Ethereum Classic Infrastructure ICO

S.J. Mackenzie[a]

(Dated: 23 July 2017)

Cryptocontracts will never reach mainstream as long as we keep pushing code to a blockchain address and announcing it to be an insurance company. What is expected; an application supports that blockchain address by interacting with it as defined by a set of business rules encoded into the application. The end user should barely see the cryptocontract address and primarily interacts with the application.

Keywords: ethereum, flow-based programming, dataflow, fractalide, HyperCard

The *Ethereum Classic Infrastructure ICO* aims to provide a rich set of development and runtime tools, a marketplace and a set of components which allow for other applications to have seamless integration with the Ethereum Classic blockchain. Only after this layer of infrastructure is in place can we support things like the Internet of Things.

I. FRACTALIDE

Fractalide is a programming language co-developed by the author, it 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 abstractions such as Hyperflow, Etherflow and Fractalmarket there is some important infrastructure level work that needs to go into Fractalide to enable the above items. Namely, a Purescript implementation of a flow-based programming domain specific language should be included into Fractalide to allow 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.

A. Reusability

A 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

A Fractalide component is capable of reproducing not only itself but all the dependencies it might have, be it OpenSSL or other Operating System level dependencies. Indeed the entire hierarchy of dependencies can be reproduced on another machine. Each component has a universally unique name yet when presented to the programmer the component name is a memorable one. Indeed components with the same name but different versions can exist on the same file system side-by-side without conflicting with each other.

II. HYPERFLOW

Hyperflow is a Fractalide module or library that implements a variation 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" 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
```

[a]Electronic mail: setori88@gmail.com.

HyperTalk was not a particularly powerful language, although it did have an avid following, it was not sufficiently expressive to achieve high end applications suitable of solving large expensive 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.

Using this approach it becomes possible to reproduce entire hierarchies of complex Hyperflow applications onto other machines yet barely lifting a finger.

B. Concepts from HyperCard to keep

1. Fast switching between run mode and develop mode

HyperCard had a zero compilation step between run mode and develop mode. So when in develop 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 develop 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 3rd 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 immediately 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 develop mode. Given we hope the Fractalmarket will host many component hierarchies we must accept these limits just as we accept the cost of downloading every webpage repeatedly. At least with this architecture many components will already be available on your system and are being reused.

There are certain advantages of running silo'ed software as is the case of HyperCard, one doesn't need to consider 3rd party libraries and how they affect the user experience.

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.

Fair compensation for contribution is a tricky problem particularly when someone copies your open source code then puts it up on the marketplace. Just as Netflix has found a market by being more convenient than The Pirate Bay, so the programmer just needs to remember the name of the component (or name of the component hierarchy) and that component (or hierarchy of components) will be reproduced on the machine instead of manually copying code, putting them into hierarchies, naming them and publishing them.

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. It's based on Dexaran's ERC 223 token standard, which ensures that transactions to unsupported addresses do not go through.

IV. ETHERFLOW

Etherflow is a set of reusable and reproducible components that allow for seamless interactions between Hyperflow applications and the blockchain. Most likely they will heavily leverage ETCDEV's work. Close collaboration between ETCDEV and the Etherflow team is inevitable.