Proposal for a Modern Software Ecosystem

Contents

Overview

> ***Note***
>
> Need area for a realistic road-map of finished products !!!!!!!!!!!!!!!!!!!!!!!!!!

Bob Dylan once wrote a song titled "Everything is Broken". We take that notion and consider it quite apropos for the conventions in today's software usage, sales, production and execution. A participant can find inefficiencies and un-warranted complexities practically everywhere in today's environment. We ask the reader to take a moment and think of their own experiences with software. Have you ever found yourself frustrated? Have you ever considered some aspect overly complicated, un-intuitive or in-flexible and thought to yourself "it shouldn't have to be this way"?

An end-user of software might have thought of how hard it is to get familiar with a software product or that the documentation and help system are simply not much help. They might have wished for a feature they needed to be available or that the there are too many features packed into the product. Small business owners may find navigating the obstacle course of software system selection and integration impossible

for mere mortals. Software engineers may have thought of many areas of foolishness that surround their profession. The lists of areas ripe for innovation surrounding the software ecosystem are quite abundant.

Other aspects of concern and inefficiency can be seen in the marketplace itself. Golieth sized companies hold monopolistic power over software marketplace silos. These concentrated marketplaces take large percentage commissions from a lowly software producer's product sales. They dictate rules that are at times overly stringent or simply used to control the domination they hold in the market.

The readers of this paper are asked to consider very thoughtfully what they themselves, specifically, would have improved, because it is that specific item or list of items that will be fixed by what is proposed in this paper and its addendums.

This paper is currently in its first draft and is itself an open source project for inspired persons to contribute their ideas to. Eventually it will reach a 1.0 status at which point it will have the appearance more of a specification than of merely a white paper. At this point I encourage you to envision the possibilities of a collaborative effort to change the current paradigm surrounding the production, consumption and usage of software. You are encouraged to join this effort to modernize the entire software ecosystem via open source software. Please become a contributing member of the CodeDepot Community.

> ***Note***
>
> Say: Bring _your_ innovations and implement them.

Introduction

Software is ubiquitous in our lives whether we are users or producers. Yet, even a less than astute observer can see that the entire ecosystem is ripe for innovation. In this paper we will begin a conversation of the direction of the needed improvements and we ask the community to participate. This community that is mentioned, are the people that engage with software whether it be in its production, sales or usage; whether they are casual or power users. Having discussion and contribution globally and easily, will ensure that the best solutions of problems we all identify, will be addressed most thoroughly and efficiently.

Below are the introductions to six foundational pillars of a proposed software ecosystem that may forever change the entire paradigm of our engagement with software. Following each introduction is a link to a addendum paper that will explain that particular component in greater detail. What is presented at this time are merely proposals that will be refined by any willing individual participant and all the participants in whole. The paper also will introduce other worthy aspects and notations pertinent to the execution of this proposal.

The CodeDepot Foundation

Together, an engaged society will undoubtedly find the best solutions to issues that face them. Over the past 40 years, the open source community has shown that open participation most often provides better results than that of a closed system. We present here a decentralized organization where every member of the community has the opportunity provide direction through proposals, discussion and ultimately to that of solutions. We suggest that Decentralized Autonomous Organizations, a new concept now available through cutting edge technology, will provide many benifits over conventional, centralized organizational structures where they are applicable.

Overview

The CodeDepot Foundation is the working name of a proposed open governance model for the CodeDepot ecosystem; where directives and decisions are decentralized and democratized amongst the entire community [1]. This notion of decentralized governance is aimed at stemming the problems of any centralized governing system that naturally creates levels of control [2].

> Bitcoin's underlying technology of the blockchain [3], brings a radical departure from the governance model of the old world. It follows trends of decentralization that have emerged through the internet in recent years including Bittorrent, Free Software and Open Source Movements and collaborative production platforms like Linux and Wikipedia. The essence of this game-changing invention is distributed trust (no need for third party reconciliation). Bitcoin solves the scaling issue of trust. With its decentralized security, we can now create a more open and inclusive society at a global scale.
>
> Bitcoin's decentralized system opens a door to a new paradigm where people can choose to abide by a protocol of consensus which is a different from the logic of domination and control of a centralized paradigm. Technology can't solve everything. Technology is just a tool. It always needs to be accounted for by democratic consensus of people. Technology should never be used to replace human interaction and connection and it should be used only to enhance it. [4]

CodeCoin, The currency of the ecosystem

The invention of blockchain technology in 2008 has provided for the world a whole paradigm shift in financial and contractual mechanisms. It is truly a marvel that will disrupt many many incumbent institutions. Systems that once required intermediaries and centralized solutions can now be peer to peer and decentralized. Trust of persons and entities unknown can know be established without the need of 3rd parties for verification or validation. Moneys can now be sent digitally without any concern for fraud or corruption. Ability to make payments in fractions of a penny are now facilitated. These are truly amazing times and we as a community of software engagers can benefit from this remarkability

CodeCoin is the proposed crytpocurrency that will be used in this software ecosystem. It is essentially a token that will be used to monetize all the goods and services within the ecosystem. This cryptocurrency will be at the center of a well considered autonomous economic model of incentives and disincentives formulated by the community. As with other aspects of this proposal, an emphasis for thought leaders to be engaged in these models is vehemently sought after and strongly encouraged.

CodeDepot, The decentralized software market place and code repository

The core component of the ecosystem is a marketplace where users and producers engage. This itself is composed of two parts where developers will deposit code into the system and users will have an interface to utilize it.

> ***Note***
>
> http://usabilitygeek.com/empathy-and-technology-relationship-makes-good-design-better/

As described in the section for CodeChain, developers will be rights holders for what essentially are snippets of code called components in this paper's nomenclature. The assemblage of components yields other composite components or finally applications that end-users will utilize for their needs. Users will have many advantages over conventional software produced today including but not limited to:

- Flexibility
- Scalability

- Security
- Ease of use
- Accessibility

Mechanisms will be built-in that provide the end-user the ability to:

- Give instantaneous feedback.
- Directly request alterations or needed features.
- Intuitive, complete documentation and autonomous tutelage.
- Hire affordable freelancers directly from their user-interface * For asistance or instruction * For software developers to provide any possibly needed customization.

> ***Note***
>
> Mention that CD is also for technical orientated content like documentation, tutorials, articles, blogs, audio and video productions.

CodeChain, The decentralized processing engine

The problems with the conventional programming paradigm are numerous. In fact, at the time of the initial draft of this paper, Querying Google's search engine for "The problems with programming languages" produced nearly 72 million results. It seems fruitless to itemize these problems, let alone the problems of the entire software centric paradigm, as a comparison to what is here proposed. Instead as an introduction we ask the following questions.

1. What if snippets of software were in essence Lego like reusable components that just snapped together into a desired structure?
2. What if once a component or a structure of components was created, it would never have to be created again by anyone? Yes, ever again as in the literal sense of forever.
3. What if once a component was created it would be shared by every piece of running software in the world that required it?
4. What if an executing software system, even a mission critical system, could be altered or replaced without even a nanosecond of down-time.
5. What if it did not matter which programming language was used to create an individual component and that any component could seemlessly communicate with others?
6. What if a software producer's time-to-market was reduced by a hundredfold?
7. What if a software's execuation was most reliable and the most possibly secure from intrusion?
8. What if a software user's privacy was held in the highest regard.

CodeChain, it will be shown, is a system that could and will provide these desirable properties as well as others.

Overview

CodeChain is a decentralized [5] operating system [6] which at its core reflects the philosphy of component based software engineering (CBSE) [7] and is entirely inspired by J.Paul Morrison's [8] Flow-Based Programming (FBP) [9]. The concepts of FBP are improved with secure decentralized computing, communication and database models from various sources including BitCoin [10], BitTorrent [11]. CBSE exhibits the very desirable property of loose coupling [12]. FBP, a particular form of dataflow programming [13] extends loose coupling to define bounded buffers, information packets with defined lifetimes, named ports, and most importantly a separate (lazy) definition of communication connections [14].

FBP: The Architectural Inspiration

> ***Note***
>
> The following is taken from J.Paul Morrison's Website [#]_.

Flow-Based Programming is a programming paradigm that uses a "data factory" metaphor for designing and building software applications. Applications are defined as networks of "black box" processes, which exchange data across predefined connections by message passing, where the connections are specified external to the processes. These black box processes can be reconnected endlessly to form different applications without having to be changed internally. FBP is thus naturally component orientated.

It views an application not as a single, sequential process, which starts at a point in time, and then does one thing at a time until it is finished, but as a network of asynchronous processes communicating by means of streams of structured data chunks, called "information packets" (IPs). In this view, the focus is on the application data and the transformations applied to it to produce the desired outputs. The network is defined externally to the processes, as a list of connections which is interpreted by a piece of software, usually called the "scheduler".

The processes communicate by means of fixed-capacity connections. A connection is attached to a process by means of a port, which has a name agreed upon between the process code and the network definition. More than one process can execute the same piece of code. At any point in time, a given IP can only be "owned" by a single process, or be in transit between two processes. Ports may either be simple, or array-type. It is the combination of ports with asynchronous processes that allows many long-running primitive functions of data processing, such as Sort, Merge, Summarize, Collate, etc., to be supported in the form of software black boxes.

Because FBP processes can continue executing as long they have data to work on and somewhere to put their output, FBP applications generally run in less elapsed time than conventional programs, and make optimal use of all the processors on a machine, with no special programming required to achieve this.

The network definition is usually diagrammatic (see: 'Component Based Programming' below), and is converted into a connection list in some lower-level language or notation. FBP is thus a visual programming language at this level. More complex network definitions have a hierarchical structure, being built up from subnets with "sticky" connections .

FBP has much in common with the Linda language in that it is, in Gelernter and Carriero's terminology, a "coordination language": it is essentially language-independent. Indeed, given a scheduler written in a sufficiently low-level language, components written in different languages can be linked together in a single network. FBP thus lends itself to the concept of domain-specific languages or "mini-languages".

FBP exhibits "data coupling", described in the article on coupling[*] as the loosest type of coupling between components. The concept of loose coupling is in turn related to that of service-oriented architectures, and FBP fits a number of the criteria for such an architecture, albeit at a more fine-grained level than most examples of this architecture.

FBP promotes high-level, functional style of specifications that simplify reasoning about system behavior. An example of this is the distributed data flow model for constructively specifying and analyzing the semantics of distributed multi-party protocols.

Limitations of FBP

At a superficial level, FBP is an ideal programming paradigm that offers quite a few benefits over conventional paradigms. At scale though, there is a limiting condition of context switching, especially so on conventional general purpose CPUs [#]_. For an FBP paradigm at scale, a point will be reached where the number of context switches on a single machine CPU, overwhelms the system and causes notable latency. On average context switching costs approximately 30 microseconds of overhead per occurrence. One benchmark of the theoretical limitations of context switching has an upper bound of 18.75% of CPU cycles wasted due to context switching. Generally, optimal CPU use, is to have the same number of worker threads as there are hardware threads when a process is CPU bound, whereas I/O bound permit more [#]_. These considerations puts the FBP paradigm at very much a disadvantaged ideal of maximal efficiency.

Component Based Programming

In order to overcome the conditional limitations of context switching per processing node in an strictly FBP paradigm, we provide here an area of consideration to help maximize the efficacy of the CodeChain system. The term Component-based Programming (CBP) is coined here for the purpose of a enlisting a stronger emphasis on components over that of data flow as it is for FBP.

The concepts fundamental to FBP (autonomous black-box components loosely coupled via lazy linkage) can be easily considered at the various phases of the compilation stack prior to execution. Essentially, what this means is that we can remove the constraints from that of each component needing to be its own execution process or thread, yet still be most loosely coupled. We can redefine components to that of a virtual model that can then be implemented by encompassing one or all of the compilation's translation stages prior to execution.

1. Source code
2. Semantic analysis
3. Intermediate Representation (IR) code and its linkage
4. Machine code and its linkage
5. Just-In-Time compilation or interpreter engine

Another most exciting and promising consideration is to apply the notion of CBP to include that of speciality hardware processors like that of GPGPUs [#]_. GPGPUs provide a processing model of thousands concurrently executing threads. Utilizing these high-scale concurrent processors, one can imagine the promise of the original FBP concept of inter-communicating processes/threads without the extreme burdens imposed when merely targeting that of a CPU architecture.

Decentralized Network

Processor Node

Secure Computing

Micro-payment Calculation

User Interfaces are Very Flexible Thin Clients

WorkSource

WorkSource is the proposal for a open governence, decentralized, peer to peer marketplace for end-users to hire freelancers. It will employ modern cutting edge technology for monetization, accounting, contractual obligation and in the case needed, arbitration. The most prominent aim is to incorporate very simple access and functionality directly into the CodeDepot user-interfaces. Unlike current freelance market places, freelancers will be made to feel as equals and not of a second class as compared to employers.

Non-technical users of software often find themselves in need of instruction or in need of customization. By incorporating direct and easy contact with software professionals, the users needs can be addressed painlessly, immediately and reliably.

Technical users find themselves paying exorbitant fees to hire developers at conventional centralized services. Often the employer will find these services confusing, frustrating and simply inadequate.

Freelancers will often find that scanning and applying for jobs simply is too time consuming.

Overview

Current freelance and other work sourcing like exchanges are usually run by a centralized corporate entity that enjoy a significant percentage of the cost of the transaction together with collecting monthly fees. In a decentralized; self-governed; peer-to-peer (P2P) marketplace there isn't any centralized entity, just a community of colleagues and clients. Freelancers enjoy the near entirety of the proceeds of their transactions without some third party dipping their greedy hands in.

New technologies, most significantly bitcoin's blockchain, have now enabled P2P marketplaces to thrive unencumbered by any need of a centralized entity or 3rd party. The need for trust is virtually eliminated providing free, flat, P2P markets.

WorkSource will be decentralized community effort that will provide reliable sources of service providers to those that need such services. Current cryptocurrency and other new technology make it possible to enable most efficient market ecosystems where trust and incentive/disincentive mechanisms are automated and built right in to the platform. This together with ideas generated and implemented by the community will make the platform most desirable and efficient to participate in.

Introduction

> WorkSource will be a superset community of sectoral communities for instance that of the software sector, namely CodeDepot [#]_.

CodeSweet

A programmers toolbox is most often burdensome and time consuming to get productive with. It could be argued that every tool in use by engineers is in some way problematic or simply incomplete. If every aspect of every tool and its interface were a component, then the programmer could fashion their tool and hence their toolbox to be just the way they liked it.

CodeSweet will be a component based toolkit where engineers have the ability to add features that they deem worthy.. leaving any others behind. Features like automation, intuitive instruction and ease of use will be of strong focus.

> ***ToDo***
>
> Emphasize the CodeChain Toolbox and how the goal is to make software production unbelievably seemless !

Other aspects will include:

- The best documentation tools and interfaces
- Employing automation as much as possible
- Search and reference to be topped by no other
- Intuition
- Cutting edge compilation and translation chains

---

1 Decentralized Computing http://en.wikipedia.org/wiki/Decentralized_computing
2 Here we define an operating system as not including kernel duties of hardware interfacing.
3 Component-Based Software Engineering http://en.wikipedia.org/wiki/Component-based_software_engineering
4 J. Paul Morrison http://en.wikipedia.org/wiki/John_Paul_Morrison
5 Flow-Based Programming http://en.wikipedia.org/wiki/Flow-based_programming
6 Bitcoin http://en.wikipedia.org/wiki/Bitcoin
7 BitTorrent http://en.wikipedia.org/wiki/BitTorrent
8 Loose Coupling http://en.wikipedia.org/wiki/Loose_coupling
9 Dataflow Programming http://en.wikipedia.org/wiki/Dataflow_programming
10 Flow-Based Programming 2nd Edition http://www.amazon.com/Flow-Based-Programming-J-Paul-Morrison-ebook/dp/B004PLO66O
11 J.Paul Morrison's Website http://www.jpaulmorrison.com/fbp
12 Context Switch http://en.wikipedia.org/wiki/Context_switch
13 How Long Does It Take To Make Context http://blog.tsunanet.net/2010/11/how-long-does-it-take-to-make-context.html
14 General-purpose computing on graphics processing units http://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units