration with the Ethereum Classic blockchain (Etherflow).

I. FRACTALIDE

Fractalide is a programming language which implements a flow-based programming language whereby every component is deterministically reproducible and entirely reusable. Although the ICO is meant to focus on the higher level items such as Hyperflow, Etherflow and Fractalmarket there is some important infrastructure level work that needs to go into Fractalide to enable smooth front end visuals. This allows for the clean implementation of the Card Paradigm as elaborated on in the Master Thesis.

The major concepts introduced by Fractalide are Reusable and Reproducible components. The entire platform hinges on these first principle concepts.

A. Reusability

Each Fractalide component is reusable because it has one unifying composable interface. The data that transmits into the component is the invariant of the system.

B. Reproducibility

Each Fractalide component and all their dependencies are capable of being reproduced. Indeed an entire hierarchy of components can be reproduced on another machine. Each component has a universally unique name yet when presented to the programmer the component's name is human readable. Components with the same name but different versions can exist on the same file system side-by-side without conflicting with each other.

a)Electronic mail: setori88@gmail.com.

II. HYPERFLOW

Hyperflow is a Fractalide module or library that implements a variant of HyperCard. HyperCard was the inspiration for the Delphi and Microsoft Visual Basic Rapid Application Development platforms which allowed non-technical users to quickly put together programs to solve their problems. It significantly lowered the bar allowing people with little to no programming experience to get up and running quickly.

HyperCard allowed one to drag-and-drop Graphical User Interface components onto a canvas called a Card. Each "screen" or canvas was a Card and a number of Cards formed a Stack. A good analogy would be; a Card is a webpage and a Stack is a website. A built-in programming language called HyperTalk allowed one to write logic to move from Card to Card.

Though a few concepts need retaining, removing and replacing to upgrade HyperCard to a modern day HyperCard suitable for interacting with the blockchain and implementing advanced applications.

A. Concepts to remove from HyperCard

1. HyperTalk

HyperTalk was HyperCard's programming language. It was an English like programming language, with the below snippet being an example.

```
on mouseUp
  put "100,100" into pos
  repeat with x = 1 to the number of card buttons
    set the location of card button x to pos
    add 15 to item 1 of pos
  end repeat
end mouseUp
```

HyperTalk was not a particularly powerful language, although it did have an avid following, it is not sufficiently expressive to write high end applications suitable for solving problems of today.

Most importantly HyperTalk isn't modularized and cannot easily compose together. As we have a hard requirement of using reusable and reproducible infrastructure in the small and in the large. We'll be replacing

HyperTalk with the previously mentioned component oriented programming language called Fractalide.

2. HyperCard lacked network access

This was the last nail in HyperCard's coffin, it's the fundamental reason why HyperCard died. Attempts to rescue HyperCard were underway at the time by adding networking capabilities, but the task proved too difficult and Steve Jobs eventually called the executioner's axe to HyperCard's head. Had the system been designed from the ground up with full network support, HyperCard would have expanded beyond the boundaries of Apple, just as the HTTP browser became prevalent. Indeed it could well have been the browser we use today.

As every component in Hyperflow is built from first principles of reproducibility and reusability it becomes trivially easy for a programmer to type the name of a component and the package manager and build system will immediately make that component available for use on their system. Packaging their component (or component hierarchy) and distributing it is equally easy.

B. Concepts from HyperCard to keep

1. Fast switching between run mode and design mode

HyperCard had a zero compilation step between run mode and design mode. So when in design mode the experimental code could be triggered to run immediately by clicking the run mode button. When in run mode GUI widgets could be manipulated and moved about again by clicking on the design mode button.

This was probably one of the most important features which is quite hard for Hyperflow to replicate for a number of reasons. For third party component hierarchies already available on the file system, we'll be able to achieve this "zero compilation" switch, but in the case the programmer calls in a component or entire hierarchy of components not presently available on the file system the transition will not be smooth. The package manager must download either the source code of all those components and build them or download the pre-compiled binaries and wire them into Hyperflow during the switch between run mode and design mode. Given we hope the Fractalmarket will host many component hierarchies we must accept these limits.

III. FRACTALMARKET

Fractalmarket intends to provide fair compensation for contribution. Meaning that if you develop and maintain a particular set of components you may optionally charge a fee for those components and wherever they are used, be it in other applications, you, the developer will be paid for your work once an end user purchases that component.

Hence if the hierarchy/application developer reuses as many components as possible, the probability of the end user already having purchased many of those components is higher and thus the total price tag of your particular hierarchy will be reduced. This way you'll be able to reduce the footprint of what's needed to be downloaded, compiled, installed and paid for, making you more competitive than a programmer that copies code and republishes.

We aim to have nice metrics which demonstrate the reusability factor of each application hierarchy.

The Fractalmarket supports a currency called FRACTAL which enables trade between component buyers and sellers. FRACTAL will be the only accepted currency on the Fractalmarket for the foreseeable future.

IV. ETHERFLOW

Etherflow is a set of reusable and reproducible components that allow for seamless interactions between Hyperflow applications and the blockchain. The code will heavily leverage ETCDEV team's work.

These components will allow users who wish to create domain specific applications such as a blockchain insurance application to present a nice user facing application with all the GUI components that can easily interact with the blockchain via reusable blockchain components.

As users will be downloading and paying for components created by ETCDEV, helping ETCDEV become more sustainable and assisting them to continue contributing to the Ethereum Classic community.

V. CONCLUSION

We hope the contributions "Ethereum Classic Marketplace ICO" offers the Ethereum Classic community are seen as valuable and worthwhile contributing towards. Creating a platform that enables a whole new layer of blockchain enterprises and services that cut out the middle man along with the high degree of component reusability hopes to reduce costs greatly.